





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 D4.3

Final SEASON control plane infrastructure

Editor Vignesh Karunakaran (ADTRAN/ADVA)

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

Version 2.0

Date September 20 2025

Distribution PU

DISCLAIMER

This document contains information which is proprietary to the SEASON consortium members that is subject to the rights and obligations and to the terms and conditions applicable to the Grant Agreement number 101096120. The action of the SEASON consortium members is funded by the European Commission.

Neither this document nor the information contained herein shall be used, copied, duplicated, reproduced, modified, or communicated by any means to any third party, in whole or in parts, except with prior written consent of the SEASON consortium members. In such a case, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced. In the event of infringement, the consortium members reserve the right to take any legal action they deem appropriate.

This document reflects only the authors' view and does not necessarily reflect the view of the European Commission. Neither the SEASON consortium members, nor a certain SEASON consortium member warrant that the information contained in this document is suitable for use, nor that the use of the information is accurate or free from risk and accepts no liability for loss or damage suffered by any person using this information.

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

REVISION HISTORY

Revision	Date	Responsible	Comment
1.0	May 15, 2025	ADTRAN	Initial ToC version.
1.1	Aug 15, 2025	ADTRAN	First set of contributions.
1.2	Sep 1, 2025	ADTRAN	Final Contributions.
1.3	Sep 5, 2025	ADTRAN	Start Quality Review
1.4	Sep 29, 2025	INF-P	Quality check
2.0	Sep 30, 2025	ADTRAN	Final version for submission.

LIST OF AUTHORS

Partner	Name Surname
Adtran (ADVA)	Achim Autenrieth, Vignesh Karunakaran, Nikhil Dsilva
CNIT	Andrea Sgambelluri, Filippo Cugini, Kyriakos Vlachos, Mohammed Ismaeel
CTTC	Ramon Casellas, Ricardo Martínez, Carlos Efrén Hernández Chulde, Josep Maria Fàbrega
UPC	Luis Velasco, Marc Ruiz, Jaume Comellas, Josep Prat, Salvatore Spadaro
WINGS	Vasilis Tsekenis, Sokratis Barmpounakis
ACC	Stephen Parker, Ibrahim Babiker
TID	Oscar Gonzalez, Pablo Armingol
FiberCop	Roberto Morro
WEST	Carlo Centofanti, Andrea Marotta, Stefano Tennina
INF-P	Cátia Pinho

EXECUTIVE SUMMARY

The deliverable reports on the final SEASON control plane infrastructure with update in the design, implementation, and integration. The overall architecture includes the integration of different network domains: Radio Access Network (RAN), Passive Optical Network (PON), Internet Protocol over Wavelength Division Multiplexing (IPoWDM), MultiBand over Space Division Multiplexing (MBoSDM) and optical transport network segments. The deliverable reports on the final version of the components in each network domain with evaluation results. With initial design, implementation solutions, and intermediate results reported in deliverable D4.1 and D4.2, this report covers the final version of the control plane solution addressing all the four tasks (T4.1 – T4.4) in WP4 that includes, monitoring and streaming telemetry, infrastructure and domain control, multidomain control, and Artificial Intelligence/Machine Learning (AI/ML) solutions.

An overview of the SEASON control plane architecture is provided in Section 2, detailing both the proposed design and overall solution architecture developed throughout the project. With architecture definition fed from WP2, and developed components in WP4 (listed in Sections 3-5), the final version of the integrated architecture with suitable interface and control specifications is reported in WP5 for final demo activities. Details on the control plane elements including Software Defined Networking (SDN) agents, network domain controllers, and multi domain orchestration are provided in Section 3. The SDN developed aims to establish standardized definitions for control of devices and individual components, including data processing units, MBoSDM nodes, and optical pluggables. The section also provides details on controlling each network domain in the network (RAN, PON, IPoWDM, MBoSDM, and optical transport) with domain controller, and details on interfaces and control lifecycles. The service and network spanning multiple domains are managed through service and network orchestrators. The integration of controllers with the orchestrator, along with the overall network management is described.

Details on the components developed in WP4 to address planning, resource allocation and network operations are given in Section 5. Interoperability among network domains impacts service quality due to network modifications and service failures. To ensure availability and reliability, solutions such as digital twins and distributed intelligence are employed for performance estimation and fault management. Effective network operation and control are further enhanced through research in Network Development Operations (NetDevOps) and resource allocation, incorporating AI-assisted energy efficiency and latency-aware approaches, with evaluations conducted to assess their effectiveness.

Comprehensive telemetry solutions are proposed to enable effective network monitoring, presented in Section 5. These solutions intelligently aggregate data from optical and RAN domains, introducing innovative monitoring approaches and supporting AI/ML-based network management. Additionally, these solutions evaluate suitable protocols and investigate streaming capabilities to decrease the overhead in the network while ensuring a balanced scalable monitoring platform.

TABLE OF CONTENTS

1 INTRODUCTION 7
2 Overview of SEASON Control plane architecture for self-managed and autonomous networking 9
3 Control Plane Elements 11
3.1 SDN Agents and Packet/Optical Control 11
3.1.1 SDN Agent and Control of Data Processing Unit (DPU) 11
3.1.2 SDN Agent for IPoWDM nodes in ROADM-free networks 13
3.1.3 SDN Agent for MB Transceiver 14
3.1.4 SDN Agent for the MBoSDM node prototype 15
3.1.5 FlexTelemetry Agent 17
3.1.6 OpenROADM Agent 19
3.2 SDN Controllers 19
3.2.1 MBoSDM Optical Controller 19
3.2.2 Ensemble OLS controller 24
3.2.3 Optical Controller for P2P and P2MP pluggable devices (OpenXR) 27
3.2.4 SDM Passive Optical Network (PON) Controller 33
3.2.5 IPoWDM Controller 37
3.2.6 RAN Intelligent controller 43
3.3 Multi-domain Orchestration 46
3.3.1 Transport Network Orchestration 46
3.3.2 Cloud-native Service Orchestration 48
3.3.3 Integration with Transport Network Orchestration 56
4 Operational, Planning and Resource allocation tools 58
4.1 OCATA Digital Twin for Multiband 58
4.1.1 Numerical Methods for C+L+S Optical Signal Propagation 58
4.1.2 Machine Learning models 62
4.1.3 Results 66
4.1.4 Concluding Remarks 74
4.2 NetDevOps for Network Control and Automation 75
4.3 Distributed Intelligence and MAS Control 79
4.3.1 Autonomous Flow Routing Operation 79
4.3.2 Models and Algorithms 83
4.3.3 Description of Workflows 87
4.3.4 Experimental Results 91
4.3.5 Concluding Remarks 97

4.4 Optimizing Energy-Efficiency with AI-Assisted SDN Controller in Integrated Packet Optical Transport Network 97
4.4.1 Description 97
4.4.2 Complete Performance Evaluation 99
4.5 DRL-based Routing and Spectrum Assignment for Enhanced Network Performance. 104
4.5.1 Latency-aware RSA based on DRL 104
4.5.2 QoT-Aware DRL-based RMBSA in MultiBand Elastic Optical Networks 108
5 Telemetry System 113
5.1 Telemetry System based on MQTT 113
5.1.1 Telemetry System for Network Topology Management 113
5.1.2 System Architecture 113
5.1.3 Concluding Remarks 115
5.2 Leveraging LLMs for Telemetry Data Analysis 115
5.2.1 Integrated LLM, RAG, and Process Mining Framework 116
5.3 RIC Telemetry System 118
5.3.1 O-DU Telemetry 118
5.3.2 O-CU Telemetry 122
6 KPI Tracking 125
7 Conclusions 128
8 GLOSSARY 129
9 REFERENCES 132

1 INTRODUCTION

Building on the design and implementation advancements from D4.1 and D4.2, D4.3 delivers the final version of the control plane infrastructure. This comprehensive solution spans the RAN, PON, IPoWDM, MBoSDM, and optical transport domain with hierarchical control of the network. It supports: (i) advanced monitoring and streaming telemetry, (ii) infrastructure and domain-level control, (iii) multi-domain orchestration, and (iv) AI-assisted planning, resource allocation, and network operations.

The establishment of hierarchical network control requires standardized Application Programming Interfaces (APIs) and protocol specifications between the control plane and the data plane. In WP4, SDN agents are developed to improve data plane visibility and support the configuration of IPoWDM, MBoSDM nodes, and optical pluggables. This helps streamline network control and management across domains using standard data models. The SDN agents also enable telemetry data streaming through standard protocol definitions and YANG models.

The network domain controllers are implemented to control different network segments like RAN, PON, IPoWDM, MBoSDM, and optical transport network. This includes integration of agents to the respective domain controllers for network operations. A spatial PON controller is developed to integrate the PON network and controlled from control plane fitting into the SDN hierarchy. The control of the MBoSDM network, its architecture, key challenges, protocol extensions, and path computation algorithms is demonstrated through a proof-of-concept using the MBoSDM controller and a multiband over SDM optical node prototype. Similarly, dedicated domain controllers are used to control and monitor the respective network domain. For novel control of the network, NetDevOps paradigm is used for infrastructure configuration and control to allow incremental network change management. To perform unified control and management of the network, orchestration components including service and network orchestrator is integrated to facilitate multi-domain network orchestration. The service and network orchestrator are integrated where the end-to-end service is managed using service orchestrator and the network orchestrator takes care of changing of service across network domains with appropriate configurations. The integrated final version of components of SEASON control plane solution and its interworking will be demonstrated as two demos 1 and 2 which will be discussed in WP5.

Operations, planning, and resource allocation tools developed in WP4 enhanced service assurance by utilizing AI-assisted fault management solutions and AI models to optimize resource efficiency, focusing on energy-aware and latency-aware strategies. A digital twin solution for multiband networks has been developed using machine learning, enabling multiband (MB) optical system simulation and Quality of Transmission (QoT) estimation.

Furthermore, a distributed intelligence approach, using a multi-agent system, has been proposed to support SDN controllers in managing network services in near real-time.

The experimental results demonstrates the ability to proactively anticipate performance degradation and dynamically reconfigure resources, thereby optimizing both service quality and network cost. To enable a comprehensive monitoring framework, telemetry solutions are proposed across network domains using appropriate telemetry agents. By integrating monitoring information provided by T3.4 in WP3, and effectively combining the telemetry data from the optical data plane, the machine learning solution leverage and exploit various use cases in advanced network management. Additionally, the telemetry system for the RAN domain has been developed to support use cases such as energy savings.

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

The SEASON project aims to design and validate a transport network infrastructure that can support beyond-5G technologies and new emerging services. The goal is to ensure capacity growth for both mid-term and long-term needs across access, aggregation, and metro/long-haul segments. In WP4, we propose a comprehensive network architecture that includes PON, SDM-PON, a converged packet-optical transport solution, MultiBand over SDM (MBoSDM), and optical transport solutions. The control plane components and applications are designed to enable:

(a) Automatic network configuration,

(b) Self-healing in case of failures,

(c) Secure access and device control, and

(d) Optimal use of network resources, making the network fully self-managed.

Figure 2.1 shows the SEASON reference transport network architecture, extending from the RAN to the core, along with the related control and orchestration technologies. The architecture is designed to support beyond-5G services through a unified, intelligent, and programmable infrastructure. It includes key segments—RAN, PON, Aggregation, Metro, and Core—connected by advanced transport technologies such as multiband over SDM transmission and switching hardware.

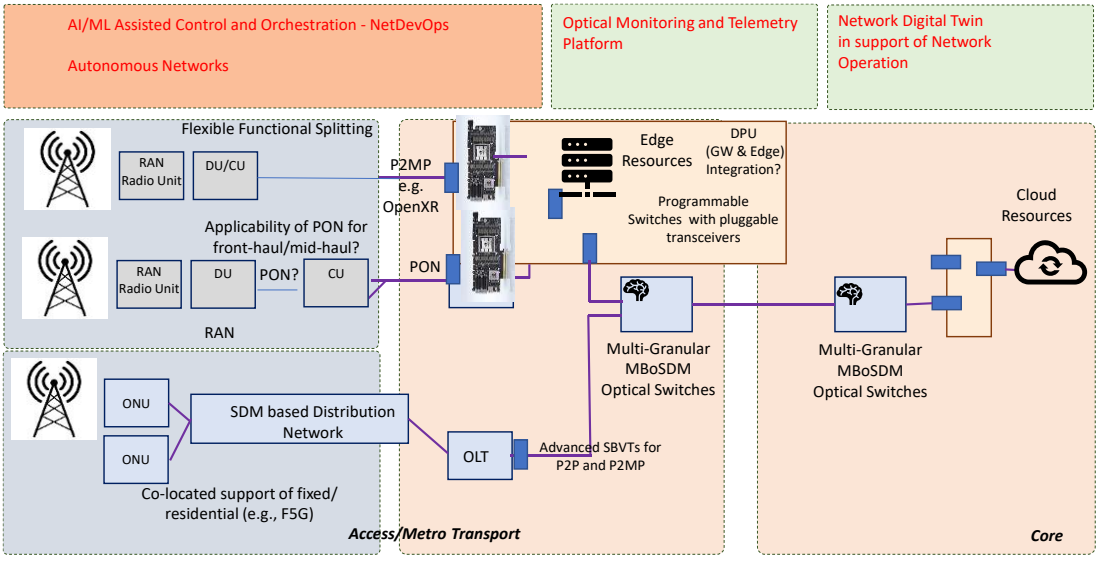


Figure 2.1: SEASON Architecture Self-managed and autonomous networking.

Dissemination Level	PUB (Public)

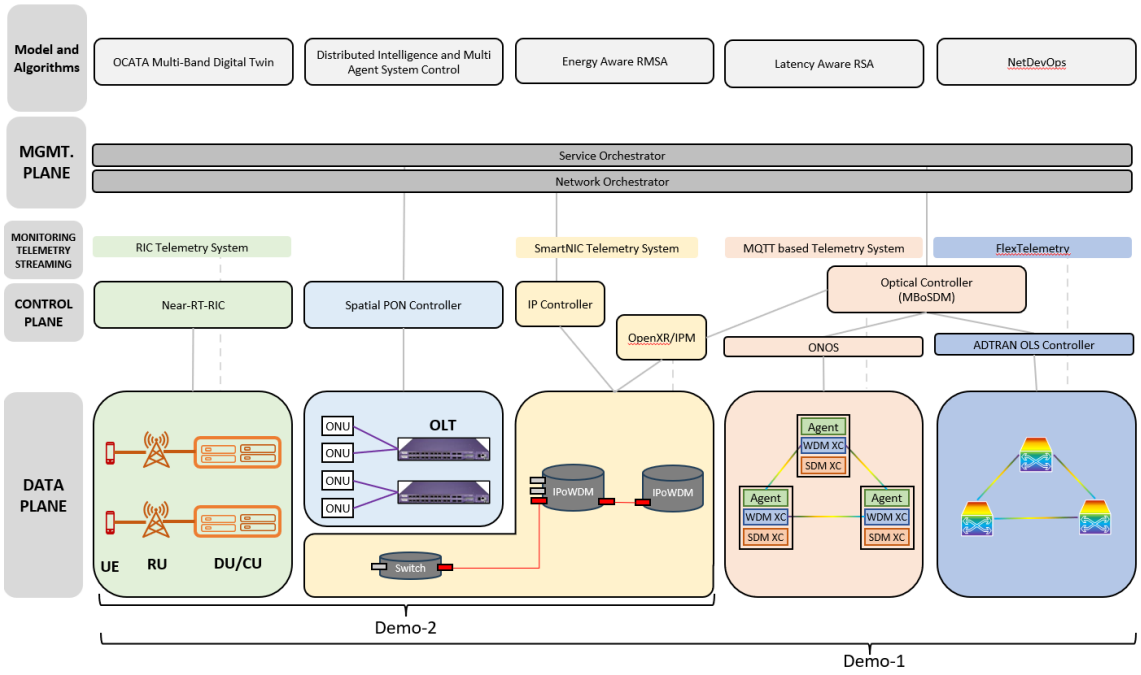


Figure 2.2: Final SEASON control plane architecture with components developed in WP4.

Figure 2.2 presents the final SEASON control plane architecture with components developed in WP4, which integrates RAN, spatial PON, packet-optical, MBoSDM, and optical transport networks. Each data plane network is managed by its corresponding domain controller in the control plane, as shown in the Figure 2.2. The interface definitions and control procedures for each domain are detailed in their respective sections: RAN (Section 3.2.6), PON (Section 3.2.4), IPoWDM (Section 3.2.3), MBoSDM (Section 3.2.1), and optical transport (Section 3.2.2). These include both vendor-neutral and native APIs, as described in the relevant sections. To enable end-to-end service control and management, the domain controllers for different network segments are integrated into a network orchestrator. The spatial PON controller, IP controller, and optical controller connect to the orchestrator in the management plane to handle the full service lifecycle.

Infrastructure monitoring and telemetry streaming solutions have been developed for all network domains. Telemetry data retrieval, using APIs and protocols, has been validated and is discussed in detail in Section 5 . The telemetry solution has been evaluated for size, overhead, latency, visibility, and scalability. To improve operational intelligence, the architecture adopts NetDevOps practices and AI/ML-driven control loops, including Multi-Agent Systems (MAS) for near-real-time decision-making. These agents operate close to the data plane to optimize resource usage, reduce energy consumption, and maintain service performance. Digital twins are also employed for advanced use cases such as fault localization, enabling predictive maintenance and proactive network management.

3 CONTROL PLANE ELEMENTS

3.1 SDN AGENTS AND PACKET/OPTICAL CONTROL

3.1.1 SDN Agent and Control of Data Processing Unit (DPU)

Recent progress in transmission technologies and network programmability is enabling the seamless integration of optical, packet, and computing resources across the edge-to-edge and edge-to-cloud continuum. At the transmission layer, compact coherent pluggable transceivers have driven the adoption of IPoWDM switches, which combine optical and packet resources within a single network element, delivering significant benefits in terms of capital costs, latency, and energy efficiency. Moreover, coherent pluggables can potentially be inserted directly into the latest generation of high-speed Smart Network Interface Cards (SmartNICs), also known as Data Processing Units (DPUs), further minimizing opto-electro-optical conversions. Both switches and DPUs can also take advantage of network programmability and hardware acceleration for networking functions, for example through P4 or DOCA technologies.

In SEASON, reported in D4.1, we designed and developed an SDN Agent enabling the configuration of a coherent pluggable module directly inserted within the SmartNIC/DPU, according to the OpenConfig model.

In SEASON, reported in D4.2, a framework for pervasive telemetry and accelerated networking that integrates optical, packet, and computing resources through DPU programmability is presented, enabling an edge-to-edge continuum. The scenario combines edge computing nodes with DPUs and IPoWDM switches with programmable ASICs, both leveraging coherent pluggable modules. Although it does not yet overcome current hardware limitations (e.g., DPU compatibility with coherent transceivers), this integrated setup supports a scalable telemetry system where switches and DPUs offload packet processing, allowing rapid reaction to failures or SLA degradations without relying on centralized SDN controllers.

The framework relies on decentralized Multi-Layer Network Telemetry (MLNT), which enables in-network generation and wire-speed processing of performance data across optical and packet layers. This approach exchanges real-time QoT and QoS information with minimal bandwidth overhead, while SDN controllers are only needed to pre-compute backup paths.

This is further enhanced by integrating DPUs at source and destination nodes, equipped with DOCA Flow APIs and Deep Packet Inspection (DPI) accelerators for efficient telemetry header injection, inspection, and packet cleaning at line rate. Experiments demonstrate that 100G DPUs can sustain hardware-accelerated DPI above 30 Gb/s, largely exceeding the requirements of pervasive telemetry in converged packet-optical-compute infrastructures. This work has been presented at the ECOC Conf. [Cug24].

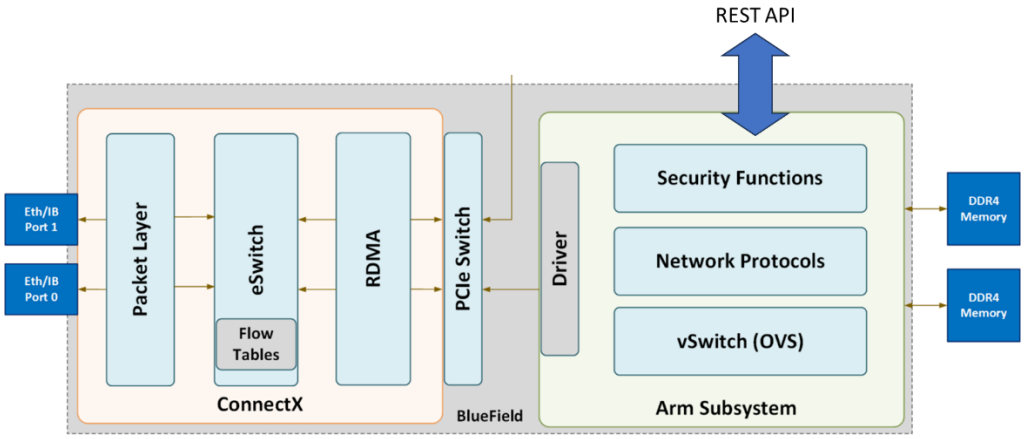


Figure 3.1: DPU architecture enhanced with REST API, part of the SDN agent, enabling automated deployment of networks functions within the DPU.

In this deliverable, we report on a novel REST API as shown in Figure 3.1, part of the SDN agent, enabling automated deployment of networks functions within the DPU.

This REST API enables lifecycle management of DPU applications, including operations such as starting, stopping, and inspecting containers. It also allows operators to validate the behavior of DPU network functions such as cyber security and forwarding.

The API begins with a discovery phase, enabling the operator to identify currently running network functions by querying the platform. At initialization, no functions are active.

This is followed by a provisioning phase, in which the operator launches a specific network function deployed as a container. In our proof-of-concept implementation, we used a DPI service combined with forwarding rules. Once running, this service inspects selected traffic flows and applies the corresponding decisions. For example, as proof of concept, forwarding packets from DPU port 0 to DPU port 1.

When the service is no longer required, the container can be cleanly stopped and removed by specifying its name. The API operates with millisecond-level responsiveness. We investigated the startup performance of a containerized L2 hairpin forwarder (vnf_fwd). Using both direct Docker CLI commands and a REST API control path, we measured the time from the command received to the point where the forwarding process became active. Results show that the application initializes in approximately 0.27–0.32 seconds. The REST API path is shown with a median difference of only 3 ms, confirming that the control interface introduces no significant overhead.

3.1.2 SDN Agent for IPoWDM nodes in ROADM-free networks

Figure 3.2 shows the overall architecture of the agents designed for the control of the IPoWDM boxes.

The SDN Agent has been designed to support the new functionalities of disaggregated IPoWDM boxes. In particular, the containerized NETCONF agent, supporting optical transmission setting, has been enhanced with packet parameters (i.e., IP layer).

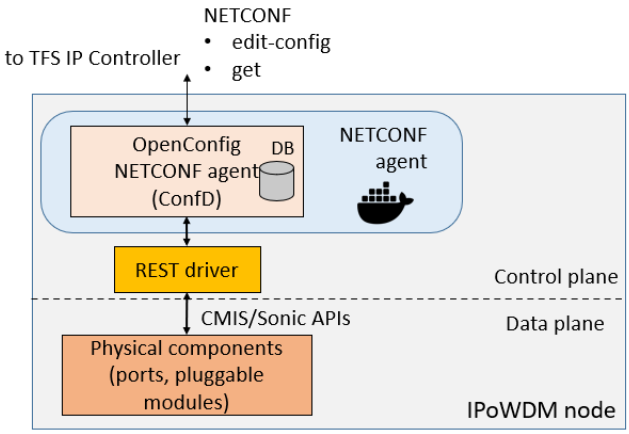


Figure 3.2: IPoWDM software components.

More specifically, the `openconfig-if-ip` has been considered for the configuration of IP address, VLAN and logical interfaces. By exploiting this model, the IP SDN Controller is able to enforce to the IPoWDM boxes the configuration of the IP address to be assigned to the Ethernet interfaces where the coherent pluggable modules are attached.

The `openconfig-bgp` and `openconfig-rib-bgp` models have been considered to handle the BGP protocol related configurations in the FRR container running in the Sonic OS. This includes the capability to activate new BGP instances, with specific BGP neighbors.

A Python REST-based driver has been designed in order to expose CMIS/Sonic configurability to the NETCONF agent, acting as REST client. Though the driver, it is now possible to perform the raw configuration of the IPoWDM components including both the packet and the optical layers, supporting the required configuration methods (i.e., configuring optical parameters of the pluggable modules, configuring IP addresses, configuring BGP neighboring sessions.

The experimental assessment is detailed in Section 3.2.5 where the Agent is implemented in a network testbed together with the enhanced TFS SDN Controller.

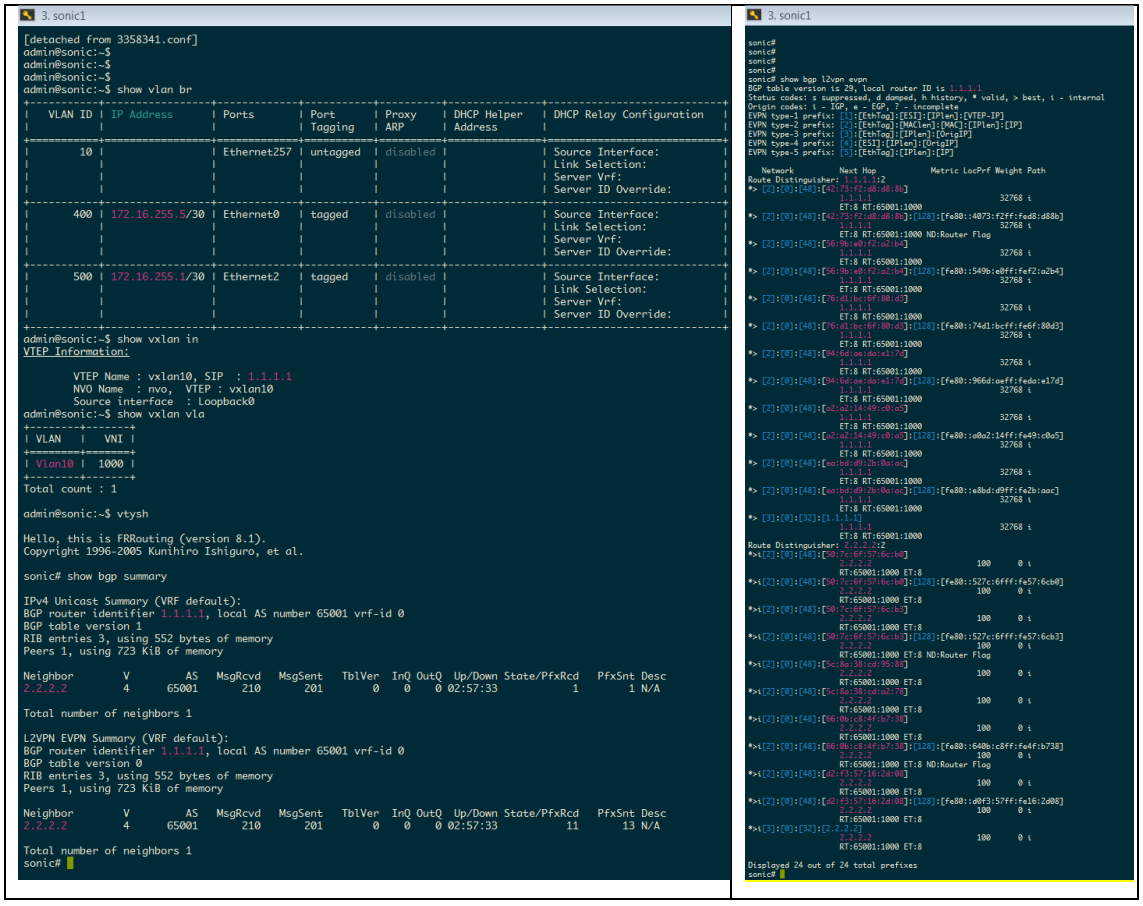


Figure 3.3: Screenshot at the SONiC white box running the SDN Agent.

Figure 3.3 shows the status of the IPoWDM box upon a successful configuration of a L2VPN (active VLANs, VXLAN, BGP, L2VPN).

This work has been published in [Sga25].

3.1.3 SDN Agent for MB Transceiver

A MB(oSDM) sliceable bandwidth/bit rate variable transceiver (S-BVT) developed in SEASON operates across the S, C, and L bands enhancing capacity scalability through its modular architecture. It is based on orthogonal frequency division multiplexing (OFDM), intensity modulation (IM), and direct detection (DD), with adaptive digital signal processing (DSP) to ensure higher flexibility and performance. The transceiver architecture enables point-to-point (P2P) and point-to-multipoint (P2MP) connectivity, while providing flexible and efficient bandwidth allocation across spectral and spatial channels, meeting the demands of future optical networks. (see Section 7.4 in WP3 – D3.2)

To enable remote programmability, the control and management functions of a device expose an API towards external applications or clients. A domain SDN controller behaves as a client. NETCONF/YANG are an industry-wide adopted transport protocol and data modeling language. A common approach to implement an SDN agent is to decompose it into two main functional blocks: a front-end, able to support NETCONF/yang, and a back-end, which is responsible for the actual device configuration using internal interfaces.

Open Optical Terminals in general cover transponders, switchponders, muxponders, etc. with the ability to switch and multiplex multiple client signals into optical signals. The agent deals with uniform components hierarchy, multiplexing stages and cross-connection logic discovery and optical channel configuration comprising Frequency, power, and operational mode. We followed the OpenConfig framework for the S-BVT, reusing the existing approach to model an optical platform and terminal device. Our current work focuses on two different operations within the OpenConfig data model: (a) optical channel configuration, and (b) logical channel assignment. The first operation addresses the configuration of an optical channel with a particular central frequency, power, and operational mode. To do this, we extended the underlying YANG model to support the dynamic configuration of the optical channels (i.e., central frequency) of the different slices, in other words, multiple optical channels in a single line port. Let us note that the operational mode field was, originally, vendor-specific and encodes the different transmission modes supported by the device/transceiver into a single integer value. The corresponding value can specify information about the transmission/signals such as modulation format, or FEC.

3.1.4 SDN Agent for the MBoSDM node prototype

A MBoSDM node prototype is currently being developed at CTTC premises as shown in Figure 3.4, to demonstrate transparent switching at the WDM and SDM level. As part of the ongoing work, the node must be controlled by the SDN control plane within the CTTC ADRENALINE testbed. The multiband over spatial division multiplexing (MBoSDM) node prototype developed in SEASON supports spectral (band and wavelength) and spatial granularities, using fixed- and flex-grid WDM technologies, C-band AWG and WSS devices, and 1x3 MB filters for S-, C-, and L-band bidirectional transmission. SDM is achieved with a 19-core multi-core fiber (MCF) and fan-in/-out devices, enabling core switching and add/drop operations via an 18x18 optical matrix. The node prototype is integrated into the ADRENALINE testbed, which operates within the C-band, and enables optimized resource utilization. Its scalable design, leveraging broader spectrum coverage and spatial channels, ensures high-capacity, efficient, and dynamic switching for future network demand (see Section 7.1 in WP3 – D3.2)

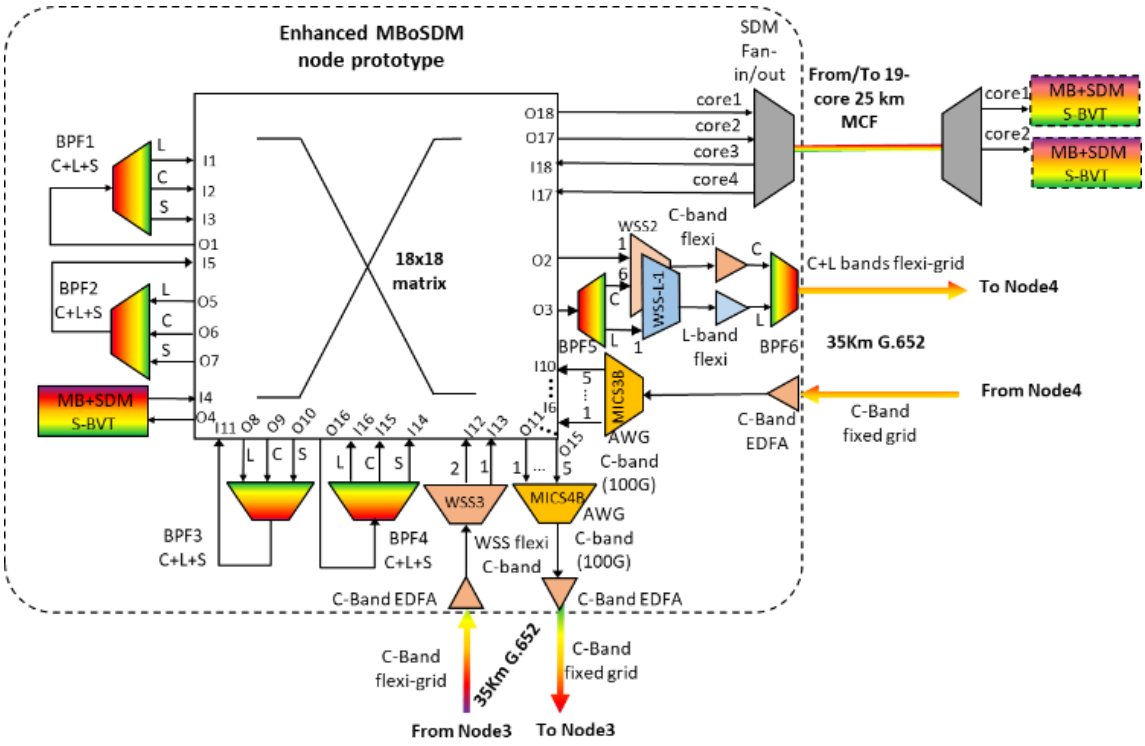


Figure 3.4: IPoWDM software components.

An SDN control plane agent has been developed using the Netopeer framework. We have defined a Yang model that maps the previously mentioned high level functions into the creation of “connections” within the node. The SDN controller uses NETCONF to create or release connections and provide the required parameters. We can see a simplified Yang model for the SDN agent in which the main data node is the list of connections that can be modified by the SDN controller. The creation (or deletion) of a connection in the data plane triggers the process of hardware configuration using the low-level interfaces.

3.1.4.1 Control Architecture

The proposed control architecture is in line with SEASON control plane as shown in D4.1 and D4.2. this includes

- An SDN Controller, in this case the FlexOpt SDN controller, properly augmented to perform DWDM, band or SDM (core) switching. This component is provided from WP4 and extended according to the hardware device model
- An SDN Agent based on NETCONF implemented within the project. The SDN agent maps the high-level operations to the configuration of matrices and WSS and related subsystems. The controller facing part is based on Net2peer opensource framework and is written in python.

Dissemination Level	PUB (Public)

- Yang model for the device. A specific yang model is being developed for the identified high-level operations in the previous section. The yang model is the reference and contract between the SDN controller and the agent and supports the identified Create/Read/Update/Delete (CRUD) operations on Cross-connections, with constraints in terms of input ports / output ports and type / layer of cross-connection

3.1.4.2 Yang model

Based on the previous considerations, an initial Yang model has been developed for the SDN agent that captures the basic information to perform operations on the device. The following snippet shows the Yang model in tree format.

```
module: season-mbosdm-node
  +--rw node
  |  +--rw address        inet:ip-address
  |  +--rw node_id        yang:uuid
  |  +--ro ports
  |  |  +--ro port* [id]
  |  |     +--ro id           uint32
  |  |     +--ro name?        string
  |  |     +--ro direction?   season-mbosdm-types:port-direction
  |  |     +--ro type?        season-mbosdm-types:port-type
  |  +--rw connections
  |     +--rw connection* [name]
  |        +--rw name           string
  |        +--rw input_port     uint32
  |        +--rw output_port    uint32
  |        +--rw input_core     uint32
  |        +--rw output_core    uint32
  |        +--rw band?          enumeration
  |        +--rw n              int16
  |        +--rw m              uint16
  +--ro info
     +--ro software_version?   string
     +--ro hardware_version?   string
```

In summary, progress on the design and development of the agent is as expected. We completed the first integrations in Q1 2025, and final integration demonstration will take place in Q3 2025.

3.1.5 FlexTelemetry Agent

This work presents the outcomes of the SEASON project with a focus on the development and deployment of a Python-based streaming telemetry application as seen in Figure 3.5 for real-time network monitoring. The application is designed with protocol flexibility, supporting industry-standard southbound interfaces such as NETCONF, gNMI, and SNMP, making it adaptable to a wide range of optical and packet network devices.

In addition to its flexible input protocol options, the application offers a plugin-based architecture for northbound integration, enabling seamless data export to message brokers (e.g., Kafka, MQTT, Redis) and time-series databases (e.g., InfluxDB). This design allows operators to easily integrate telemetry data into existing analytics and visualization pipelines, such as Grafana, for both real-time monitoring and long-term trend analysis.

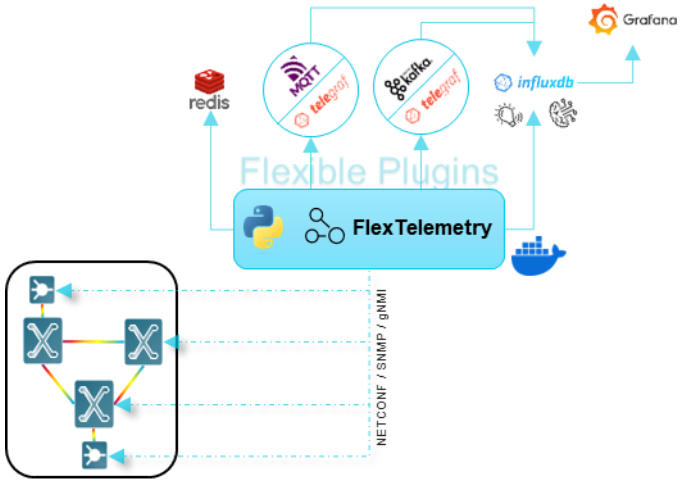


Figure 3.5: FlexTelemetry Architecture: Northbound integration with flexible output message brokers, and southbound integration with adaptable protocols.

As part of the SEASON project, the telemetry application was successfully deployed and tested at the Fraunhofer HHI lab. The deployment focused on collecting telemetry data from optical transport components, including transponders, ROADM nodes, and optical amplifiers. The platform demonstrated its capability to stream telemetry in parallel from multiple devices, leveraging Python's multiprocessing to maintain low-latency and high-throughput performance across concurrent data streams. The Figure 3.6 illustrates the telemetry payloads being dispatched to the selected northbound plugins.

```
Intitiating Streaming
2025/06/23 16:26:36 Initiating data streaming driver netconf
2025/06/23 16:26:36 Initiating data streaming driver netconf
Initiating TeraFlex
Initiating QuadFlex
Streaming from IP: 172.20.132.230
Streaming from IP: 172.20.132.231
✅ Published to the InfluxDB bucket
✅ Published to the InfluxDB bucket
✅ Published to the Kafka topic : QUADFLEX_172.20.132.230
✅ Published struct to MQTT: QUADFLEX_172.20.132.230
✅ Published to Redis channel: QUADFLEX_172.20.132.230
✅ Published to the Kafka topic : TERAFLEX_172.20.132.231
✅ Published struct to MQTT: TERAFLEX_172.20.132.231
✅ Published to Redis channel: TERAFLEX_172.20.132.231
```

Figure 3.6: Application output showing data delivery to various message brokers and time series databases.

3.1.6 OpenROADM Agent

In the framework of the SEASON project and in line with its objectives, the OpenROADM agent developed for the control of advanced optical pluggable module (ZR/ZR+), already described in [D41], has been enhanced for managing also spatial optical matrices. To this purpose, the OpenROADM <roadm-connections> concept has been extended to create connections between OTS interfaces. Basically, the optical matrix is modeled as a peculiar ROADM device able to switch the whole Optical Transport Section. Consequently, the <source> and <destination> containers of the <roadm-connection> one, besides Network Media Channel interfaces as for "classical" ROADMs, can also reference OTS interfaces, as shown in the following snippet.

```
<org-openroadm-device xmlns="http://org/openroadm/device"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <roadm-connections nc:operation="merge">
     <connection-name>OTS-CTP-DEG1-TTP-to-OTS-CTP-DEG1-
TTP</connection-name>
     <opticalControlMode>off</opticalControlMode>
     <target-output-power>0</target-output-power>
     <source>
        <src-if>OTS-DEG1-TTP</src-if>
     </source>
     <destination>
        <dst-if>OTS-DEG1-TTP</dst-if>
     </destination>
   </roadm-connections>
 </org-openroadm-device>
```

Moreover, to interact with real hardware, exploiting agent flexibility, it has also been developed a driver for a GlimmerGlass G300 48x48 optical switch.

The agent has been integrated and tested with the CTTC FlexOpt optical controller and will be employed in the FiberCop demo foreseen for early autumn 2025.

3.2 SDN CONTROLLERS

3.2.1 MBoSDM Optical Controller

CTTC FlexOpt SDN framework has been used as an SDN controller for MBoSDM networks. During the first reporting period, the main aspects were defined, including its usage in single domain emulated scenarios. The second period has focused on polishing the implementation, better support north bound interfaces and TAPI extensions for SDM and to integrate control of the actual node agent prototype, as described earlier in the document.

We present the overall architecture and main challenges, the proposed protocol extensions and the implemented path computation and resource allocation algorithm. We illustrate the solution with a proof-of-concept using CTTC FlexOpt SDN controller, which includes an ongoing multiband over SDM (MBoSDM) optical node prototype.

3.2.1.1 Background

When using SDM, in single level optical networks, parallel links are deployed either as bundles of fibers (BoF) or as multi-core fibers with fan-in / fan-out systems. To date, such SDM systems are mostly point-to-point and switching only happens at the media channel (DWDM) level and SDM or spatial layer is systematically terminated at each node and transported OTSi signals are switched (their media channels, referred to as DWDM switching). Although systematically terminating spatial layer connections and switching at the DWDM layer has potentially the best efficiency in terms of blocking probability and network capacity -- given the fine granularity of switching at the upper layers -- in practice, this approach has multiple drawbacks when scaling to required capacity involving many fibers and/or spatial cores. For example, the required Wavelength Selective Switches (WSS) may become a bottleneck or simply unfeasible due to the number of required ports to ensure contention-less, color-less and contention-less nodes or not being able to cover the whole optical spectrum with the required granularity, which requires more complex node architectures introducing e.g. per-band filtering and dedicated planes.

On the other hand, multi-level networks extend single level networks by introducing different switching granularities (levels), and these may encompass, for example, WDM switching (media channel switching, either in fixed- flexi- grid) and Fiber / Core switching in single nodes. Efficient path computation algorithms and control plane architectures are required to operate such networks.

3.2.1.2 Overall architecture

The control plane architecture relies on a single SDN controller that is responsible for the Optical Line System (OLS). This OLS controller exports a North Bound Interface that extends the transport application programming interface (TAPI) to support the provision of media channels or SDM connections as seen in Figure 3.7. Main entities such as Connectivity Service End-Points (CSEPs), Connection End Points (CEPs) and Node Edge Points (NEPs) have been augmented to convey information about the status of e.g. fibers or cores within a single traffic engineering link.

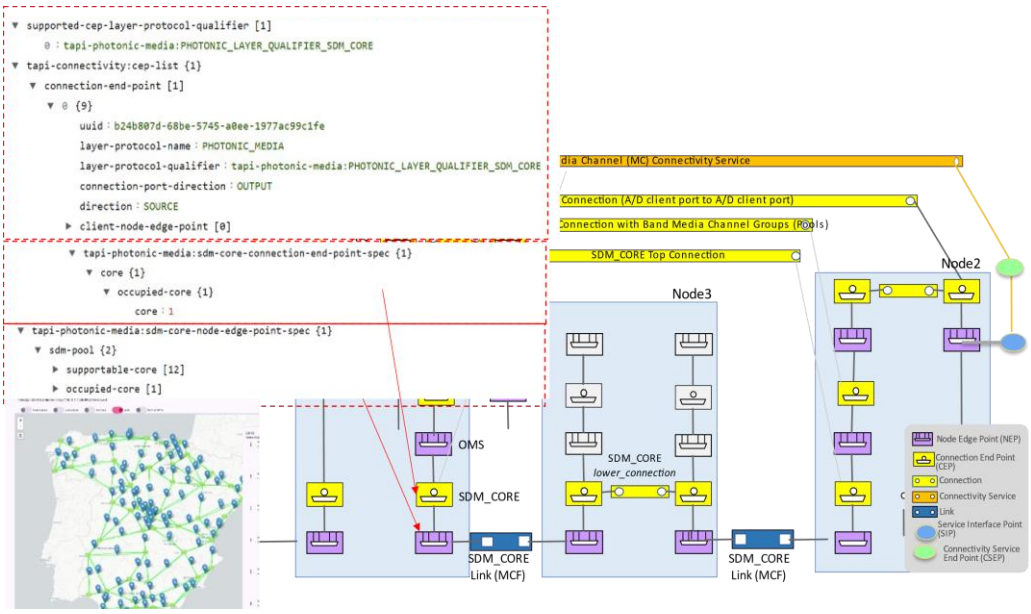


Figure 3.7: Graphical Representation showing the TAPI context and the modelling of the network showing the extensions to SDM related to core allocation, es exposed by the MBoSDM OLS SDN Controller.

3.2.1.3 Path Computation Algorithm

The path computation function is a key function of the OLS controller. The objective is to find a path between two endpoints (Service Interface Points) and allocate a continuous media channel (frequency slot) in a multiband context. For this, the function may decide to allocate lower layer connections (SDM connections) that become higher layer links. The complexity of such algorithms is high, and in this work, we have devised an algorithm that works as follows: upon the arrival of a connection, try to find a path in the DWDM layer, ensuring sufficient QoT based on predefined parameters such as reach. A given DWDM link has a pool of the C, L and S bands. If this path computation fails, the algorithm shall look for additional SDM connections – with or without core/fiber continuity constraints, based on hardware limitations and capabilities) and given the additional DWDM links they induce, try again the DWDM allocation. If the algorithm cannot find a WDWM frequency slot and cannot create a SDM connection directly between the end nodes, the algorithm fails.

3.2.1.4 Control Plane Workflow

The operation of the SDN control plane starts with the onboarding of the controlled topology. In this work, we have considered different topologies with real or emulated hardware. For example, one of the topologies is based on the Telefonica topology with 65 nodes, and 270 links with 5 cores per MCF. The Operator Support System or Client may retrieve the TAPI context and

available resources, including the different nodes, links and their switching capabilities as well as available resources.

At some point, the user proceeds with media channel service provisioning S1 between nodes. Upon request, the path computation algorithm finds a path which may or may not involve the provisioning of SDM connections (with continuous core between endpoints in the case of core continuity constraint) and the subsequent instantiation of virtual link(s). Such DWDM links are managed in virtual network topology (VNT) and keep track of both the underlying supporting connections as well as the client services. Note that links induced by SDM connections can be reused by different DWDM services (grooming). When the service connections are released if a SDM connection is no longer used by a WDM connection, the supported DWDM link is removed and the SDM connection is released.

As part of the provisioning process, the OLS controller interacts with the SDN agents within the network elements to configure the optical nodes. Based on a high level abstraction there are different operations that can be done: ADD or DROP operations of a media channel between an add/Drop port and a degree port, cross-connect a given media channel given its frequency slot and input/output ports and, for the SDM layer ADD or DROP a client port of the SDM layer into the SDM fiber or cross-connect a core between an input and output core. The next section details the SDN agent developed in our prototype.

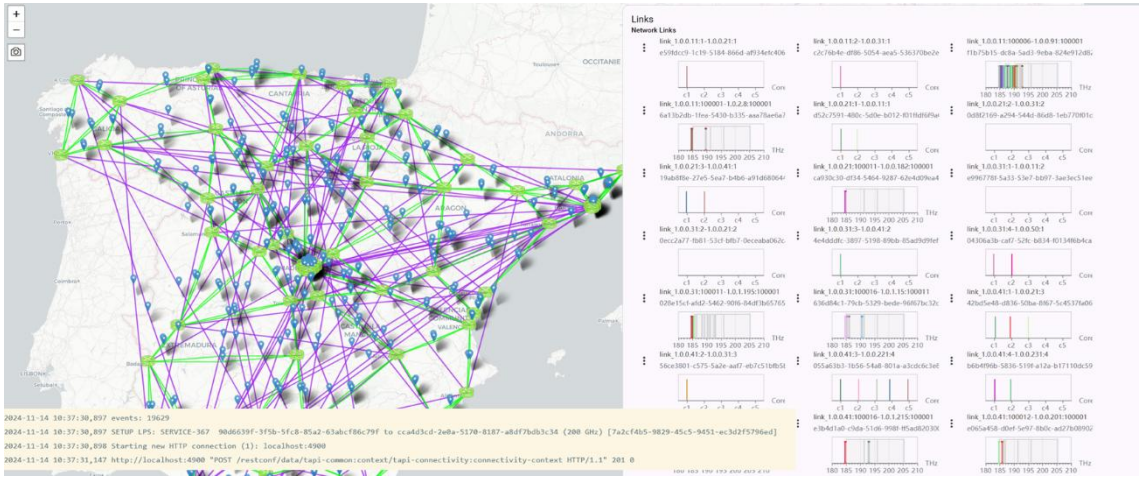


Figure 3.8: Network state after batch of service requests: 336 connectivity services:110 SDM core - supporting 110 virtual DWDM links (purple) and 226 flexi-grid services. Links show core allocation (SDM) and freq. slots (DWDM).

Let us note that when the data plane is emulated, the controller can be used to evaluate the performance of the system and the algorithm, by either using Markovian processes or evaluating the accepted traffic load (network capacity) to reach a blocking rate of e.g.1%. The Figure 3.8 shows the network state retrieved via the RESTCONF/ TAPI interfaces after batch of service requests: 336 connectivity services:110 SDM core - supporting 110 virtual DWDM links (purple)

and 226 flexi-grid services. Links show core allocation (SDM) and freq. slots (DWDM). Links are either physical (SDM links) or logical WDWM links supported by a connection in the SDM layer.

3.2.1.5 Integration with the MBoSDM node prototype

For the integration of the actual node protype, an SDN agent has been implemented and a Yang model defined. The yang model supports the dynamic creation of cross-connections across different layers and this is mapped to low level interfaces (see Figure 3.9). Further details of the performance will be reported in WP5.

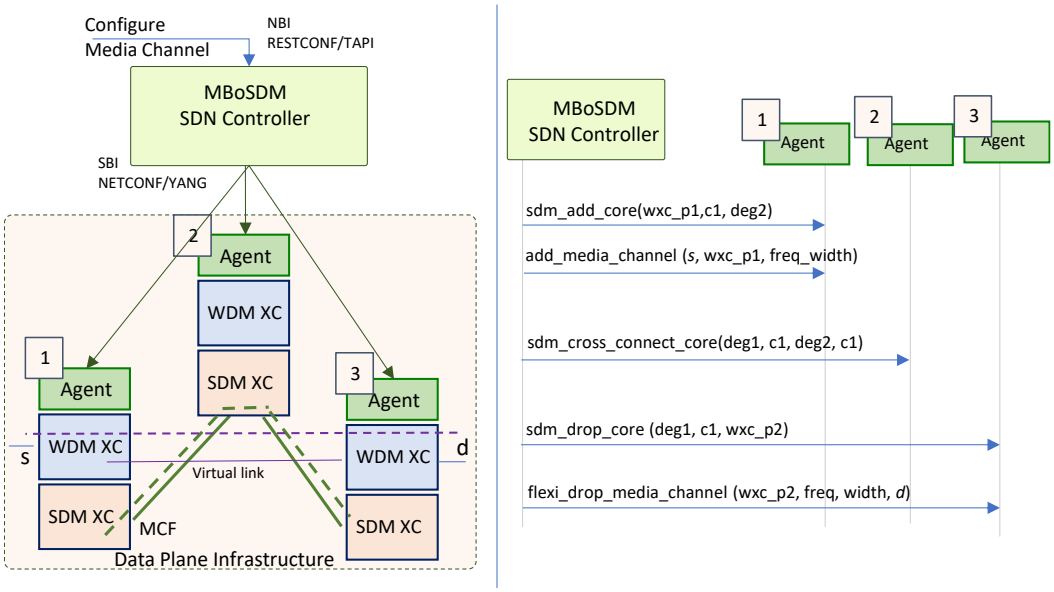


Figure 3.9: High Level messages between the SDN controller and the SDN agent, based on the pre-defined operations.

3.2.1.6 Usage as Network Orchestrator

In the context of Demo 1 with WP5, the FlexOpt SDN controller has also been used to act as a network orchestrator for the optical domain, providing overarching control for the MBoSDM and Adtran OLS domains. This will be detailed in WP5 deliverables.

3.2.1.7 Conclusion

SEASON has implemented a control plane for single domain MBoSDM networks, exporting a standard north bound interface. control plane for MBoSDM networks, showing the main challenges and design choices. We have demonstrated its applicability in MBoSDM nodes both at the control plane level with large topologies as well as the ongoing proof-of-concept being done in the ADRENALINE CTTC testbed. The summary of this work has been published in [Cas25] and will be subject of upcoming presentations at iPOP2025 in Tokyo, Japan.

3.2.2 Ensemble OLS controller

The optical controller proposed in SEASON for OLS control is based on the Adtran Ensemble Network Controller (ENC) software solution and provides a northbound ONF Transport-API (TAPI) v2.1 interface. OLS components are managed through southbound interfaces using native YANG models and protocols such as NETCONF and SNMP. The controller handles optical service provisioning, including port configuration, channel frequency assignment, and other OLS operations. On the northbound side, ENC supports TAPI v2.1 for integration. Details on OLS service provisioning, service lifecycle management, and API integration are provided in previous deliverables D4.1 and D4.2.

Based on the proposed work, two key innovations have been implemented:

- End-to-End Cross-Border Service Provisioning and Monitoring, and
- Network Automation using NetDevOps.

3.2.2.1 End-to-End Cross-Border Service Provisioning and Monitoring

We demonstrated an end-to-end cross-border service provisioning and monitoring approach using Eclipse Dataspace Components (EDC) Connectors. This solution improves scalability, performance, and regulatory compliance while removing the need for traditional manual contracts between international network operators and transit gateway providers.

As network demands grow rapidly, automating cross-border service provisioning has become essential. Our demonstration applies SDN principles to dynamically provision network routes, ensuring optimal performance. By integrating the EDC Connector, the system enables policy-driven interactions between different network operator domains, allowing automated end-to-end service provisioning in line with contractual requirements and predefined policies.

3.2.2.1.1 Software Defined Networking

The OLS controllers connect northbound via TAPI to the multi-domain controller or network orchestrator. This setup enables dynamic route management across two operator domains. In our work, we integrate the TeraFlowSDN controller with the EDC Connector so that different TeraFlowSDN instances can share network service details securely and in compliance with policies, while autonomously provisioning resources and creating end-to-end service paths.

3.2.2.1.2 Data Sovereignty and Compliance

Data sovereignty is key to our solution, ensuring stakeholders keep control of their data through strict access and usage policies. Since network data is sensitive, sharing it across operators and the transit provider requires a framework that protects ownership. We use the EDC Connector,

part of the open-source EDC framework, to manage data exchange while enforcing policies. Dataspace connectors have already proven effective in regulated sectors like healthcare. In our work, the EDC Connector automates data-sharing agreements and enables secure, policy-based interactions between TeraFlowSDN instances. This protects shared information (e.g., service descriptions) and supports performance monitoring within each domain, ensuring compliance with agreements.

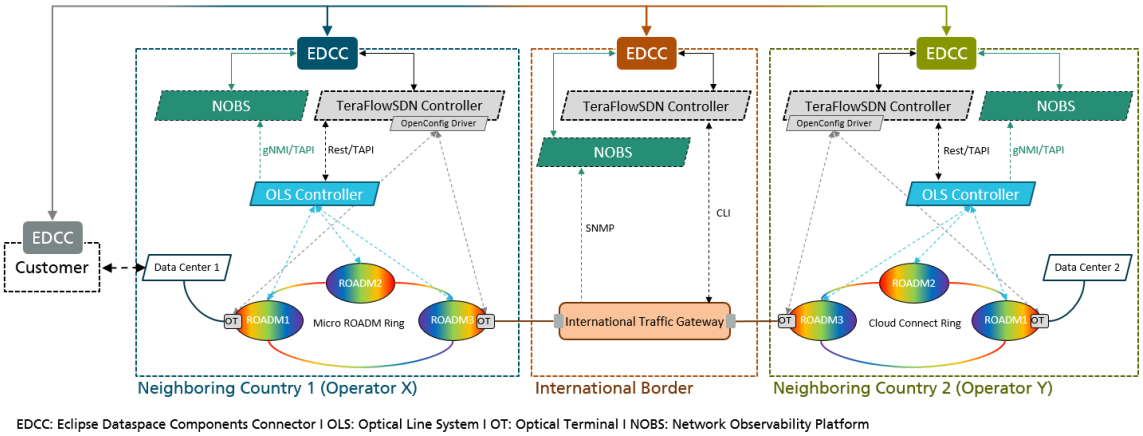


Figure 3.10: Overall Architecture: End-to-End Cross Border Service Provisioning.

3.2.2.1.3 Cross-Border Service Provisioning and Monitoring

When a customer requests an end-to-end cross-border service (as shown in Figure 3.10), the EDC Connector coordinates provisioning and monitoring between the local operator, transit provider, and target operator—removing the need for manual negotiations and paperwork.

3.2.2.1.4 Implementation and Working

The work demonstrates automated end-to-end cross-border service provisioning. When a customer requests a cross-border service (as shown in Figure 3.11), the local operator's EDC Connector sends a service request to the transit gateway, which forwards it to the destination operator. After approval, TeraFlowSDN instances in both domains share service details via the EDC Connector. The TeraFlowSDN controllers then provision resources and activate the service through OLS controllers using TAPI. Customers can monitor service performance through their operator's EDC Connector, provided policy requirements are met. More details on the components and experiment workflow are provided in Milestone 5.2 of WP5. The work has been published and demonstrated and published in OFC 2025.

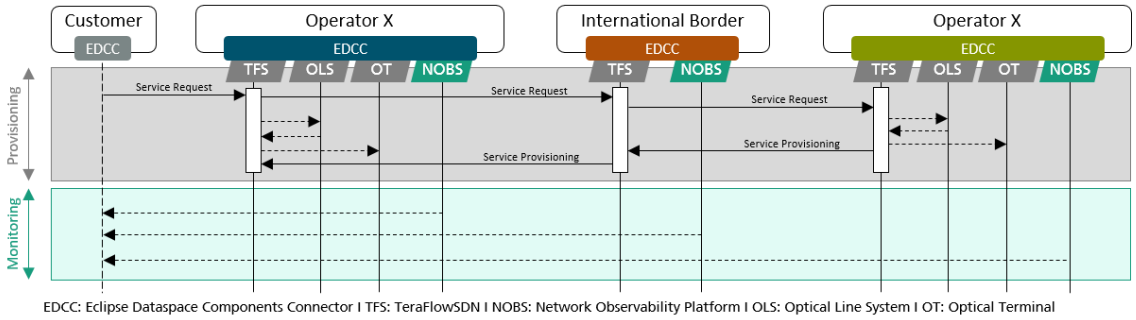


Figure 3.11: Interaction between different demo components at various stages of service provisioning and monitoring.

3.2.2.2 Network Automation using NetDevOps.

We implemented an automated NetDevOps pipeline for configuration management, version control, and deployment validation in optical transport networks. The pipeline integrates Ansible playbooks, Python based NETCONF clients, and TAPI RESTCONF APIs for device configuration and end-to-end service provisioning. With Git based version control, automated rollbacks, and dynamic resource utilization, it ensures scalability, reliability, and efficient management across diverse network devices. By leveraging digital twin environments, the proposed approach enables pre-deployment testing to ensure error-free configurations before live implementation. Performance evaluation highlights significant improvements in configuration time, resource optimization, and error reduction compared to traditional methods, addressing key challenges in managing large-scale optical network infrastructures.

Adopting the NetDevOps approach offers significant advantages over traditional methods, addressing key areas, (a) Scalability is enhanced through automated pipeline, which efficiently manage increased number of dynamic device and service configurations. (b) Reliability is improved with consistent automated pipeline jobs and network state management using version control. (c) Latency is improved effectively by automating routine tasks, leading to faster deployment cycles and quicker responses to dynamic network demands. Traditionally, DevOps practices have been primarily associated with software development, with limited adoption in network management. By extending NetDevOps methodologies, we bring automation and agility where such practices have not been extensively explored in optical network. Our approach leverages industry standard protocols, such as NETCONF and RESTCONF for configuration in a partially disaggregated network architecture. Additionally, we ensure seamless integration with optical controllers and orchestration platforms by adhering to ONF-TAPI in the northbound.

3.2.2.2.1 Implementation and Working

We implemented an automated workflow for managing network device configurations and service provisioning in optical networking, focusing on version control and Continuous Integration/Continuous Development (CI/CD) operations. The implementation leverages a NetDevOps approach to streamline complex network adjustments, improve deployment accuracy, and reduce manual errors through automation.

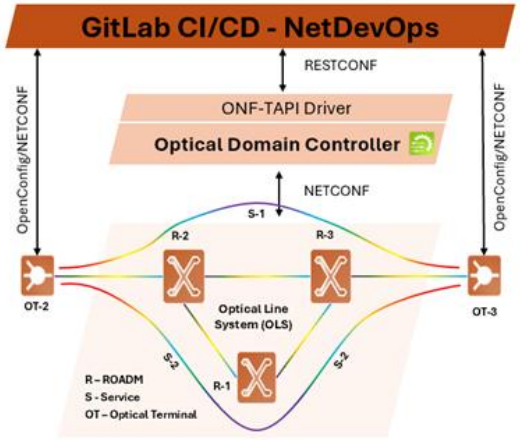


Figure 3.12: NetDevOps Demonstration Architecture in Optical Network Digital Twin.

A digital twin of the optical network is used that mirrors the actual infrastructure, enabling safe testing of configurations before deployment as shown in Figure 3.12. By replicating devices and network conditions, engineers can validate changes, identify potential issues, and ensure everything works as expected. The pipeline executes configurations in the digital twin which emulates an actual device, applying changes to virtual devices and verifying their impact. If successful, the tested configurations are then pushed to the actual network environment. This approach reduces the risk of errors, minimizes disruptions, and ensures reliable deployment, enhancing the overall efficiency of network management.

The proposed approach achieves scalability and reliability by integrating Git-based version control for automated rollbacks, significantly reducing human error and downtime. Performance evaluations demonstrate improved operational efficiency and time savings compared to traditional methods, and the results are presented in Section 4.2.

3.2.3 Optical Controller for P2P and P2MP pluggable devices (OpenXR)

One of the novel aspects that SEASON is addressing is the control of the point-to-point (P2P) and point-to-multipoint (P2MP) pluggables in support of emerging and constrained connectivity

services. A promising technology is OpenXR, and the Open XR Optics Forum (https://openxropticsforum.org/) is the multi-source agreement (MSA) working group for XR optics, the industry's first point-to-multipoint coherent pluggable transceiver technology, with solutions to scale from the network edge across metropolitan, national, long-haul networks, simplifying network operations to build smarter, more automated networks. By enabling services such as 5G and packet services, supporting new service speeds of 400G, 800G and beyond, and the scale of C+L solutions to enable network scale with more efficient network architectures.

Point-to-multipoint (P2MP) coherent optics based on digital subcarrier multiplexing (DSCM) [Welch21] has the potential to provide connectivity in access, aggregation and metropolitan networks at a low cost. P2MP technologies use DSCM to split the bandwidth into multiple Nyquist digital subcarriers (SCs) at a lower symbol rate. Each individual SC can be treated independently of all others, including modulation, management, power, aggregation, etc., and thus can be routed to different destinations, allowing a greater degree of flexibility with respect to classical fixed point-to-point (P2P) transceivers. In [Her23] the authors proposed that P2MP can be dynamically assign bandwidth providing extra resources to an end-user during the peak hours and re-assigning that extra capacity to other clients whose peak hours occur later on.

In summary, one of the activities that SEASON has developed within WP4 is the integration of the control of XR pluggables in the overall SEASON control architecture. This has been accomplished by extending the functional element of the CTTC FlexOpt optical controller to control P2MP services. The chosen model is based on delegating low level details of the pluggable configuration to the IPM (Intelligent Pluggable Manager) while the Optical controller provides a network view and a standard and open north bound interface.

3.2.3.1 Control Architecture

The P2MP programmable pluggables implement a dual management interface as described in [Hand24]. The FlexOpt Optical Network Orchestrator is able to process user requests for P2P and P2MP services. The FlexOpt exports a TAPI based north bound interface and delegates the configuration of the pluggables to a dedicated controller (the XR Intelligent Pluggable Manager or IPM) and the configuration of the optical line to a dedicated OLS controller. The main task of FlexOpt is to process the user request, to perform path computation and resource allocation (with optional path validation with external entities) and to manage the lifetime of services.

3.2.3.2 Pluggable device Discovery

The first step involves the discovery, from the Optical Controller, of the available pluggable devices, hosts and Network Demarcation Units. This involves several API calls to the underlying IPM module, once a token has been obtained during an initial authentication phase. In particular, the FlexOpt orchestrator periodically obtains the IPM OpenID token and proceeds

with the discovery of hosts (routers) and pluggable devices (modules) using a REST interface over HTTPs. Discovered network elements are mapped to TAPI network nodes and module Ethernet client ports are mapped to TAPI DSR Service-Interface-Points (SIPs) as shown in Figure 3.13. The Optical Line System is modelled as a single TAPI abstracting media channel switching and can represent a ROADM based network controlled by e.g. a TAPI-enabled OLS controller node or a transparent optical splitter. In this case, 7 modules are discovered.

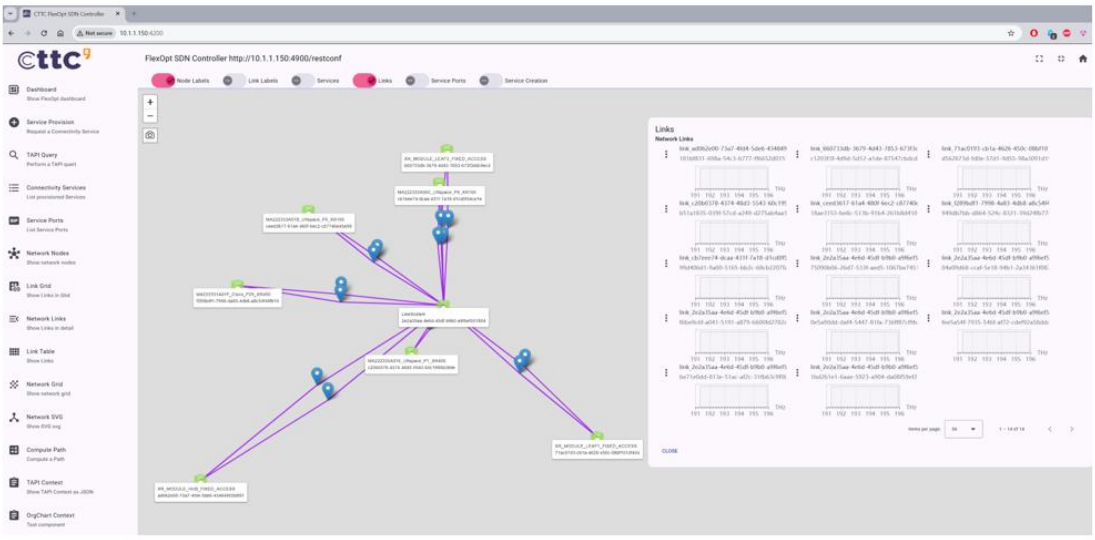


Figure 3.13: Scenario and Network Topology for the control of P2MP pluggable devices.

3.2.3.3 Service provisioning workflow

One of the key features is the proposal of a standard north bound interface (NBI) based on Transport API (TAPI) that allows the provisioning of P2MP services. For this we have used the generic “Digital Service Rate” layer protocol qualifier, in such a way that different ports of the pluggable modules have been given a Service Interface Point, so thanks to the Graphical User Interface (see Figure 3.14) the operator can select up to 4 x 100 Gb/s services from the HUB module to each of the available modules.

The FlexOpt receives the request from the northbound interface (NBI) and, as part of the process: (1) it computes the nominal central frequency for the DSCM and requests the creation of an XR network or constellation to the IPM; (2) it configures the optical line system as needed; and (3) as well as the creation of the service between the Ethernet ports in the router available breakouts. For this, the creation of the XR network involves selecting the frequency of the constellation (e.g., 191375000), the modulation (i.e., 16QAM) as well as the selection of the XR hub module (configured in L1 mode). Once the XR network is created, it proceeds to connect additional leaves to the XR network selecting the leaf modules. Finally, the “port mode” connection is requested with a third call specifying the Ethernet endpoints (client AIDs).

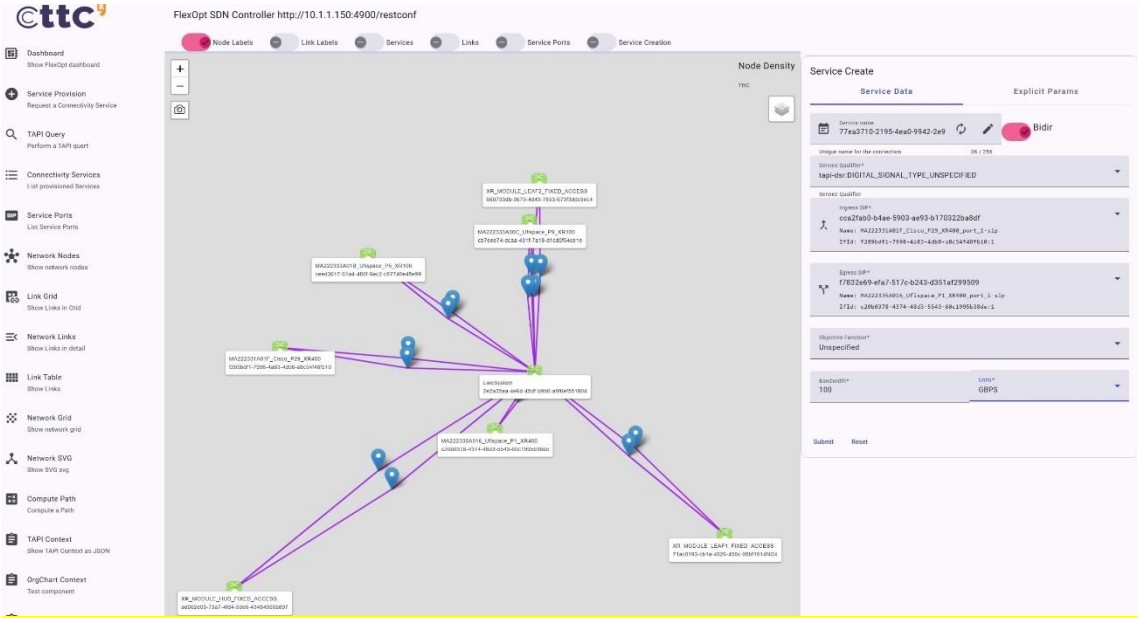


Figure 3.14: Provision of a 100G service between the XR Hub and one of the leaves, using a standard TAPI North Bound Interface.

The establishment of a 100 Gb/s service between breakout port 1 of the Cisco router and port 1 of the UfiSpace router necessitates a specific configuration process. To facilitate the 100 Gb/s service, it is imperative to configure the breakout mode on the interfaces involve in the service, specifically those utilizing a 400G pluggable. This configuration results in the creation of four subcarriers, each capable of supporting an individual link. The procedure entails:

I. Activation of breakout mode on the 400G QSFP-DD ports of both the Cisco and the UfiSpace router.

II. Allocation of one of the four available subcarriers (in this instance, subcarrier 1) for the requisite 100 Gb/s service.

III. Configuration of appropriate frequency and modulation parameters for the selected subcarrier to accommodate the 100 Gb/s data rate.

IV. Establishment of the logical connection between the designated breakout ports on each router.

V. Verification of link status and performance metrics to ensure the operational integrity of the 100 Gb/s service.

The frequency is obtained through the optical controller, which performs the corresponding frequency assignment to the involved equipment. This method enables efficient utilization of the 400G pluggable capacity, facilitating the provisioning of multiple high-speed services on a single physical port. This methodology enables efficient utilization of the 400G pluggable capacity, facilitating the provisioning of multiple high-speed services on a single physical port.

The remaining subcarriers may be allocated for additional services or reserved for future capacity expansion, thus providing flexibility in network resource allocation.

As previously mentioned, the processing of the TAPI request involves correlating with the Optical Line System status, obtaining a reference to the underlying XR network/constellation and creating the services. The first time that the network is instantiated, the Optical controller must select a central frequency and instantiate the XR network, whose payload is shown in the following snippet/capture in Figure 3.15.

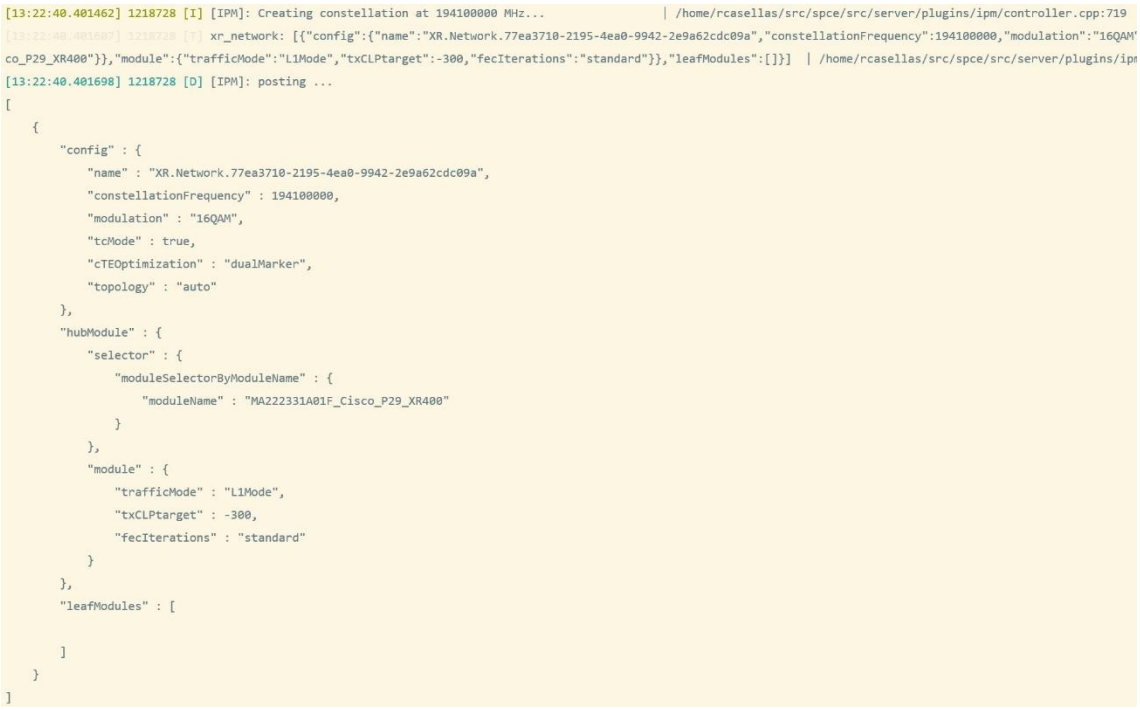


Figure 3.15: Interface Optical Controller / IPM to create the XR network, selecting the HUB module and the constellation central frequency and the selected modulation format.

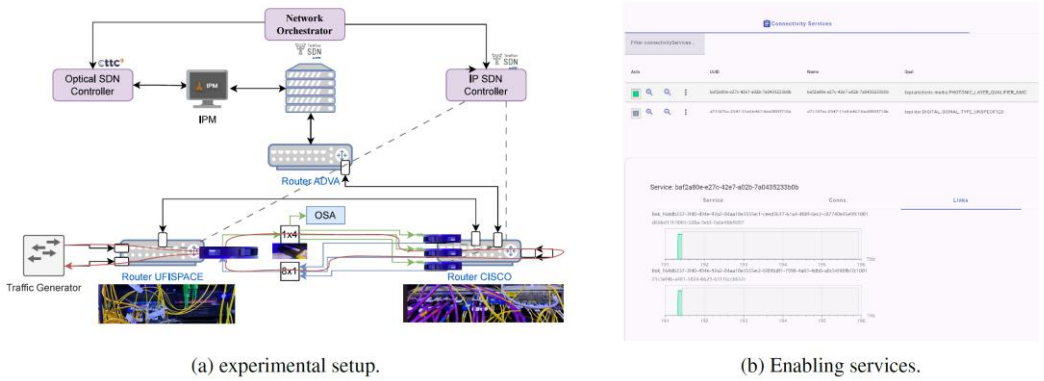
3.2.3.4 Experimental Setup

The experimental setup has been implemented in Telefonica's Future Network Laboratory (see Figure 3.16 and Figure 3.17, utilizing two IPoWDM routers: a Cisco NCS 57B1 running IOS 24.3.1 and an Ufispace router with Drivenets software. Both were equipped with 400G QSFP-DD ports and 400G and 100G P2MP programmable pluggables. The interconnection between nodes is achieved through a fiber optic structure incorporating a 1:4 splitter and a 1:8 splitter.

In terms of system management, intelligent pluggable manager (IPM) has been deployed on a virtual machine, connected to the routers via a switch. Additionally, an intermediate router serves as a management interface; this device is a whitebox router running ADVA software,

enabling centralized and efficient management of network components, thus facilitating the programming and control of P2MP pluggables within the experimental environment.

For service configuration, a hierarchical controller is employed, which communicates with both an IP SDN controller and an optical SDN controller. The optical SDN controller is located in the CTTC laboratory and connected to the Telefonica's laboratory via a VPN, ensuring seamless integration and control across the distributed experimental setup.



(a) experimental setup. (b) Enabling services.

Figure 3.16: Control of P2MP pluggables, experimental setup.

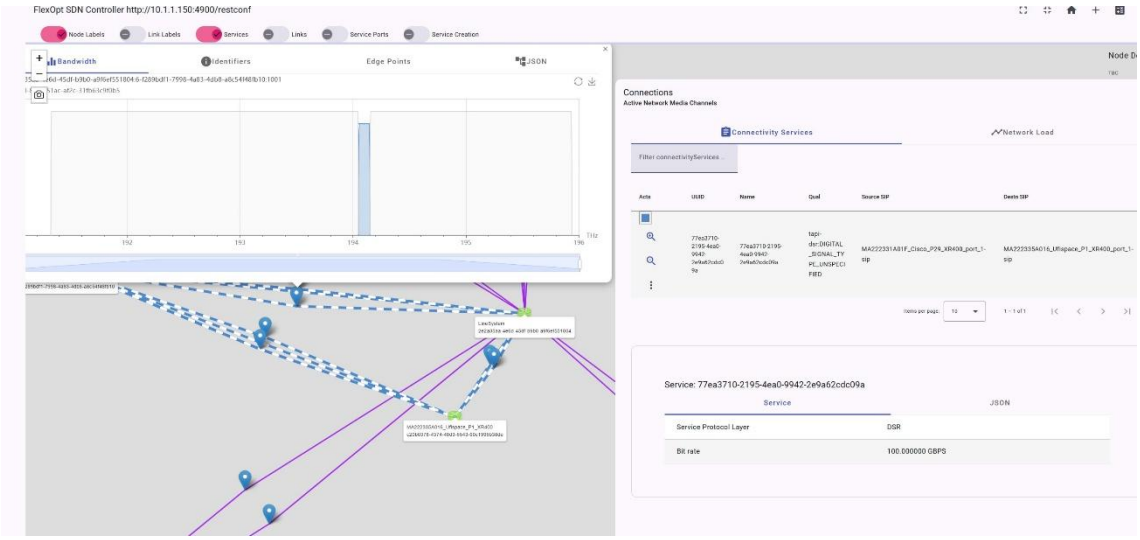


Figure 3.17: GUI of the established P2MP service with the hub and the leaves, showing the centered optical spectrum.

3.2.3.5 Conclusions

We have practically demonstrated the integration of the P2MP pluggable devices into the overall SEASON control plane, showcasing the provisioning of 100 Gb data services in a P2MP scenario by dynamically configuring the XR network parameters and constellation and dynamically

attaching additional leaves to the hub, thanks to the efficiency and flexibility of the digital sub-carrier multiplexing.

This solution was shown during the OFC2025 exhibition demonstration, and the resulting video is available at https://www.youtube.com/watch?v=rbjGV4mpGHU

3.2.4 SDM Passive Optical Network (PON) Controller

The SDM Passive Optical Network (PON) Controller is a dedicated component within the SEASON control plane responsible for managing and configuring advanced PON infrastructures that employ Space-Division Multiplexing (SDM) techniques. This Software-Defined Networking (SDN) controller acts as an intermediary between high-level orchestration systems and the underlying PON hardware, providing a programmable interface that abstracts the complexity of the physical devices and enables flexible service configurations.

From a functional standpoint, the controller supervises optical devices such as Optical Line Terminals (OLTs) and manages spatial aggregation/disaggregation elements. It addresses capacity scaling by exploiting spatial switching by activating or deactivating spatial channels based on traffic demand, while simultaneously maintaining energy efficiency. This dynamic adaptation ensures that PON resources are scaled in line with fluctuating traffic patterns, balancing throughput and sustainability.

3.2.4.1 Architecture and Design of the SDM PON Controller

The SDM Passive Optical Network (PON) Controller is an SDN-based component of the SEASON control plane tasked with managing and configuring PON infrastructures enhanced by space-division multiplexing. It presents a programmable interface to high-level orchestrators, abstracting the underlying optical hardware and spatial switching elements to enable flexible service provisioning. As part of its remit, the controller supervises optical line terminals (OLTs), optical network units (ONUs) and aggregation/disaggregation devices, scaling spatial resources according to traffic demand and network policies. Detailed mechanisms for multi-domain resource allocation and energy-aware operation are described in the following subsections.

3.2.4.2 Interfaces

To provide seamless interoperability between orchestration platforms and the PON equipment, the controller defines two distinct categories, northbound and southbound, each tailored to its respective integration needs. These interfaces are summarized below:

- **Northbound Interface (NBI):** The controller exposes a REST-based API that allows integration with higher-level controllers (e.g., the TeraFlow SDN controller) and orchestrators. This API is documented using OpenAPI/Swagger and supports standard

CRUD operations on network configuration objects, service definitions, and energy monitoring data

- **Southbound Interface (SBI):** Configuration and monitoring of PON devices are performed via NETCONF over SSH, leveraging vendor-specific YANG models (such as Calix) extended with custom models tailored to SDM-PON control. These models encapsulate features such as ONU identification, Ethernet and ONT-Ethernet interfaces, VLAN mapping, traffic profiles, and dynamic bandwidth allocation mechanisms.

3.2.4.3 Models

To enable fine-grained service provisioning over the PON, the controller uses a dedicated YANG schema known as PonCosProfile. This class-of-service model specifies the parameters required to configure VLAN-based service profiles on the PON. The model includes attributes such as a name (string) to identify the profile and a priority (integer) to define its precedence. It also captures a bandwidth_type field, constrained to the value explicit, which indicates that bandwidth allocations are explicitly defined. Upper and lower bandwidth limits are expressed through maximum_bw and min_bw (both integers). Finally, the cos_type (string) designates the class-of-service and can take one of two values: expedited for low-latency, high-priority traffic or assured for guaranteed but less time-sensitive traffic.

```
PonCosProfileModel{
    name                string
                        The name of the PonCosProfile.


    priority            integer
                        The priority to assign.
    bandwidth_type      string
                        example: explicit
                        The cos_type to assign.
                        Enum:
                        [ explicit ]
    maximum_bw          integer
                        The maximum bandwidth
    min_bw              integer
                        The minimum bandwidth.
    cos_type            string
                        example: expedited
                        The minimum bandwidth.
                        Enum:
                        [ expedited, assured ]
}
```

The REST API implemented by the SDM PON Controller exposes a set of endpoints for managing class-of-service profiles, retrieving and modifying the controller's running configuration, and instantiating services. Table 3-1 summarizes the most relevant endpoints, indicating the HTTP method, resource, and corresponding URI. The first block of entries in the table refers to operations on PonCosProfile objects—allowing their retrieval, creation, modification, and deletion—while the subsequent entries relate to queries and updates of the running

configuration and service provisioning via unique service identifiers. These endpoints encapsulate the CRUD functionality described above and form the programmatic entry points through which higher-level orchestrators interact with the PON controller.

Table 3-1:REST API endpoints.

S.No	Interface	Description
1	**REST** PonCosProfile	**METHOD**: GET **URL**: https://IP:PORT/api/v1/pon-cos-profile/{NAME}
2	**REST** PonCosProfile	**METHOD**: POST **URL**: https://IP:PORT/api/v1/pon-cos-profile **BODY EXAMPLE:** `{ "name": "string", "priority": 0, "bandwidth_type": "explicit", "maximum_bw": 0, "min_bw": 0, "cos_type": "expedited" }`
3	**REST** PonCosProfile	**METHOD**: PUT **URL**: https://IP:PORT/api/v1/pon-cos-profile/{NAME} **BODY EXAMPLE:** `{ "name": "string", "priority": 0, "bandwidth_type": "explicit", "maximum_bw": 0, "min_bw": 0, "cos_type": "expedited" }`
4	**REST** PonCosProfile	**METHOD**: DELETE **URL**: https://IP:PORT/api/v1/pon-cos-profile/{NAME}
5	**REST** RunningConfig	**METHOD**: GET **URL**: https://IP:PORT/api/v1/running-config
6	**REST** RunningConfig	**METHOD**: POST **URL**: https://IP:PORT/api/v1/running-config
7	**REST** RunningConfig	**METHOD**: DELETE **URL**: https://IP:PORT/api/v1/running-config
8	**REST** RunningConfig	**METHOD:** POST **URL:** https://IP:PORT/api/v1/service/{SERVICE-ID}

3.2.4.4 Control Functions

- **Resource Allocation:** The controller dynamically allocates resources across time, spectrum, and space based on service requirements and network policies. This includes time-domain scheduling for low-latency services, spectrum assignment to match OLT capabilities, and spatial channel selection to balance capacity and power consumption
- **Energy Monitoring:** To support energy-aware operation, the controller monitors the power consumption associated with activating spatial channels and reports this information via the REST API. This enables higher-level systems to make informed decisions about service placement and resource scaling
- **Service Provisioning:** The controller supports the creation, retrieval, updating, and deletion of PON services. A specific YANG model, PonCosProfile, defines class-of-service profiles with parameters such as name, priority, bandwidth type (e.g., explicit), maximum and minimum bandwidth, and class-of-service type (expedited or assured). Through the REST interface, operators can push these profiles to configure services with precise performance guarantees.

3.2.4.5 Implementation and Demonstration

The SDM PON Controller has been realized as a fully functional SDN-based system that exposes a RESTful northbound API. This interface, documented using the OpenAPI specification and accessible through Swagger, allows administrators and higher-level systems to perform create, read, update and delete (CRUD) operations on network configurations and service profiles. Leveraging the same interface, the controller dynamically adjusts the number of active spatial channels in response to real-time traffic demands—deactivating unused resources during low-load periods and reactivating them when traffic increases—thus optimizing energy consumption without compromising service performance.

On the southbound side, the controller communicates with the PON hardware via NETCONF over secure SSH connections. To ensure interoperability with commercially available equipment, it leverages the Calix YANG model, which identifies optical network terminals (ONTs) by serial number, defines Ethernet and ONT-Ethernet interfaces, and specifies ingress/egress rules and VLAN mappings. Additional custom YANG models extend these capabilities to support SDM-specific operations such as configuring traffic profiles and dynamic bandwidth allocation. Service provisioning is handled through NETCONF using class-of-service profiles; for example, expedited forwarding services are configured by pushing PonCosProfile parameters—name, priority, bandwidth type, maximum and minimum bandwidth, and class-of-service type—to the OLT, ensuring precise traffic shaping and quality of service.

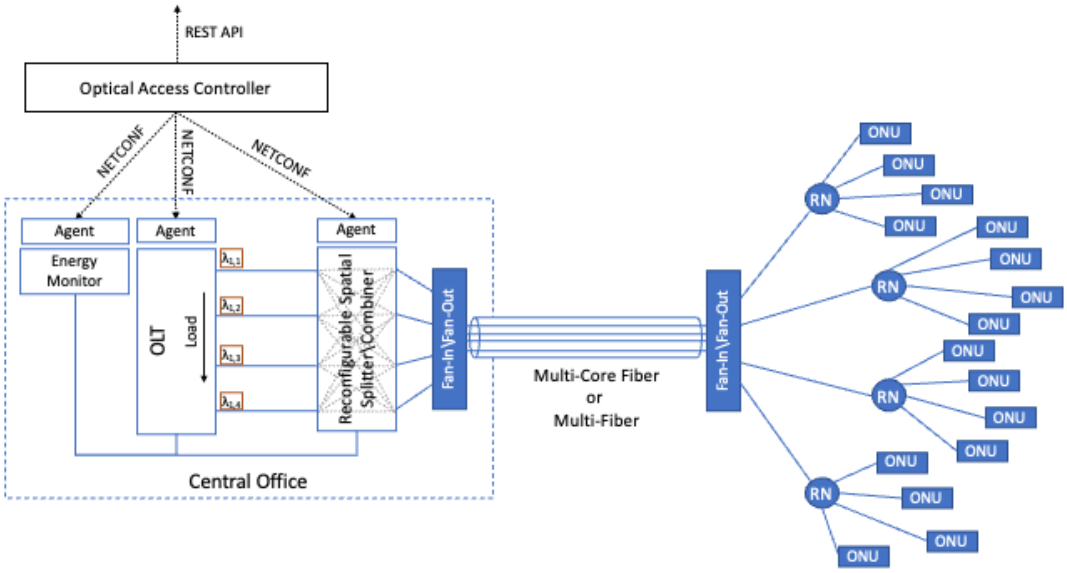


Figure 3.18: Overview of the controller architecture.

Figure 3.18 illustrates this control architecture. The PON Controller interacts via its REST API and NETCONF interfaces and is energy-aware of the OLT. A reconfigurable spatial splitter/combiner; the fan-in/fan-out modules connect the multi-core trunk to remote nodes and ONUs. This dynamic spatial aggregation and disaggregation framework optimizes energy consumption. Using NETCONF-based control and continuous telemetry collection, the system dynamically manages OLT port activity by aggregating or disaggregating spatial channels in response to traffic load. This mechanism allows the PON to switch between low-power/low-capacity and high-power/high-capacity configurations, achieving measurable energy savings without degrading performance. A demonstration using commercially available hardware visualizes power and traffic metrics in real time via a Grafana dashboard, illustrating how the controller's real-time decision-making can optimize both energy usage and throughput in SDM-enabled passive optical networks.

3.2.5 -IPoWDM Controller

This section summarizes the work done in the SEASON project on the SDN Control of IPoWDM networks, also considering the emerging network architectures leveraging on degree-1 ROADMs (also called ROADM-free). Furthermore, this section reports on the novel development to support L2 VPNs within the ETSI TeraFlow (TFS) SDN Controller, thus providing integrated IP and Optical control of the underlying disaggregated network infrastructure. Indeed, TFS is not only responsible for the configuration of the packet layer (IP address configuration, VLAN setup, enforcement of routing protocols and policies) but it is also able to perform the configuration of the optical transmission parameters of the pluggable modules. The TFS IP SDN Controller is able to communicate with IPoWDM devices through NETCONF relying on OpenConfig model. On the

device side, a specific NETCONF agent has been adopted (see Sect. 3.1.2 in this document), in order to enable the control plane communication towards the SDN Controller and to perform the physical resources configuration through CMIS/Sonic APIs. The new functionalities have been experimentally validated in a multi-layer testbed encompassing IPoWDM boxes for the complete deployment of a L2VPN service.

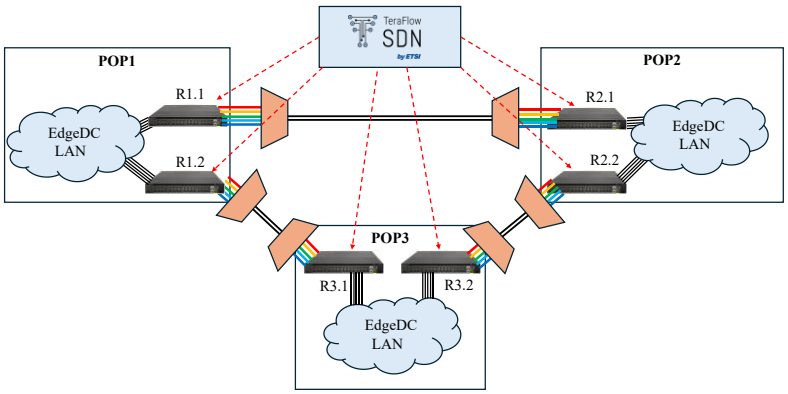


Figure 3.19: Reference ROADM-free scenario with multiple POPs.

Figure 3.19 shows the details of the reference scenario, considering three POPs (i.e., POP1, POP2, and POP3). Each POP is adopting a pair of IPoWDM boxes to act as gateway for the inter-POP connections. Other than grey 100G Ethernet interfaces for the local connections to the EdgeDC LAN devices, the boxes are equipped also with coherent pluggable modules ZR/ZR+ at 400G. The interconnection among the POPs is performed with point-to-point links with the coherent pluggable modules multiplexed together by means of simple AWGs (or ROADMs on blade devices where reconfigurability is needed). Each IPoWDM node is equipped with a NETCONF agent, running in a container and relying on the SONiC-based OS, able to expose a NETCONF interface towards the SDN Controller and relying on CMIS APIs for the communication with the underlying physical resources. The NETCONF agents adopt OpenConfig YANG modeling for the representation (and abstraction) of the node, being capable of performing both the packet (IP addresses, BGP protocol, VLAN assignment) and the optical (frequency, TX-power-level and application code) configuration. At the control-plane, an instance of an IP SDN Controller (relying on ETSI TFS open-source initiative) is considered to perform the configuration of all the IPoWDM nodes of the network. The communication is realized by means of the NETCONF protocol, relying on the OpenConfig model. In the reference scenario, AWG devices are envisioned for the multiplexing over the point-to-point fiber of the coherent channels among POPs. In case of ROADM on blade devices, that require the spectrum (re-)configuration, two main different options are possible: (i) relying on a specific instance of Optical SDN Controller (following the TIP MANTRA approaches), or (ii) following the state-full distributed approach, avoiding the adoption of the Optical SDN Controller.

ETSI TeraFlow is emerging as a strong candidate for the open-source implementation of SDN controller. In particular, this framework proposes a cloud native platform for the deployment of network services, enabling closed loop operations and automatic configuration of the devices. In the context of this activity, novel functionalities have been proposed. Following a bottom-up approach, novel contributions have been proposed at driver, service routines and north bound interface (NBI).

At the driver level, the NETCONF OpenConfig driver has been extended to support IPoWDM scenario. The novel driver includes the automatic discovery of the components required for the activation of the network services, including interfaces, ports, pluggable transceivers, optical-channels. For each component, the relevant details/parameters are automatically discovered during the onboarding of each device, parsing the reply of the NETCONF get RPC. The collected data is then stored in the context database, available during the lifecycle of the system.

From the IP service view, the existing service handler has been expended to support L2VPN activation. This has been achieved by considering a new vendor for the IPoWDM boxes and implementing the required configuration routine.

At NBI level, a new API has been designed for the configuration of the optical parameters of the coherent pluggable modules. In fact, following the single approach of the TIP MANTRA the IP SDN controller is responsible for the full configuration of the IPoWDM boxes, including both the IP stack and the optical parameters of the modules. The API has been conceived to support the control of one or more pluggable modules with a single request.

A payload as shown in Figure 3.20 sent to the NBI REST API to configure a device object related to the pluggable connected to the interface Ethernet192 of node IPoWDM1 with frequency 193.2THz, operational mode 7 and TX-power level 0dBm. The payload can include more than one device object, enabling the possibility to activate more than one pluggable (i.e., configuring the two endpoints of an end-to-end connection or configuring “super-channels” composed of multiple pluggable modules).


```
{
  "devices":[
    {
      "frequency": 193200000,
      "target-output-power": 0,
      "operational-mode": 7,
      "port": "192",
      "device_name": "IPoWDM1"
    }
  ]
}
```


Figure 3.20: Data model of the TFS NBI.

3.2.5.1 Experimental Validation

In order to validate the proposed solution, an experimental testbed including two POPs interconnected with a point-to-point link has been considered. Each POP exploits an IPoWDM EdgeCore box as gateway for the remote connections. Each box has been equipped with two QSFP-DD ZR+ pluggable modules at 400G with coherent capability. Two AWG with 100GHz granularity have been adopted to perform the multiplexing of the channels along a point-to-point link among the two POPs, composed by a spool of SMF fiber with length 45km. The adoption of the AWG allows to filter the signals, still guaranteeing the coexistence of multiple channels with viable quality of transmission. Each IPoWDM box has been equipped with a containerized NETCONF agent and the REST driver for the configuration of the hardware. A TFS IP controller instance has been adopted to control the two IPoWDM boxes.

The TFS controller has been adopted to configure a L2VPN service encompassing the IPoWDM boxes. With the considered hardware (EdgeCore switches with Sonic OS) the MPLS tunneling is not supported. For this reason, the L2VPN is realized by relying on a different tunneling solution (i.e., VXLAN) still adopting BGP exchange of remote MAC addresses and VNI.

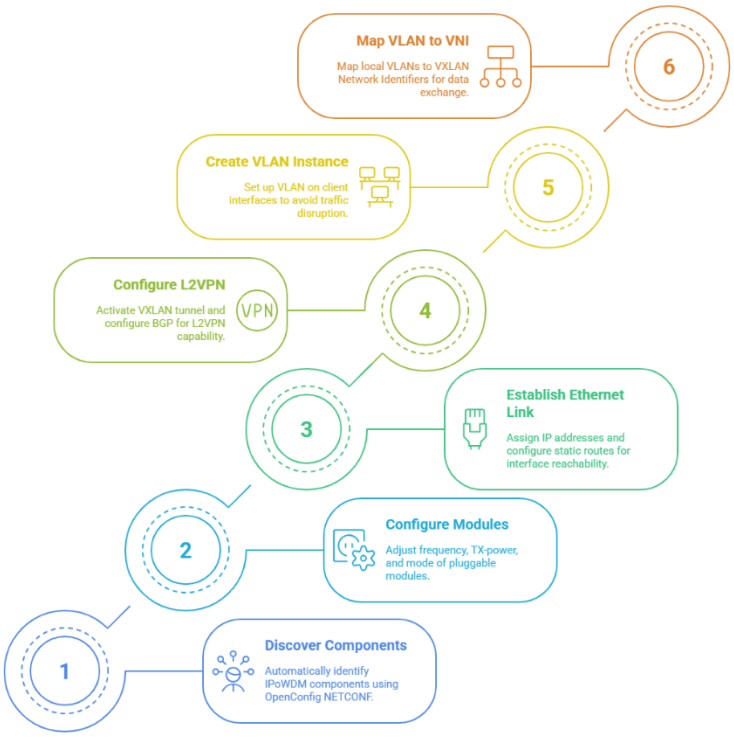


Figure 3.21: complete activation of the L2VPN service.

Following, the overall procedure (as seen in Figure 3.21) for the complete activation of the L2VPN service is described.

1) Initially, the TFS IP controller discovers the IPoWDM components, including ports, interfaces, transceivers, optical-channels and logical-channels, the discovery is automatically performed by the OpenConfig NETCONF driver after the onboarding of the device.
2) Then, relying on the NBI previously described, the TFS controller exposes an API to perform the configuration of the two selected pluggable modules, i.e., the modules attached to port Ethernet192 (associated with component channel-192). In this way frequency, TX-power levels and operational mode are configurated in the two coherent transceivers at 400G. All the time that the pluggable frequency is changed, around 60s are needed for the activation of the packet link among the two interfaces.
3) Once the Ethernet link is ready, an IP address is assigned to the two interfaces (point-to-point network address space). Since only pure BGP-based routing is available in the EdgeCore boxes (no OSPF is implemented in the FRR routing instance) a static route is then configured to enable the Loopback interface reachability. This steps takes around 10.
4) The real L2VPN configuration is then performed. This includes the activation of a vxlan tunnel, with local IP the Loopback interface. The vxlan is configured to act as Network Virtualization Overlay (NVO), supporting the Ethernet Virtual Private Network (EVPN). Then, the BGP instance for the exchange of data related to the L2VPN capability is configured (in our case an iBGP neighboring is activated with AS number 65001). This configuration is performed by the agent relying on the vtysh virtualized cli exposed by the FRR routing instance and takes around 15s for the complete adjacency setup.
5) A VLAN instance is then created (i.e., VLAN 10) on the client interfaces (in the experiment the 10Gbps interfaces of the EdgeCore switches Ethernet256 and Ethernet257 have been considered). To avoid traffic disruption on the other existing interfaces attached to the VLAN, this step performs the shutdown of the interface, the inclusion in the VLAN and again the interface activation. This step takes around 10s.
6) Finally, each local VLAN that will be participating to the L2VPN are mapped to a VXLAN Network Identifier (VNI), being able to perform later the route target for the data exchange among the nodes (in the experiment VLAN 10 is mapped to the VNI 1000 in both the nodes).

```
sonic# show bgp l2vpn evpn
BGP table version is 36, local router ID is 1.1.1.1
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1.1.1.1:2
*> [2]:[0]:[48]:[42:73:f2:d8:d8:8b]
                    1.1.1.1                                32768 i
                    ET:8 RT:65001:1000
*> [2]:[0]:[48]:[42:73:f2:d8:d8:8b]:[128]:[fe80::4073:f2ff:fed8:d88b]
                    1.1.1.1                                32768 i
                    ET:8 RT:65001:1000 ND:Router Flag
*> [2]:[0]:[48]:[56:9b:e0:f2:a2:b4]
                    1.1.1.1                                32768 i
                    ET:8 RT:65001:1000
*> [2]:[0]:[48]:[56:9b:e0:f2:a2:b4]:[128]:[fe80::549b:e0ff:fef2:a2b4]
                    1.1.1.1                                32768 i
                    ET:8 RT:65001:1000
*> [3]:[0]:[32]:[1.1.1.1]
                    1.1.1.1                                32768 i
                    ET:8 RT:65001:1000
Route Distinguisher: 2.2.2.2:2
*>i[2]:[0]:[48]:[50:7c:6f:57:6c:b0]
                    2.2.2.2                     100        0 i
                    RT:65001:1000 ET:8
*>i[2]:[0]:[48]:[50:7c:6f:57:6c:b0]:[128]:[fe80::527c:6fff:fe57:6cb0]
                    2.2.2.2                     100        0 i
                    RT:65001:1000 ET:8
*>i[2]:[0]:[48]:[50:7c:6f:57:6c:b3]
                    2.2.2.2                     100        0 i
                    RT:65001:1000 ET:8
*>i[2]:[0]:[48]:[50:7c:6f:57:6c:b3]:[128]:[fe80::527c:6fff:fe57:6cb3]
                    2.2.2.2                     100        0 i
                    RT:65001:1000 ET:8 ND:Router Flag
*>i[3]:[0]:[32]:[2.2.2.2]
                    2.2.2.2                     100        0 i
                    RT:65001:1000 ET:8
```

Figure 3.22: L2VPN configuration.

Figure 3.22 shows the status of the L2VPN configuration at the end of the procedures. Two BGP nodes are involved IPoWDM node1 (1.1.1.1/32) and IPoWDM node2 (2.2.2.2/32), with different route distinguisher, able to distribute the learned MAC addresses belonging to the L2VPN. In general, the overall L2VPN activation (considering the effective end hosts able to communicate over brand new established data plane packet layer), including coherent pluggable modules frequency reconfiguration, takes around 110s (<2 minutes). In case the configuration of the frequency is not needed, the overall time shorts to 50s.

Conclusions

This work demonstrates that the evolution of control plane solutions is crucial for the efficient deployment of point-to-point IP over DWDM (IPoWDM) architectures, driven by the growing demands of hyperscale cloud services. We have shown that modern IPoWDM nodes in ROADM-free scenarios can be effectively managed by a single IP SDN Controller, which centrally orchestrates both the packet layer and the optical transmission parameters of coherent pluggable modules. The ETSI TeraFlow framework now supports NETCONF protocol with OpenConfig YANG models for IPoWDM devices equipped with NETCONF agents and CMIS/Sonic APIs. The proposed control solution has been experimentally validated in a testbed scenario, demonstrating full automation and correct operation of an end-to-end L2VPN service across two

IPoWDM nodes. Experimental results show that the overall data plane ready configuration time, including optical tuning (frequency and TX power configuration) and IP configuration, requires around 110s, with around 50s needed only by the optical tuning of the coherent pluggable modules. These results confirm the feasibility and operational efficiency of a simplified, ROADM-free IPoWDM network controlled through a unified SDN-based control plane. This work has been published in [Sga25].

3.2.6 RAN Intelligent controller

This section provides a brief overview of the RAN Intelligent Architecture and describes how the RIC, through the deployment of xApps and the use of the E2 interface, is able to monitor the telemetry coming from RAN nodes and then take action based on that telemetry. The section also provides details of the configuration of the network that was deployed at the SEASON trial sites and gives some sample performance metrics.

3.2.6.1 RIC Architecture and the telemetry to action workflow

Figure 3.23, below shows the architecture of Accelleran's RAN Intelligent Controller (RIC) that was deployed at the SEASON trial sites.

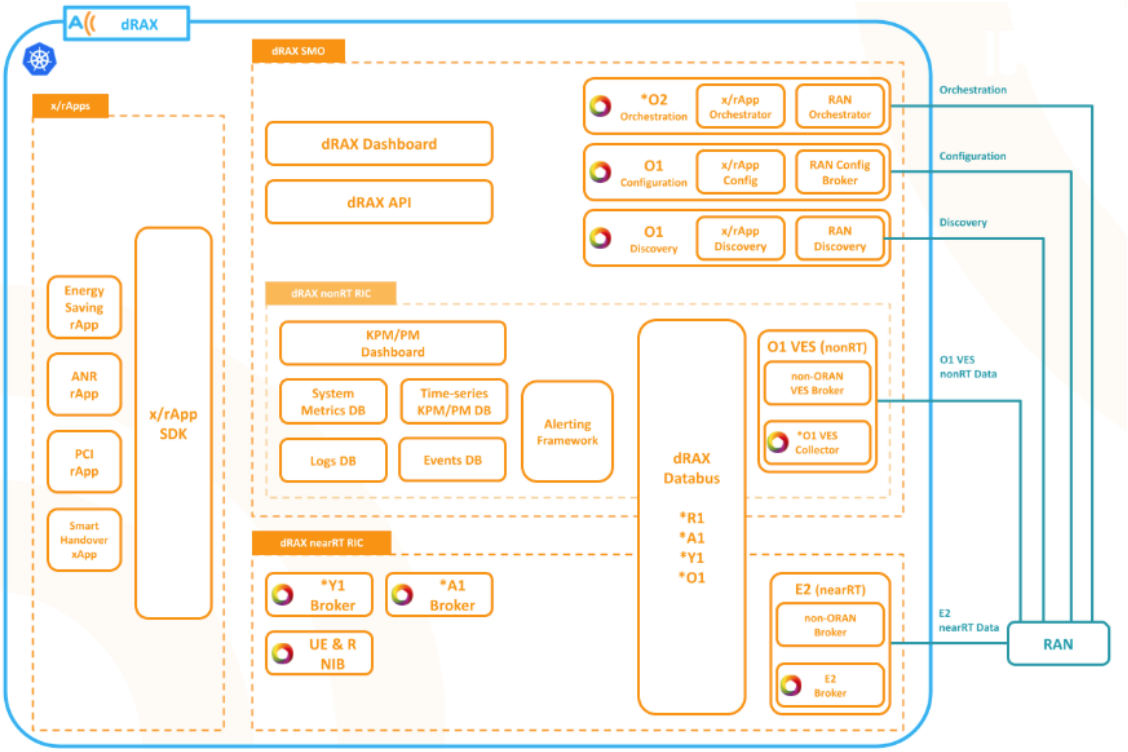


Figure 3.23: Accelleran RIC Architecture.

Dissemination Level	PUB (Public)

The RIC supports the deployment of xApps that collect telemetry and can then, based on that telemetry, decide to perform actions on the deployed network. This telemetry to action workflow happens over the E2 interface. In the O-RAN architecture, the E2 interface connects the Near-Real-Time RIC (Near-RT RIC) to RAN O-CU and O-DU nodes and is responsible for both telemetry collection and control actions. It supports the collection of real-time network data and enables RIC applications (xApps) to make decisions based on live telemetry, such as traffic load, signal quality, or resource usage.

The following sequence describes, in a high-level way, the general telemetry to action workflow.

1. Telemetry collection: The RAN node sends real-time performance data (e.g., user load, interference, throughput) to the Near-RT RIC via E2SM-KPM or other telemetry service models.
2. Decision-making: xApps running on the RIC analyze this data and determine whether action is needed e.g., to reduce congestion, improve QoS.
3. Action triggering: The RIC sends an E2 control message containing the defined E2SM action e.g., initiate handover, adjust transmit power.
4. Execution: The RAN node receives and applies the command and may optionally report back status or acknowledgment.

Further details of the E2 telemetry that is available in the RIC can be found in Section 5.3.

3.2.6.2 Network Configuration and Performance Metrics

The 5G network what was deployed used the following configuration:

- Bandwidth: 50 MHz
- Duplexing Technique: TDD
- TDD pattern: DDDDDSUUUU 5D1S4U
- ARFCN: 641660 (Frequency 3624.9 MHz)
- Frequency Band: n78
- PLMN: 00101

Two RUs were deployed with PCIs of 1 and 2 respectively.

The following figures (Figure 3.24, Figure 3.25, Figure 3.26, Figure 3.27, Figure 3.28, Figure 3.29), taken from the RIC dashboard, provide a representative snapshot of system performance. In all cases a single UE was deployed and active during metrics collection.

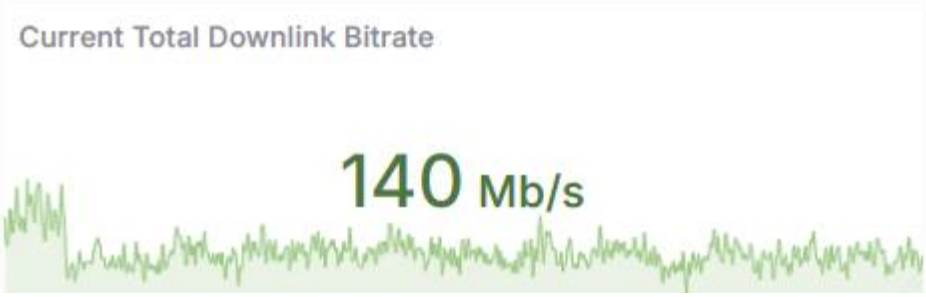


Figure 3.24: Downlink Bitrate (Mbps).

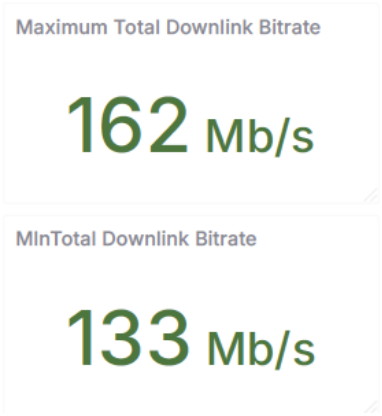


Figure 3.25: Maximum and Minimum Downlink Bitrate (Mbps).

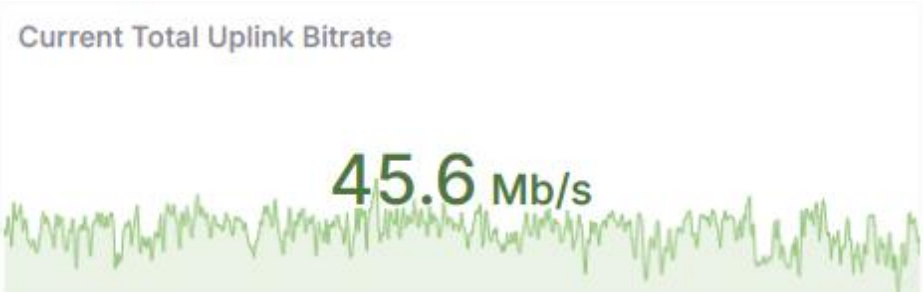


Figure 3.26: Uplink Bitrate (Mbps).

Dissemination Level	PUB (Public)

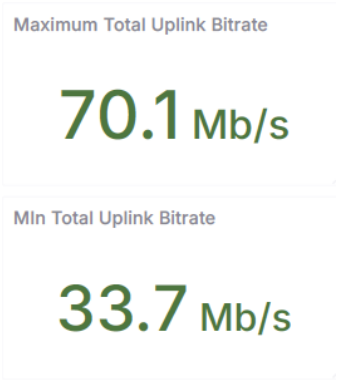


Figure 3.27: Maximum and Minimum Uplink Bitrate (Mbps).



Figure 3.28 Downlink MCS (Modulation and Coding Scheme).



Figure 3.29: Uplink MCS (Modulation and Coding Scheme).

3.3 MULTI-DOMAIN ORCHESTRATION

3.3.1 Transport Network Orchestration

The Transport Network Orchestration (TNO) in SEASON is provided using the Open Source TeraFlow SDN Controller framework and the IETF Network Slice Controller (NSC). TNO maintains a layered view of the transport network, from PON to metro-core, based on the information received by the domain controllers. It can assist in the creation of end-to-end services with the aid of the domain network controllers, which take care of the service creation in their respective

scopes. In the context of SEASON, the main connectivity service that is going to be provided is the “slice” service. Figure 3.30 shows the TNO workflow.

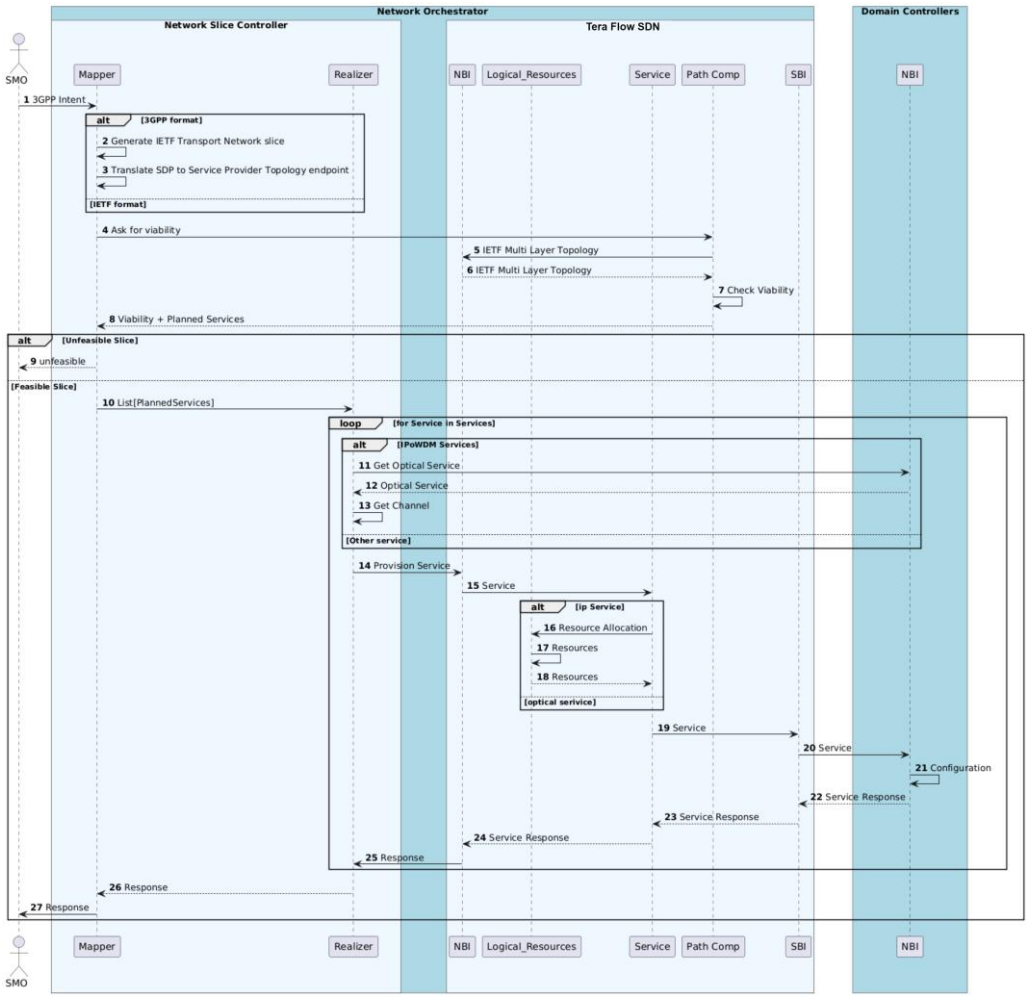


Figure 3.30: Transport Network Orchestration Workflow.

3.3.1.1 Use Case 1: Create/Delete optical slice

The creation and deletion of slices in the optical layer is the task of the NSC. This component is in charge of managing the services in IETF or 3GPP format associated to Optical Slices. These slices must be translated into 3GPP format for the optical controller to make the necessary configurations. To do this translation, the NSC interacts with the TFS Orchestrator to obtain the necessary optical parameters. Figure 3.30 shows an example of an optical slice to be sent to the optical controller.

3.3.1.2 Use Case 2: Create/Delete Connectivity Services

Once the preconfigured optical slices are available, the connectivity services are configured. The services arrive at the NSC in IETF or 3GPP format and are managed to generate different services per domain. Once the domains involved in the configuration have been identified, TFS takes care of talking to each domain controller.

The services are divided into three types of subservices: Optical, IPoWDM and IP:

- Optical: performs the configuration of the media channels and assigns the corresponding optical slice.
- IPoWDM: performs the configuration of the pluggables in the IPoWDM routers, assigning port, frequency, etc.
- IP: performs the connectivity configuration at level 2 or 3.

3.3.2 Cloud-native Service Orchestration

The SEASON service orchestrator is a Kubernetes-native control layer that converts high-level service intents into concrete compute, storage and network resources while shielding applications from the heterogeneity of the underlying optical and packet infrastructure. It is implemented as a lightweight controller that watches for “service-intent” objects inside the cluster, decides where each workload should run, and coordinates with the transport SDN infrastructure to ensure that the required connectivity exists before traffic is released. A small REST North-Bound Interface (NBI) replaces ad-hoc terminal commands. An operator or an OSS component posts a YAML document that names the container image, the number of replicas, and any environment-specific parameters; no explicit knowledge of the physical network is needed at this level. The orchestrator persists the request and immediately begins reconciliation.

During reconciliation the controller obtains a live view of the underlay network through its south-bound interface to the TeraFlow SDN. It receives a stream of network metrics such as bandwidth capacity and current link latency; those figures are continuously joined with pod-level CPU, memory and traffic counters scraped by Prometheus and fed into a short-horizon predictor. When the service orchestrator foresees that, within the next telemetry cycle, throughput or latency will outgrow it instructs HorizontalPodAutoscaler to add replicas and, in parallel, requests the SDN layer to reserve the extra bandwidth or to switch to a lower latency path that those replicas will require.

3.3.2.1 Case Study 1: Automated recovery from an optical fiber failure

This section presents the results of a simulated network failure scenario designed to evaluate the resilience and automated recovery capabilities of the SEASON architecture. The experiment

focuses on a fiber cut event affecting a critical primary path and measures the system's performance in detecting the failure, activating a protection path, and restoring service.

Scenario Overview

The network under test consists of interconnected optical nodes and links, as depicted in Figure 3.31. This topology includes a gNB, Mellanox, edge compute nodes, ROADM ring, MBoSDM and a central office.

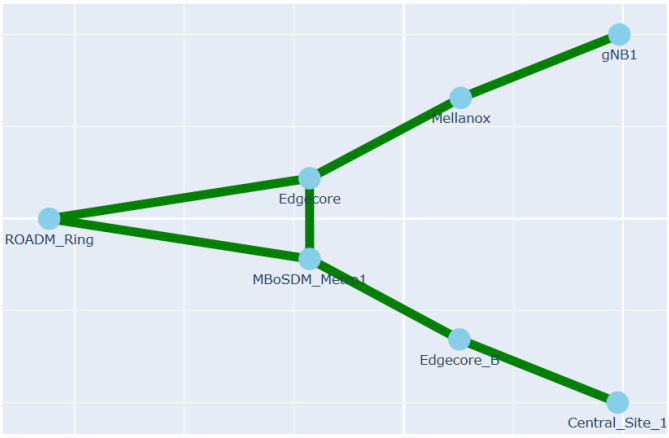


Figure 3.31: Network Topology.

The simulation establishes ten distinct services (SRVC001-SRVC010). These services can be conceptualized as application workloads or microservice instances which require reliable underlying network connectivity. Initially, each active service contributes to a cumulative traffic load on the primary optical path. The scenario unfolds in the following stages:

1. Stable operation: All services operate normally on the primary path with an end-to-end latency of 0.200 ms.
2. Path degradation: At t=79s, traffic load reaches 200 Gbps, causing the primary path to enter a DEGRADED state, shown in Figure 3.32. The system correctly identifies the quality-of-transmission (QoT) drop but maintains service on the primary path as it is still functional.

```
79.0000,TELEMETRY_OPTICALLINK_QOTSTATE_REPORT_COLLECTED,"Source: L_gNB1_Mellanox, Data: {
metric: QoTState, value: DEGRADED}"
79.0000,TELEMETRY_OPTICALLINK_QOTSTATE_REPORT_COLLECTED,"Source: L_Mellanox_Edgecore, Data: {metric: QoTState, value: DEGRADED}"
79.0000,TELEMETRY_OPTICALLINK_QOTSTATE_REPORT_COLLECTED,"Source: L_Edgecore_ROADM, Data: {metric: QoTState, value: DEGRADED}"
```

Figure 3.32: Transition to DEGRADED condition.

3. Hard failure: At t=184s, a FIBER_CUT is simulated on the L_ROADM_MBoSDM link, inducing a total service outage, output shown in Figure 3.33.

```
184.0000,SCENARIO_FAILURE_INTRODUCED,"LinkID: L_ROADM_MBoSDM, FailureType: FIBER_CUT_ON_PRIMARY_LINK"
184.0000,TELEMETRY_OPTICALLINK_QOTSTATE_REPORT_COLLECTED,"Source: L_ROADM_MBoSDM, Data: {
metric: QoTState, value: FAILED}"
184.0000,TELEMETRY_LINKFAILURE_COLLECTED,"Source: L_ROADM_MBoSDM, Data: {link_id: L_ROADM_MBoSDM, failure_type: HARD_FAILURE}"
184.0000,PATH_AFFECTED_BY_FAILURE,"PathID: path_23dec980, LinkID: L_ROADM_MBoSDM"
184.0000,PATH_AFFECTED_BY_FAILURE,"PathID: path_1e3fb54e, LinkID: L_ROADM_MBoSDM"
184.0000,PATH_AFFECTED_BY_FAILURE,"PathID: path_7bf9300f, LinkID: L_ROADM_MBoSDM"
```

Figure 3.33: Transition to FAILURE condition.

4. Automated rerouting: The failure is first detected at a low level, for instance, by an Optical Line System (OLS) controller like the Ensemble OLS Controller, which registers a Loss of Signal alarm from the affected ROADM. This alarm is propagated upwards as a telemetry event to the network orchestrator. The orchestration layer detects the failure and initiates a failover procedure, rerouting affected services to a pre-configured protection path, showing in Figure 3.34 and in Figure 3.35.

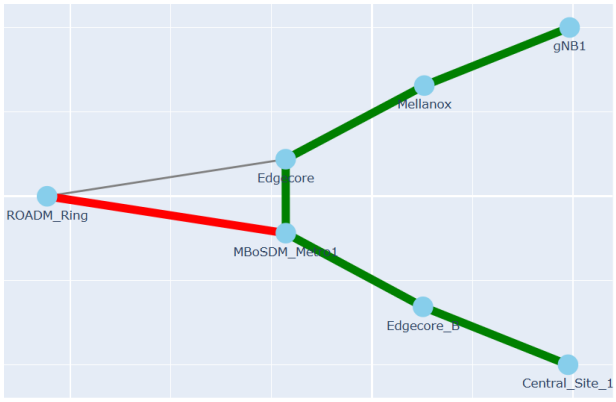


Figure 3.34: Rerouting of the streams due to failed link.

```
194.0000,SERVICE_RESTORED_ON_PROTECTION,"{'ID': 'prot_path_23dec98
194.0000,PATH_E2E_LATENCY,"Service: SRVC001, LatencyMs: 0.250"
204.0000,SERVICE_RESTORED_ON_PROTECTION,"{'ID': 'prot_path_1e3fb54
204.0000,PATH_E2E_LATENCY,"Service: SRVC002, LatencyMs: 0.250"
214.0000,SERVICE_RESTORED_ON_PROTECTION,"{'ID': 'prot_path_7bf9300
214.0000,PATH_E2E_LATENCY,"Service: SRVC003, LatencyMs: 0.250"
224.0000,SERVICE_RESTORED_ON_PROTECTION,"{'ID': 'prot_path_176cb63
224.0000,PATH_E2E_LATENCY,"Service: SRVC004, LatencyMs: 0.250"
```

Figure 3.35: Service restored logs by using the protection path.

5. Path repair and normalization: After a period, the failed link is repaired, and services are automatically switched back to the primary path.

```
349.0000,TELEMETRY_OPTICALLINK_STATE_CHANGE_COLLECTED,"Source: L_ROADM_MBoSDM, Data: {
old_state: FAILED, new_state: NORMAL}"
349.0000,REPAIR NOTIFICATION RECEIVED,Link: L ROADM MBoSDM
```

Figure 3.36: Information about the new state of the link experiencing outage.

```
354.0000,SERVICE_SWITCHED_BACK_TO_PRIMARY,"{'ID': 'path_23dec980',
354.0000,PATH_E2E_LATENCY,"Service: SRVC001, LatencyMs: 0.200"
354.0000,SERVICE_SWITCHED_BACK_TO_PRIMARY,"{'ID': 'path_1e3fb54e',
354.0000,PATH_E2E_LATENCY,"Service: SRVC002, LatencyMs: 0.200"
354.0000,SERVICE_SWITCHED_BACK_TO_PRIMARY,"{'ID': 'path_7bf9300f',
354.0000,PATH_E2E_LATENCY,"Service: SRVC003, LatencyMs: 0.200"
```

Figure 3.37: Service is rerouted from the initial path after the fix.

Key Performance Indicators (KPIs)

Failover Time: The time elapsed from the hard failure (t=184s) to the successful restoration of a service on the protection path.

- First service restored (SRVC001): 10 seconds (at t=194s). This initial 10-second interval includes the time for failure detection at the physical (ROADM/OLS) and protocol layers, propagation of this information to the network orchestrator, orchestrator processing to identify affected services and available protection paths, and signaling the path change to network elements.
- Last service restored (SRVC010): 100 seconds (at t=284s). The results show a sequential restoration process. After the initial 10-second overhead for the first service, subsequent services were restored at approximately 10-second intervals. This sequential approach is a common strategy to manage control plane load, ensure resource stability on the protection path, and allow for monitoring during a mass failover event. Thus, the total service restoration window for all ten services spanned 90 seconds (from t=194s for SRVC001 to t=284s for SRVC010).

Service Switchback

Following the repair of L_ROADM_MBoSDM (completed at t=349s, Figure 3.36), services were seamlessly switched back to the primary path at t=354s. The 5-second delay between link repair validation (t=349s) and the initiation of switchback (t=354s, Figure 3.37) represents a Wait-to-Restore (WTR) timer to ensure the repaired path is stable before reverting traffic, preventing potential service flapping if the repair were intermittent. Upon switchback, the original E2E latency of 0.200 ms was restored for all services, as shown in Figure 3.38.

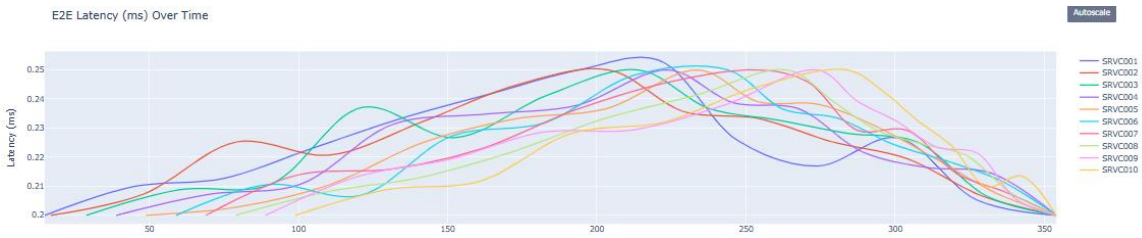


Figure 3.38: Latencies experienced by the services.

Integrated Orchestration and Network Control

The SEASON architecture is designed for seamless interplay with higher-level service orchestrators, such as Kubernetes and TFS. This integration is key to achieving agile and resilient end-to-end service delivery.

- **Facilitating rapid service creation and upscaling**: The prompt provisioning of network paths by SEASON is a critical enabler for the overall service creation time. When a new application service is requested or an existing one needs upscaling, SEASON rapidly establishes the necessary underlying network connectivity.
- **Ensuring stability for orchestrator decisions**: During network events like the failover (t=184s to t=284s), SEASON's automated network restoration minimizes the disruption perceived by the orchestrator. While Kubernetes health checks might detect temporary unreachability for some pods, the network's quick self-healing (first service restored in 10s) prevents the orchestrator from triggering premature or unnecessary drastic actions like widespread pod rescheduling or downscaling. This network stability allows the orchestrator to make more informed decisions based on actual application state rather than transient network issues.
- **Streamlined operations through layered automation:** By autonomously managing network path selection, failure recovery, and QoT assurance, SEASON allows the service orchestrator to focus on its core responsibilities: application deployment, scaling, and lifecycle management. Kubernetes can operate with the assurance that the underlying network is intelligent and responsive. This layered automation simplifies operations and reduces the need for complex, cross-layer control logic to be embedded within the orchestrator itself. For example, the WTR timer (t=349s to t=354s) is a network-level decision by SEASON that ensures path stability before Kubernetes would observe the primary path as fully restored for its managed pods.
- **Holistic resilience and performance:** The synergy between SEASON's network automation and the orchestrator's service management capabilities results in enhanced end-to-end resilience and performance. Network path optimization and rapid recovery directly benefit application **pod** connectivity and user experience. The data-driven KPIs

from this simulation confirm that the SEASON architecture provides a robust foundation for such integrated, orchestrated service environments.

3.3.2.2 Case Study 2: Proactive resilience under RAN-initiated demand surge

This section evaluates the architecture's performance under a demand-surge scenario, a common operational challenge in 5G networks. Unlike the reactive failure recovery detailed previously, this experiment tests the system's proactive capabilities. It assesses the complete, automated workflow from demand detection in the RAN, through cross-domain orchestration, to the coordinated scaling of both application and network resources to prevent service degradation.

Scenario Overview

The experiment utilizes similar network topology (Figure 3.39) but focuses on a different operational sequence. The network initially supports a baseline traffic load from four distinct gNBs (gNB1-gNB4). In this stable state, only one Ten UEs per gNB are active, each generating a steady 100 Mbps data stream. This results in a predictable, low-level aggregate throughput of approximately 1 Gbps, with a stable end-to-end latency of 0.200 ms for all associated services.

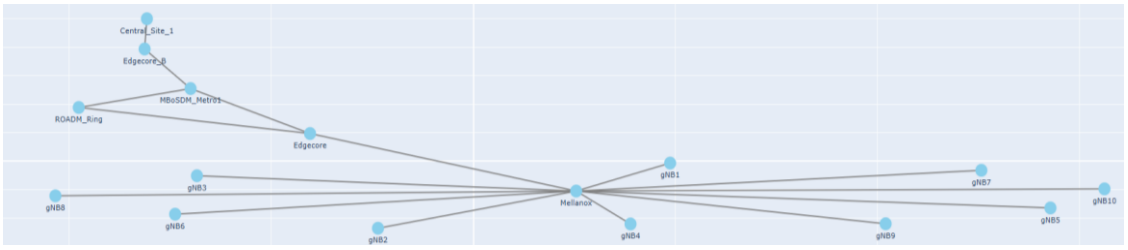


Figure 3.39: Network topology.

The simulation unfolds through the following critical stages, which are visualized in the dashboard plots referenced throughout this section:

1. **Stable operation (t=0s to t=78s):** The system operates under normal, low-load conditions. Per-gNB traffic is stable at 1 Gbps, and latency is nominal, as shown in the initial phase of the plots in Figure 3.40a (Link Traffic Load) and Figure 3.40b (Per-Service Latency).

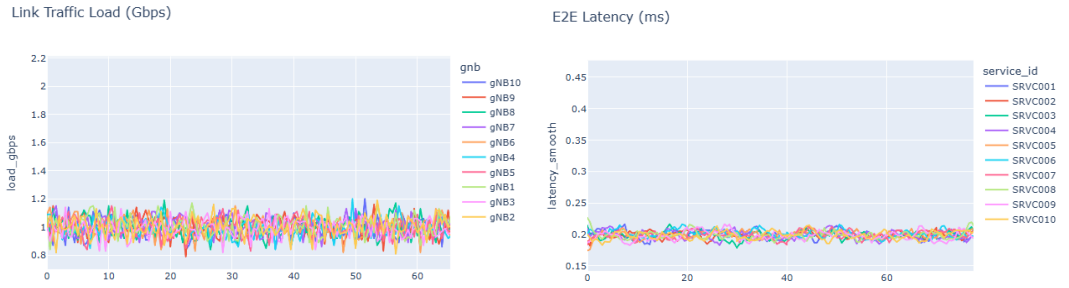


Figure 3.40: Throughput (a) and latency (b) experienced by the services before the traffic surge.

Sudden demand surge (t=79s): A coordinated surge event is triggered, simulating a flash-crowd scenario where ten UEs at each of the ten gNBs begin streaming high-bandwidth content simultaneously. The RAN Intelligent Controller (RIC) controller at each gNB instantly detects this ten-fold increase in local demand. It immediately sends a RAN_DEMAND_SURGE_DETECTED report to the service orchestrator (visualized in the communication timeline). This message is critical, as it serves as the trigger for the entire cross-domain scaling response.

2. **Transient bottleneck and congestion (t=79s to t=90s):** The aggregate traffic demand abruptly jumps to 20 Gbps. The existing optical path, sized for the previous load, becomes a bottleneck. During this transient interval:
   - **Throughput:** The aggregate throughput is constrained by the available capacity, plateauing at the physical limit but exhibiting significant jitter as network buffers absorb the excess traffic (Figure 3.41a, aggregate throughput).
   - **Latency:** End-to-end latency for all services begins to climb smoothly from 0.200 ms toward a peak of approximately 0.6 ms (Figure 3.41b). This controlled increase, rather than a sharp spike, reflects the system's ability to manage congestion through buffering while the scaling solution is being orchestrated.

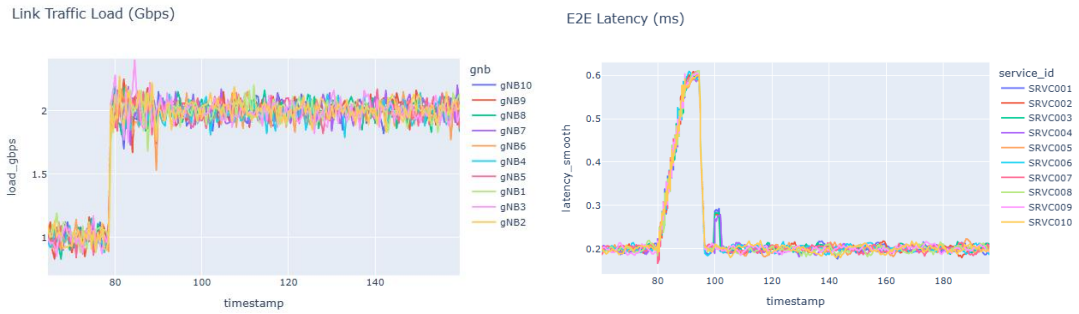


Figure 3.41: Throughput (a) and latency (b) experienced by the services after the traffic surge.

3. **Automated, Multi-Layer Resource Scaling (t=80s to t=95s):** The orchestrator, having received the surge alerts, executes a coordinated, multi-layer scaling strategy:

- **Optical network reprovisioning (t=80s - t=90s):** The transport-layer orchestrator immediately processes the surge reports. It calculates the required additional capacity and, at t=80s, begins the required provisioning. This involves the TFS commanding the underlying OLS controller to light up additional wavelengths or instructing an MBoSDM controller to allocate an additional fiber core to the path. This process completes at t=90s, marked by a RESOURCE_SCALE_UP event, at which point the underlying network capacity is significantly expanded (Link Capacity).
- **Service-layer autoscaling (t=92s):** Concurrently, the service orchestrator (simulating a Kubernetes Horizontal Pod Autoscaler, or HPA) responds to the increased application load and CPU/memory pressure. At t=92s, it triggers a POD_SCALE_UP event, increasing the number of replicas for the media-edge service from to handle the additional processing demand (Figure 3.42).

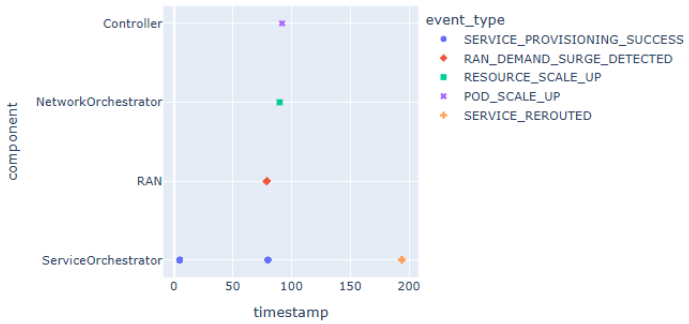


Figure 3.42: Actions taken during the sudden traffic surge from the network and service orchestrators.

4. **System stabilization at new capacity (t > 95s):** With both network and application resources scaled, the system reaches a new, stable equilibrium. Latency for all services recedes and stabilizes well within typical performance thresholds. The per-gNB throughput remains at a steady 10 Gbps, now comfortably accommodated by the expanded infrastructure.

Integrated orchestration and cross-layer synergy

This experiment highlights the critical importance of synergy between the service orchestrator, the transport network orchestrator, and the RAN:

- RAN as a sensor for proactive scaling: The architecture leverages the RIC not just for connectivity, but as a critical real-time sensor of user demand. By translating local UE activity into a standardized RAN_DEMAND_SURGE_DETECTED event, it provides the transport layer with the predictive insight needed to act before severe congestion leads to widespread packet loss and a collapse in application performance.

- Decoupled but coordinated scaling: The simulation demonstrates a highly effective, decoupled scaling model. The transport network focuses on its core competency: rapidly provisioning bandwidth. The service orchestrator focuses on its own: scaling application instances. The success of this approach lies in the timing as rapid optical provisioning ensures that by the time the new pods are ready to accept traffic, the underlying network infrastructure is already in place to support them.

3.3.3 Integration with Transport Network Orchestration

TeraFlowSDN (TFS) provides transport-layer orchestration across multi-domain, multi-technology networks, including packet and optical layers. In SEASON, Kubernetes governs the service plane while TFS governs the transport plane, and the integration between them translates service intents into concrete transport configurations. The project adopts the IETF Network Slice model so high-level requirements, such as bandwidth guarantees and latency budgets, can be expressed as intents, without embedding transport-specific details into the service layer. TFS then maps these intents into path computations and device configurations across the transport infrastructure, providing the end-to-end connectivity properties required by applications such as enhanced Mobile Broadband.

The current proof-of-concept uses a pragmatic REST integration to bridge the orchestrator with TFS. TFS exposes simple endpoints to the dashboard and control logic, forwarding topology and telemetry requests to service orchestrator based on IETF slice semantics. The adapter includes a minimal HTTP client configured by environment variables so it can target the appropriate TFS instance and authentication scheme. Although this initial bridge relies on REST, the target architecture anticipates gRPC for low-latency, bi-directional interactions once the requisite northbound services and protobuf schemas are available. gRPC's use of HTTP/2 and efficient binary serialization is well suited to streaming telemetry and event notifications from TFS to the service orchestrator, while also supporting rapid intent acknowledgments and status updates in the opposite direction. As these interfaces stabilize, the system can shift from periodic polling to event-driven control loops, closing latency gaps and improving responsiveness under fluctuating network conditions. In effect, SEASON achieves a clean separation of concerns—Kubernetes scaling and healing services, TFS provisioning and assuring transport, and Kafka carrying real-time signals between them while maintaining a tight, standards-aligned feedback loop that delivers the performance and adaptability required by advanced applications.

A) IETF Topology Interface

The IETF topology interface allows the TNO to retrieve and manage network topology using standard IETF models, such as Network Slice. TNO exposes REST endpoints so that orchestrator

dashboards and control logic can request information on network nodes, links, and connectivity metrics.

Example:

Service orchestrator sends a GET request to /restconf/data/ietf-network:networks to obtain a graph of all active optical and packet nodes, formatted according to IETF YANG models. This data is used for path computation and network visualization.

B) IETF Intents Interface

The IETF intents interface enables services to define network requirements using abstract, high-level statements (intents), following IETF Network Slice semantics. TNO receives these intents and autonomously computes paths and configures devices to meet the specified connectivity guarantees.

Example:

An orchestrator transmits an intent with parameters such as "bandwidth": 1000 Mbps, "latency": 5 ms, "slice-type": "eMBB" to /nsc/slice. TNO interprets this request, performs multi-domain path computation, and allocates resources across optical and packet domains to fulfill the application's needs without exposing underlying technology details.

4 OPERATIONAL, PLANNING AND RESOURCE ALLOCATION TOOLS

4.1 OCATA DIGITAL TWIN FOR MULTIBAND

C+L+S multiband (MB) optical transmission has the potential to increase the capacity of optical transport networks, and thus it is a possible solution to cope with the traffic increase expected for the years to come. However, the introduction of MB optical technology needs to come together with the needed tools that support network planning and operation. In particular, Quality of Transmission (QoT) estimation is needed for provisioning optical MB connections. In this section, we concentrate on modelling MB optical transmission to provide fast and accurate QoT estimation and propose Machine Learning (ML) approaches based on neural networks, which can be easily integrated into an Optical Layer Digital Twin (DT) solution. We start by considering approaches that can be used for accurate signal propagation modelling. Even though solutions like the Split-step Fourier method (SSFM) for solving the non-linear Schrödinger equation (NLSE) have limited application for QoT estimation during provisioning because of their very high complexity and time consumption, they could be used to generate datasets for ML model creation. However, even that can be hard to be carried out on a fully loaded MB system with hundreds of channels. In addition, in MB optical transmission, the Inter-channel Stimulated Raman Scattering (ISRS) becomes a major effect, which adds more complexity. In view of that, the Fourth Order Runge-Kutta in Interaction Picture (RK4IP) method is evaluated as an alternative to reduce time complexity, which is complemented with an adaptive step size algorithm to further reduce the computation time. We show that RK4IP provides an accuracy comparable to that of SSFM with reduced computation time, which enables its application for MB optical transmission simulation. Once datasets are generated using the adaptive step size RK4IP method, two ML modelling approaches are considered to be integrated in the OCATA DT, where models predict optical signal propagation in the time domain. Being able to predict the optical signal in the time domain, as it will be received after propagation, opens opportunities for automating network operation, including connection provisioning and failure management. In this section, we focus on comparing the proposed ML modeling approaches in terms of models' and QoT estimation accuracy.

4.1.1 Numerical Methods for C+L+S Optical Signal Propagation

In this section, we consider a C+L+S MB transmission system as the one illustrated in Figure 4.1, where a lightpath of total distance d connects sites A and Z using channel λ_p. The transponders (TP) include a transmitter (Tx) and a receiver (Rx), and we assume that it may operate in the different transmission bands. The generated signals are multiplexed and transmitted over the optical fiber, where MB optical amplifiers (OA) amplify the optical signal at every span. Two optical spans of length l_1 and l_2, respectively can be observed in Figure 4.1. We consider a MB

OA as a system with one OA per band, e.g., an EDFA for the C and L bands and a thulium-doped fiber amplifier (TDFA) for the S band, together with waveband (de)multiplexers.

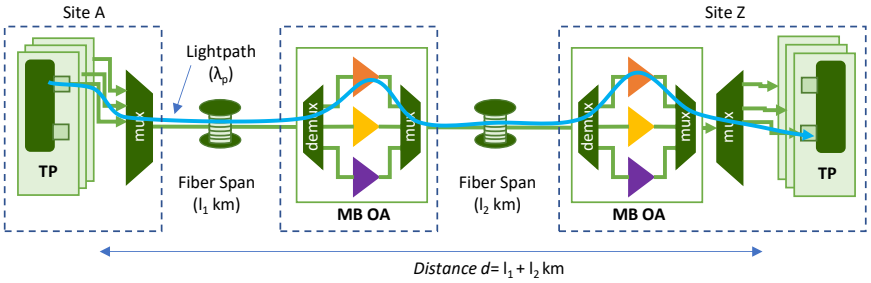


Figure 4.1: General scenario of C+L+S MB transmission. Illustrative example of a lightpath.

The propagation of the MB WDM optical signal travelling through the optical fiber can be modelled using the NLSE. In this regard, it is worth noting that channels in each of the C+L+S bands are impacted differently by not only linear effects like dispersion and fiber attenuation, but also by the ISRS effect, that potentially increases non-linearities.

In the next subsections, we first introduce the NLSE and then present the SSFM, which is probably the most popular approach for solving the NLSE. In view of the high computational complexity of using the SSFM on fully-loaded MB systems, we propose the RK4IP method, as an alternative approach for solving the NLSE faster. Finally, an adaptive step size algorithm is presented to further reduce the computation time.

C) Split-Step Fourier Method

The NLSE is the most accurate tool for modeling the propagation of optical signals along the optical fiber and includes both linear and non-linear transmission effects. We concentrate in the most relevant effects, like chromatic dispersion and fiber attenuation linear effects and self-phase modulation and cross-phase modulation non-linear Kerr-effects. In addition, the energy transferred from incident photons to the vibrational modes, results in the generation of new photons with different frequencies, which originates the ISRS and the self-steepening effect.

In this case, the NLSE modeling the signal propagation in the time domain (t) can be written as eq. (1), where ω_0 is the carrier frequency, coefficients $\beta_{(\cdot)}$ represent the different order of dispersion, $A(z,t)$ is the field of the optical signal at propagation distance z, $R(t)$ is the non-linear response function, and γ is the non-linear coefficient.

$$\frac{dA(z,t)}{dz}+\frac{\alpha}{2}A(z,t)+\beta_1\frac{dA(z,t)}{dt}+i\beta_2\frac{d^2A(z,t)}{dt^2}-\beta_3\frac{d^3A(z,t)}{dt^3}= \\ =i\gamma\left(1+\frac{i}{\omega_0}\frac{d}{dt}\right)\left(A(z,t)\int R(t')\cdot \mathrm{A}\left(\mathrm{z},(\mathrm{t}-t')\right)^2dt'\right) \quad (1)$$

$R(t)$ can be computed as equations (2)-(5), where $h_R(t)$ is the Raman response function, f_R is the peak Raman gain, and parameters $f_{(\cdot)}$ and $\tau_{(\cdot)}$ adjust the Raman response function.

$$R(t) = (1 - f_R)\delta(t) + f_R h_R(t) \tag{2}$$

$$h_R(t) = (f_a + f_c)h_a(t) + f_b h_b(t) \tag{3}$$

$$h_a(t) = \tau_1(\tau_1^{-2} + \tau_2^{-2})e^{-(t/\tau_2)}\sin(t/\tau_1) \tag{4}$$

$$h_b(t) = [(2\tau_b - t)/\tau_b^2]e^{(t/\tau_b)} \tag{5}$$

In equation (1), we observe that the amplitude of the optical signal depends on both transmission distance and time. In consequence, two sources of numerical error can be identified when solving the NLSE coming from the size of the related steps. The numerical error coming from the time derivative (which also depends on the distance step size) can be circumvented by re-writing eq. (1) in the frequency domain (ω) as:

$$\frac{d\tilde{A}}{dz} + i\tilde{A}\cdot(\beta(\omega) - \beta_0 - \beta_1\cdot\omega) = \\ = -i\gamma\cdot\left(1 + \frac{\omega}{\omega_0}\right)\cdot F\left[(1 - f_R)\cdot A\cdot|A|^2 + f_R\cdot A\cdot F^{-1}\left[\tilde{h}_R\cdot F[|A|^2]\right]\right] \tag{6}$$

where,

$$\tilde{A}(z,\omega) = FT(A(z,t)) \tag{7}$$

$$\tilde{h}_R = FT\big(h_R(t)\big) \tag{8}$$

and FT denotes the Fourier Transform operator.

The Split-Step Fourier Method (SSFM) is the most common method for solving the NLSE. SSFM relies on computing the solution in small steps, while computing the linear and the nonlinear parts separately; the linear part is solved in the frequency domain, whereas the nonlinear part is solved in the time domain.

D) Fourth Order Runge-Kutta in Interaction Picture Method

The RK4IP method can be expressed in the time domain as:

$$A(z,t) = F^{-1}\left[exp(\frac{h}{2}\cdot D)\cdot F[A_1 + K_1/6 + K_2/3 + K_3/3]\right] + K_4/6 \tag{9}$$

$$A_1 = F^{-1}\left[exp(\frac{h}{2}\cdot D)\cdot\tilde{A}(z,\mathrm{t})\right] \tag{10}$$

$$K_1 = F^{-1}\left[exp(\frac{h}{2}\cdot D)\cdot F[h\cdot N(A(z,t))]\right] \tag{11}$$

$$K_2 = h \cdot N\left(A_1 + \frac{K_1}{2}\right) \tag{12}$$

$$K_3 = h \cdot N\left(A_1 + \frac{K_2}{2}\right) \tag{13}$$

$$K_4 = F^{-1}\left[h \cdot N(exp\left(\frac{h}{2} \cdot D\right) \cdot F[A_1 + K_3])\right] \tag{14}$$

where D is the dispersion operator, $K_{(\cdot)}$ are the RK method coefficients, and $N(\cdot)$ is the non-linear operator.

The RK4IP method can also be applied in the frequency domain, as:

$$\tilde{A}(z,\omega) = exp(\frac{h}{2} \cdot D)(\tilde{A}_1 + \tilde{K}_1/6 + \tilde{K}_2/3 + \tilde{K}_3/3) + \tilde{K}_4/6 \tag{15}$$

$$\tilde{A}_1 = F^{-1}\left[exp(\frac{h}{2} \cdot D) \cdot \tilde{A}(z,\omega)\right] \tag{16}$$

$$\tilde{K}_1 = exp\left(\frac{h}{2} \cdot D\right) h \cdot \tilde{N}\left(\tilde{A}(z,\omega)\right) \tag{17}$$

$$\tilde{K}_2 = h \cdot \tilde{N}\left(\tilde{A}_1 + \frac{\tilde{K}_1}{2}\right) \tag{18}$$

$$\tilde{K}_3 = h \cdot \tilde{N}\left(\tilde{A} + \frac{\tilde{K}_2}{2}\right) \tag{19}$$

$$\tilde{K}_4 = h \cdot \tilde{N}(exp\left(\frac{h}{2} \cdot D\right)(\tilde{A}_1 + \tilde{K}_3)) \tag{20}$$

E) Adaptive step size algorithm

An adaptive step size algorithm can be defined to further reduce the computation time required by the SSFM and the RK4IP method. Algorithm 4-1 sketches the pseudocode. The algorithm receives as input the current step size (Δz) and the global error/tolerance (δ_G) and return the new step size. The algorithm first computes the signal waveform error (δ) (line 1 in Algorithm 4-1) and then updates the step size accordingly (lines 2-4). The algorithm ensures that the step size does not exceed a maximum value specified by *MAX*_Δz (line 5) and returns the new step size (line 6).

Algorithm 4-1 Adaptive Step Size algorithm for MB WDM signal propagation.

INPUT: Δz, δ_{Gf}
OUTPUT: Δz
1: $\delta \leftarrow LEM(\cdot)$
2: **if** $\delta < \delta_G /2$ **then** $\Delta z_{new} \leftarrow \Delta z * 2^{\eta}$
3: **else if** $\delta > 2\ \delta_G$ **then** $\Delta z_{new} \leftarrow \Delta z / 2$

4: **else** $\Delta z_{new} \leftarrow \Delta z$

5: **if** $\Delta z_{new} > MAX_\Delta z$ **then** $\Delta z_{new} \leftarrow MAX_\Delta z$

6: **return** Δz_{new}

We consider the Local Error Method (LEM) approach to estimate δ at each step. LEM estimates the step size at each iteration by using two solutions of equation (15), the coarse $\tilde{A}_c(z, \omega)$ and the fine $\tilde{A}_f(z, \omega)$ ones. Solution $\tilde{A}_c(z, \omega)$ is calculated over a single full step of size d, whereas solution $\tilde{A}_f(z, \omega)$ is computed by taking two smaller steps of size $d/2$. It starts with the initial condition $\tilde{A}_1$ and propagates the signal over the first half-step to obtain an intermediate solution $\tilde{A}_{mid}$. Then, $\tilde{A}_{mid}$ is used as the initial condition to propagate across the second half-step. Finally, the two half-steps are combined to produce the fine solution $\tilde{A}_f(z, \omega)$. The local error δ is computed as:

$$\delta = \left\|\tilde{A}_f(z,\omega) - \tilde{A}_c(z,\omega)\right\| / \left\|\tilde{A}_f(z,\omega)\right\| \quad (21)$$

4.1.2 Machine Learning models

In this section, we focus on modelling optical signal propagation by applying ML techniques to produce estimate outputs in a fast way. The models can be integrated into the OCATA time domain digital twin and they can be used for network operation applications, like QoT estimation during provisioning and failure management.

A) OCATA MB Time Domain DT Modelling and QoT estimation

Figure 4.2 presents the OCATA DT architecture for MB, where incoming optical signals generated using modulation format *m*-QAM are processed in the time domain, i.e., IQ optical constellation samples *X* are used as inputs. Specifically, an input sample *X* consist of a set of symbols x∈*X*, where each x=[x^I, x^Q] belongs to one of the *m* distinct constellation points (CP). A features extraction (FeX) module computes a set of features *Y* characterizing each CP of the received signal by applying Gaussian mixture model (GMM) fitting. For such characterization, we assume that every CP follows a bivariate Gaussian distribution, so *Y* contains 5 features for each CP *i*, i.e., $Y^i = (y)^i = (\mu^I, \mu^Q, \sigma^I, \sigma^Q, \sigma^{IQ})^i$, where: 1) μ^I and μ^Q represent the mean position for *i* over I and Q axes, respectively; and 2) σ^I and σ^Q are the variance of *i* over the axes and σ^{IQ} is the covariance between the axes. The dispersion of symbols belonging to each CP provides valuable insight of the level of both linear (LI) and NLI noise affecting the signal, which can be related to QoT-related indicators like pre-FEC BER and SNR.

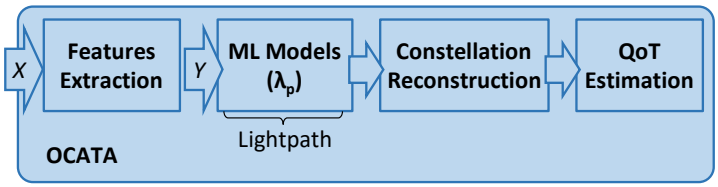


Figure 4.2: Main building blocks of the OCATA-MB digital twin.

Sample generation and FeX can be conveniently combined into a single Feature generation module. Algorithm 4-2 presents such feature generation that produces the initial features to be propagated afterwards. The algorithm receives as input the Tx model, the required Tx configuration, e.g., the modulation format, and the length of the PRBS to be generated. After generating the bit sequence (line 1), the initial IQ optical constellation *X* is obtained by sampling, shaping and modulating the bit sequence following the received Tx configuration (line 2). Next, GMM fitting is performed to compute the set of bivariate Gaussian distribution that better fits the generated optical constellation *X* (line 3) and the computed features are returned (line 5).

Algorithm 4-2. Features Generator.

INPUT: *Txmodel*, *Txconfig*, *n_bits*
OUTPUT: *Y*
1: *bitSequence* ← PRBS(*n_bits*)
2: *X* ← *Txmodel*.generateIQConst (*bitSequence, Txconfig*)
3: *Y* ← GMMfitting(*X*)
4: **return** *Y*

The feature generation is then followed by a ML model that estimates the changes in the input features as a result of the propagation of the signal on the optical components in the route of the lightpath. In contrast to the classical OCATA DT for C-Band transmission, where ML models are assumed as independent of the specific channel, models need to be trained for the specific channel of interest (λ_p) in the case of MB optical transmission.

We evaluate two methodologies for modelling signal propagation: *i*) end-to-end lightpath models, which get as inputs features *Y* from the Tx and estimates the features at the Rx after the signal propagates along the route of the lightpath over a distance *d*; and *ii*) single optical spans that include an OA and an optical fiber of a given length. Span models receive a set of input features *Y* from the signal generated by the Tx or after the signal has traversed some previous optical spans, and produces as output the value of the features characterizing the IQ optical constellation of the optical signal after the propagation through the optical span. In this case, models for end-to-end lightpath are created by concatenating span models as defined by the route of the lightpath.

Two ML-based approaches are considered: *i*) FLANN, which because of their simplicity can be used to model end-to-end lightpaths; and *ii*) DNN that produce accurate models but, being more complex than FLANNs, are proposed for modeling optical spans.

To reduce the number of inputs and outputs, as well as to simplify the structure of the ML models, especially in the case of the DNNs, a subset of selected CPs (*selCP*) is considered for the propagation of the features. Precisely, two exterior and two interior CPs are and, from them, a Constellation Reconstruction block reconstructs the features for the non-propagated CPs with high accuracy (eq. (22)). Functions $F^i(\cdot)$ can be defined as linear combinations of the propagated features.

$$Y^i = F^i([Y^j, \forall j \in selCP]) \quad \forall i \in CP \tag{22}$$

Finally, the QoT estimation block predicts the QoT of the lightpath from the features of the received signal. Specifically, the parameter Φ^i_{out} is defined as:

$$\Phi^i_{out} = 1 - P\left(x \subset A^i \middle| x \sim \mathcal{N}(Y^i)\right) \tag{23}$$

where Φ^i_{out} represents the probability of receiving a symbol originally sent as part of CP *i*, out of the detection area A^i assigned to that CP. Note that the estimated pre-FEC BER can be computed based on Φ^i_{out} for all the CPs as equation (24).

$$\text{pre-FEC BER} \sim \frac{1}{m \cdot \log_2(m)} \sum_{i=1}^{m} \Phi^i_{\text{out}} \tag{24}$$

In equation (24), the average Φ_{out} probability is interpreted as an estimation of the symbol error rate (SER) and the pre-FEC BER is derived assuming that 1 symbol error causes only 1 bit with error (which is reasonable under Gray coding).

For illustrative purposes, Figure 4.3 shows the building blocks for the QoT estimation of the lightpath represented in Figure 4.1 that traverses two optical spans, each of 80 km. In Figure 4.3a, the FLANN end-to-end model predicts the features of the signal at the receiver in Site Z after the signal propagates on λ_p along *d* km. The constellation reconstruction block estimates the features of the non-propagated CPs and the QoT estimation block predicts the pre-FEC BER. In Figure 4.3b, two span DNN models trained to signal propagation on λ_p along l_1 and l_2 km are concatenated to model the end-to-end lightpath. All the blocks except the models themselves are common for both approaches.

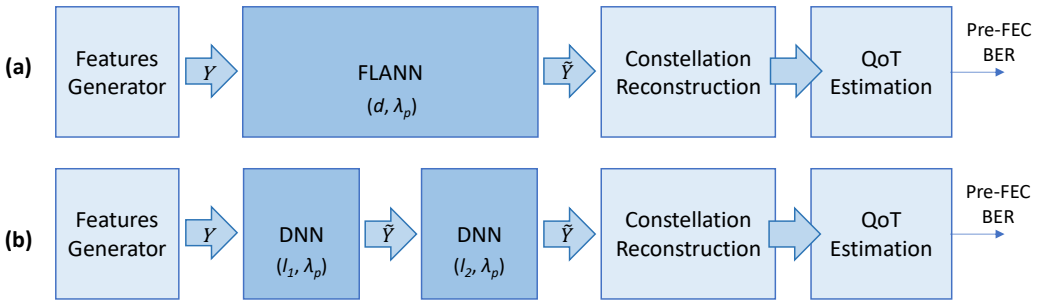


Figure 4.3. End-to-end (a) vs per-Optical Span (b) modelling and QoT estimation.

Dissemination Level	PUB (Public)

B) End-to-end Lightpath modeling using FLANNs

The FLANN feature model structure is presented in Figure 4.4. The complete FLANN model for end-to-end lightpath modelling consists of one feature model for each of the considered features $\{(\mu^I, \mu^Q, \sigma^I, \sigma^Q, \sigma^{IQ})^i, i \in selCP\}$.

Each feature model consists of a functional expansion block, where each input feature y is extended to capture its non-linear behavior. In particular, we assume that the functional expansion is based on trigonometric functions $\phi_{(\cdot)}(y)$, except $\phi_1(y)=y$. Next, each output of the expansion block is multiplied by a weight $w_{(\cdot)}$. Because no activation function is used, the estimated value of the feature is given by equation (25), where O is the order of the FLANN model:

$$\tilde{y} = bias + \sum_{k=1}^{2O+1} w_k * \varphi_k(y) \tag{25}$$

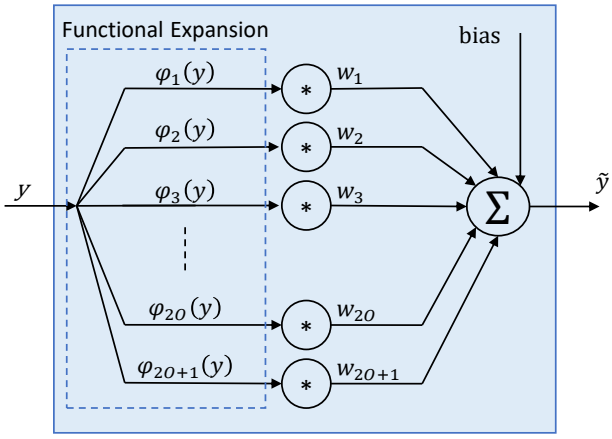


Figure 4.4. FLANN feature model structure for end-to-end modeling.

The training of the model targets at finding the weights that minimize the error between the predicted value and the value from the training dataset. FLANN lightpath models can be trained with high precision for each channel and a specific total distance d, and stored in a model database ready to be used, e.g., during lightpath provisioning.

However, even though FLANN lightpath models can be trained for every channel in the C+L+S bands, they cannot be trained for every arbitrary total distance d. A strategy is to pre-train models for incremental distances in steps of the most common span length, e.g., 80 kms. However, a real lightpath can traverse different span lengths, e.g., 65 and 75 km, so the models can present imperfections. In addition, variation over time of the behavior of optical devices, e.g., the noise figure of the OAs, new splices in the fiber, etc. can also impact the accuracy of the

models. For this very reason, the models should be tuned periodically, using the monitoring measurements collected from the TPs, once the lightpath is established.

C) DNN-based Concatenated Model

In this case, span DNNs are specifically trained for every channel and optical span configuration. This reduces the number of models that need to be pre-trained, as compared with the FLANN end-to-end lightpath modeling. In addition, NLI noise depends not only on the length of the span, but also on the accumulated distance that the signal has already traversed at the starting of the optical span on the route of the lightpath. In consequence, span DNNs require the accumulated distance *z* as additional input.

The lightpath model is produced by selecting the most appropriate span DNN models from the model database and concatenate them in the order specified by the route of the lightpath. In addition, the traversed distance needs to be also propagated and connected to the related input of the span DNNs. Figure 4.5 represents the concatenated model for the lightpath represented in Figure 4.1.

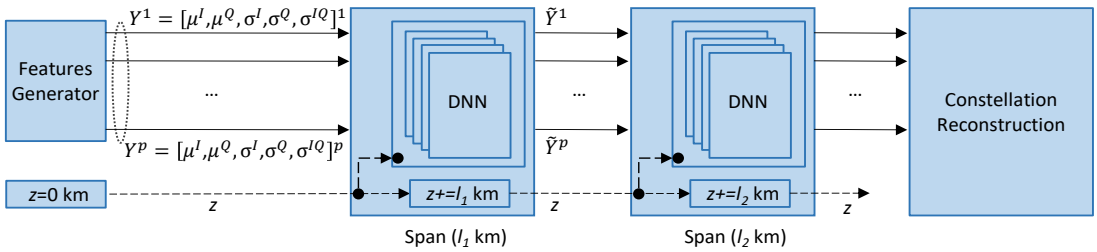


Figure 4.5: Concatenated DNN model.

As in the case of the FLANN models, the concatenated DNN model can present imperfections because of the accumulated error and variation of the behavior of optical devices over time. For this very reason, the concatenated DNN models need to be tuned periodically from the monitoring measurements collected from the TPs.

4.1.3 Results

In this section, we first describe the MB optical system under simulation and modelling. Next, we evaluate the performance of the RK4IP method with the adaptive step size w.r.t. the SFFM method in terms of both accuracy and computation time. The selected simulation method is used to generate a large dataset to be used for ML modelling MB optical propagation and ultimately, estimating the QoT. The evaluation is carried out on three different channels in each of the bands.

A) MB Optical System

We assume that the optical transponders generate 16QAM@32GBd signals shaped by a root-raised cosine filter with a 0.06 roll-off factor. For the C+L+S system, we consider 337 optical channels, with 50 GHz channel spacing for full spectrum usage. Table 4-1 summarizes the configurations used in the MB transmission simulator.

Pseudorandom binary sequences are used as input of every channel. The signal is propagated through spans of standard single mode fiber (SSMF), ranging 70-100km, with a launch power of 0 dBm. Gain flattening filters are used at the end of each fiber span to compensate for the power tilt induced by the ISRS effect. The attenuation factor, chromatic dispersion and non-linear coefficient (γ) vary with frequency. The fractional contribution of the delayed Raman response is 0.245.

Propagation in optical spans is modeled by solving the NLSE. We consider: τ_1=12.2·10^{-3} [ps], τ_1=32·10^{-3} [ps], τ_b=96·10^{-3} [ps], f_a=0.75, f_b=0.21, f_c=0.04, and f_R=0.18. EDFAs and TDFAs are modeled as ideal OAs characterized by a single gain and noise figures. The adaptive step size algorithm defined in Section 4.1.1. E) to reduce the computation time is run with δ_G = 10^{-4}, η = 5, and $MAX_\Delta z$ = 10m. Finally, at the Rx, a DSP block performs ideal chromatic dispersion compensation and phase recovery.

Table 4-1. Considered MB optical systems.

	S Band	C Band	L Band
Wavelength range[nm]	1486.5 - 1530.0	1530.4 - 1568.4	1568.8-1621.5
Bandwidth [THz]	5.7	4.75	6.2
Num of Channels (Ids)	116 (1-116)	96 (117-212)	125 (213-337)
Type of Amplifier	TDFA	EDFA	EDFA
Amplifier Noise Figure [dB]	6.5	5	5
Non-linear coefficient range [(W m)$^{-1}$]	1.31 - 1.41	1.24 - 1.31	1.13 - 1.23
Dispersion range [ps/nm/km]	12.55 - 15.58	15.60 - 18.02	18.1 - 21.69
Attenuation range (dB/km)	0.19 - 0.21	0.18 - 0.19	0.18 - 0.20

B) MB Optical System Simulation

The MB system described in the previous section was simulated using the MATLAB-based simulator of a coherent WDM system. The simulator was executed on an Intel Core i7-6700 processor with 16 GB RAM and running Windows 11 64-bit. For this very reason, pseudorandom sequences were limited to 2^{14} bits length.

We compared the traditional SSFM method with the RK4IP method enhanced with the adaptive step size algorithm (see Section 4.1.1). In addition, a fixed (2 m) step size approach is also

considered for comparison purposes. Specifically, four RK4IP methods have been evaluated: 1) RK4IP with adaptive step size in the time domain (A-RK_t); 2) RK4IP with adaptive step size in the frequency domain (A-RK_ω); 3) RK4IP with fixed step size in the time domain (F-RK_t); and 4) RK4IP with fixed step size in the frequency domain (F-RK_ω). For the evaluation, we considered one single optical span of 80 km and all the channels with launch power of 0 dBm, i.e., a total launch power of 25.27 dBm. In addition, the SSFM method was executed with a fixed step size of 0.01 m.

The comparison is carried out by comparing the ISRS gain $S(\lambda_i)$ computed by the RK4IP methods and SSFM for every channel, where $S(\lambda_i)$ is defined as the ratio between the channel power after fiber propagation and its value without the SRS, i.e., considering f_R=0 (equation (26)). This approach allows to specifically analyze the feasibility of using the RK4IP methods for MB optical transmission. Comparison of the ISRS gain per channel is averaged using the Relative Squared Error (RSE) defined in equation (27).

$$S(\lambda_i) = \frac{P(\lambda_i)}{P_{\text{no ISRS}}(\lambda_i)} \tag{26}$$

$$RSE = \frac{1}{N}\sum_{i=1}^{N} \frac{|S_{\text{RK4IP}}(\lambda_i) - S_{\text{SSFM}}(\lambda_i)|^2}{|S_{\text{SSFM}}(\lambda_i)|^2} \tag{27}$$

Figure 4.6 plots the RSE, where we observe that all the considered RK4IP methods provide solutions leading to very small error compared to the SSFM method, which make them a valid approach for simulation of MB optical transmission.

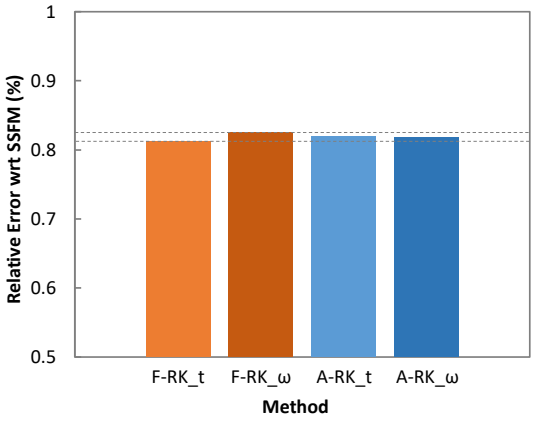


Figure 4.6. Relative error of RK4IP methods with respect to SSFM.

Figure 4.7 shows the ISRS gain profile after fiber propagation by executing the A-RK_ω method for launch power of -4, -2, and 0 dBm per-channel, i.e., 21.27, 23.27, and 25.27 dBm of total launch power, respectively. A decrease in the ISRS gain tilt from around 7 to just 3 dB is observed

as the total input power decreases. Note that even for launch powers as low as -4 dBm per-channel, the impact of ISRS is still noticeable.

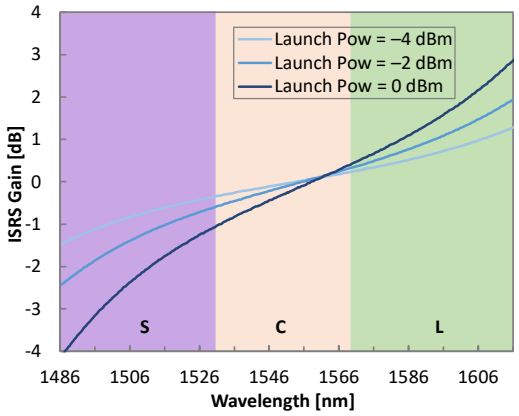


Figure 4.7 ISRS gain observed for different channel launch powers.

Figure 4.8a plots the adaptation of the step size, as provided by Algorithm 4-1, and Figure 4.8b plots the error, computed as Equation (21), with the propagated distance. We observe that the adaptive step size algorithm provides good accuracy by fixing small step sizes during the initial propagation and then relaxing the step size when the propagated distance increases.

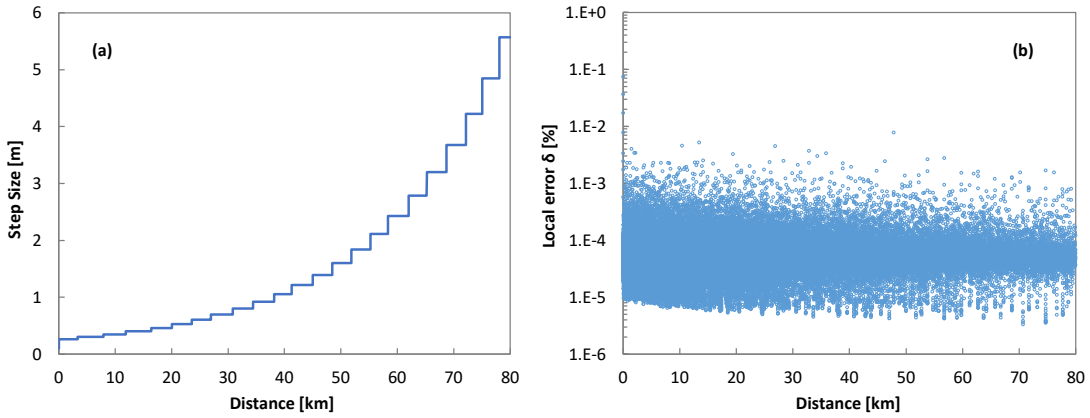


Figure 4.8 Evolution of the step size (a) and error (b) with propagation distance.

The adaptation of the step size is responsible for a reduction in the number of FFT operations that are needed. Figure 4.9a reports the number of FFT operations executed during the execution of the different RK4IP methods. We observe that the number of FFT operations heavily decreases when the adaptive step size algorithm is used. This translates into a reduction of the total computation time as observed in Figure 4.9b. In particular, the reduction in computation time enabled by the adaptive step size algorithm is over 90% (from over 3 hours to just 15 min). Interestingly, running the RK4IP methods in the frequency domain reduced the

computation time in over 40% even though the number of FFTs operations is slightly higher as compared to the time domain. This is a consequence of the higher complexity of the related FFT operations.

In conclusion, the RK4IP with adaptive step size in the frequency domain (A-RK_ω) provides good accuracy when compared to the SSFM method, while requiring much smaller computation time. Consequently, we used this method to generate the large dataset required for ML training.

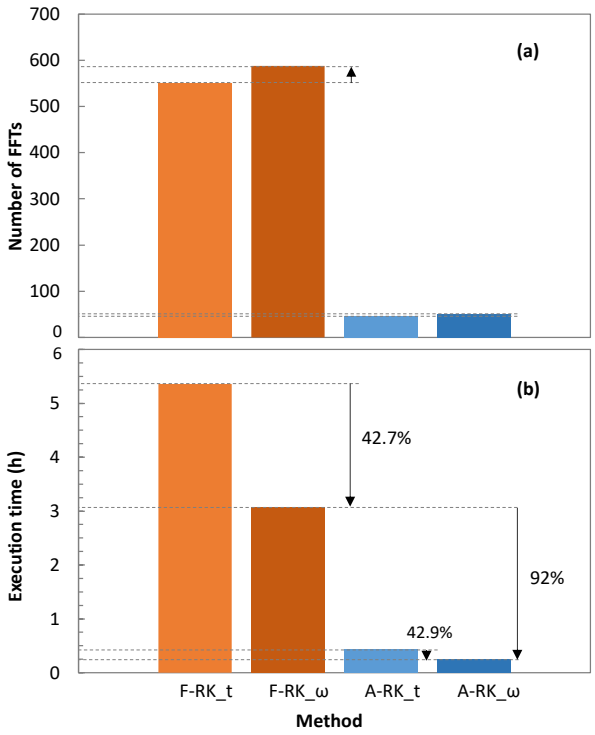


Figure 4.9. Number of FFTs required (a) and total computation time of RK4IP methods (b).

C) MB Optical Propagation Models

A dataset with 2^{16} 16-QAM symbols for each of the 337 channels in the optical MB system and for different lightpath distances ranging from 1-10 optical spans of 80 km, i.e., from 80 to 800 km, was generated using the A-RK_ω method, as proposed in the previous subsection. Armed with such dataset, FLANN models for end-to-end lightpaths and DNN models for optical spans were trained, validated and tested.

FLANN models have: *i*) 20 inputs for the features of the selected CPs; *ii*) order O=2; *iii*) 20 outputs for the propagated features of the lightpath; and *iv*) they were trained using the Least mean squares (LMS) optimizer with learning rate 8×10^{-3} and mean squared error as loss function for up to 1000 epochs with batch size of 64.

The DNN link models have: *i*) 21 inputs, i.e., one additional input for accumulated distance; *ii*) two hidden layers with 12 neurons and *tanh*(·) activation function; *iii*) 20 outputs for the propagated features for the optical span; and *iv*) they were trained with the Adam as optimization algorithm with learning rate 10^{-3} and mean squared error as loss function for up to 1000 epochs with batch size of 64.

To simplify the evaluation, we concentrate on analyzing the performance for one single channel per optical band, specifically channels 1, 167, and 337 in the S, C, and L bands, respectively.

Figure 4.10 plots the evolution of features μ^I, μ^Q, σ^I, σ^Q, σ^{IQ}, characterizing the bivariate Gaussian distribution of the exterior CP [–3+3i], with the propagation distance in terms of optical spans, as obtained from the simulator and predicted by the ML models. In general, the value of the features changes almost linearly with the distance, which is the reason behind the simplicity of the trained ML models, i.e., order 2 for the FLANNs and two hidden layers with 12 neurons for the DNNs. With these very simple structures, the predicted features are very closed to the ones from simulation. Figure 4.11 shows the maximum absolute error between the features predicted by the ML models w.r.t. the ones from simulation for CP [–3+3i]. In all cases, the error is smaller than 5%.

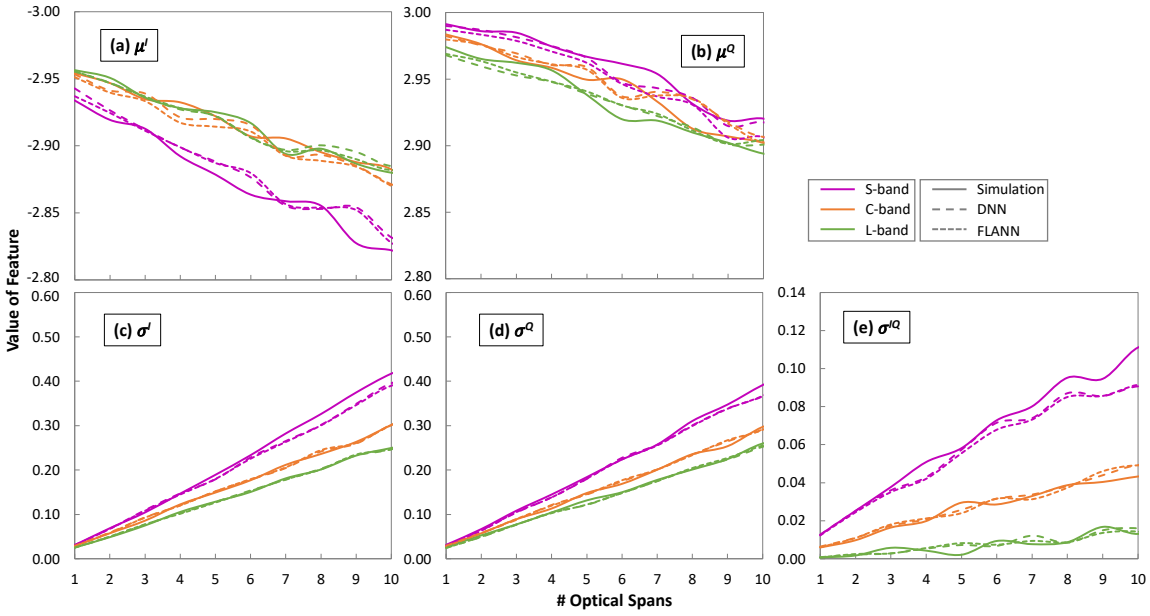


Figure 4.10. Evolution of value of the features with the number of optical spans.

Dissemination Level	PUB (Public)

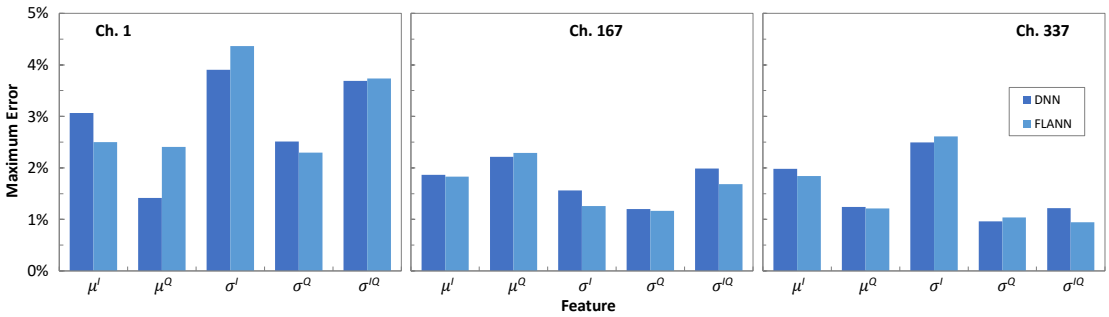


Figure 4.11. Maximum features error of ML models.

Among the different impacts on the shape of the CP caused by the evolution of the features observed in Figure 4.10, the increase of the CP variance is considered the most important one as the increased variance can make some of the received symbols to fall out of the detection area assigned to that CP, which increases the pre-FEC BER. Finally, we observe very different trends for the channels under analysis in the different bands, which anticipates also different optical performance of the associated lightpaths in terms of QoT.

D) QoT Estimation

Before analyzing the accuracy of the QoT estimation, let us observe the shape of two CPs, the interior [1–*i*] and exterior [–3+3*i*], for the three channels under analysis. Figure 4.12 reproduces the contours representing the different levels of the bi-variate Gaussian distributions that characterize the CPs after 3 optical spans. The contours correspond to levels 0.1 and 0.01 of the variances of the respective Gaussian distributions. We observe strong similarities between both distributions for the three different bands, which validates the proposed modelling approaches. It is interesting to observe the changes on the shape of the constellations for the different bands, which is more noticeable in the exterior CP. In particular, we observe that the contours clearly exceed the detection area for that CP in the case of ch. 1, which is a clear indication of high pre-FEC BER.

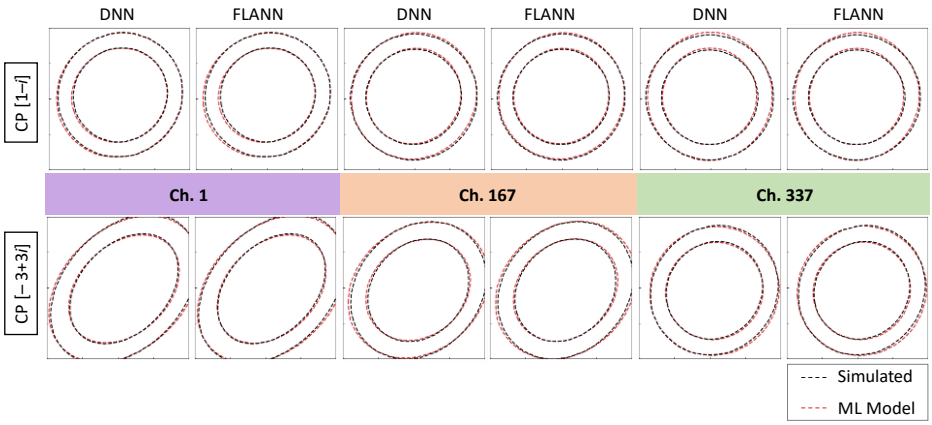


Figure 4.12. Contours of two CPs from the simulator and ML models.

With the above in mind, let us evaluate the accuracy of the QoT estimation for the three channels under analysis. We assume a pre-FEC BER threshold equal to 10^{-2} in all the cases. QoT estimation is carried out with the features of all the CPs in the IQ constellation, i.e., after constellation reconstruction.

Figure 4.13 plots the pre-FEC BER as a function of the span count for the three channels under analysis. The results show that the models can closely predict the actual QoT for all the channels. We observe a really high accuracy of both ML models for QoT estimation. In particular, both models can be used to anticipate whether a candidate route and wavelength assignment will meet the required QoT. Note the importance of wavelength selection for the different channels under analysis, where a lightpath assigned with channel 1 is only able to provide pre-FEC BER values under the threshold for 3 or less spans, while channels 167 and 337 can support one more span.

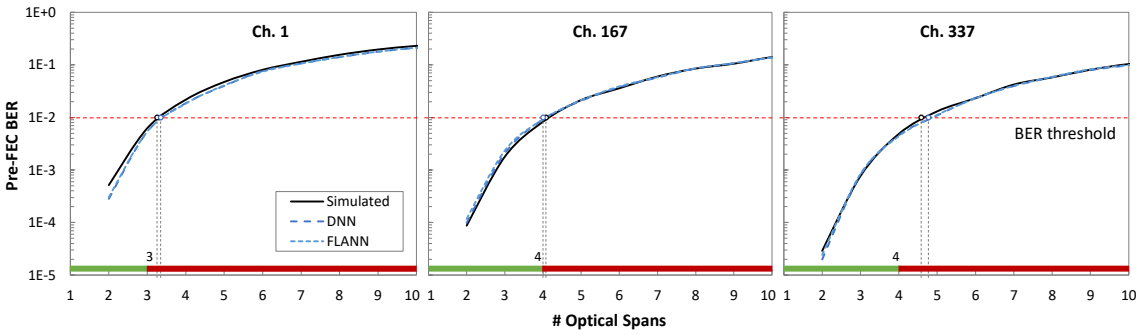


Figure 4.13. Estimated Pre-FEC BER.

Dissemination Level	PUB (Public)

4.1.4 Concluding Remarks

The foreseen introduction of MB optical transmission in transport networks requires the availability of accurate and low complex tools for network planning and operation. In particular, the introduction of digital twining solutions can provide the needed support for network automation. In this section, we have first analyzed the feasibility of using SSFM solving the NLSE modelling optical signal propagation. We have observed that the large number of channels in a fully loaded MB system and the need to consider the ISRS effect discourage using that solution in general, even for generating offline simulation data. In view of that, the RK4IP method has been evaluated as an alternative solution to reduce the computation time. The RK4IP method can be solved in the time or in the frequency domain, so both formulations have been considered. In addition, an adaptive step size algorithm to further reduce the computation time has been proposed.

For evaluation purposes, four RK4IP methods have been compared, considering adaptive and fixed step size in the time and frequency domains. The obtained results have shown that all four methods provide relative error well under 1% w.r.t. the SSFM method, being the adaptive step size RK4IP method solved in the frequency domain (A-RK_ω) the one that provides the smallest computation time as a result of the lowest number of FFT operations needed. Therefore, we selected the A-RK_ω method for simulation of MB optical transmission systems with several configurations. Such dataset can be used for modelling MB systems using ML, so applications like estimation of the QoT of unestablished lightpaths can be made available offering high accuracy with very low complexity.

In this section, we target modelling the signal propagation in the time domain, so the models can be integrated into the OCATA optical time domain DT. Two modelling approaches have been proposed and evaluated. The first approach targets modelling the propagation of the optical signal along an end-to-end lightpath, while the second one targets at modelling single optical spans, so models of end-to-end lightpaths can be easily created by concatenating spans models as defined by the route of the lightpath. In both cases, different models are pretrained using the dataset generated by simulation thus, producing a model dataset where each model is characterized by the length of the lightpath or the optical span and the used channel. Two ML techniques have been investigated: *i*) FLANNs, which because of their simplicity are used for modelling end-to-end lightpaths; and *ii*) DNNs, which are proposed for modelling single optical lightpaths. In both cases, the models are integrated in OCATA, so tools for features generation, constellation reconstruction, and QoT estimation have been used.

The ML modelling approaches have been evaluated in terms of their accuracy for modelling features propagation, as well as in terms of the accuracy of QoT estimation, specifically the pre-FEC BER. The results showed that the accuracy of the approaches is very similar and very high in general, with error well below 5% w.r.t. to the data from simulation. Regarding pre-FEC BER

estimation, the results showed not only its accuracy to estimate the pre-FEC BER value, but also its application for candidate route and wavelength assignment validation, by comparing such estimation to the BER threshold.

In conclusion, both approaches have been validated to be integrated in the OCATA DT. Even though the optical span modelling seems more flexible as it reduces the number of models to be pre-trained, the availability of both approaches provides a method for checking deviations in the models and adds robustness to OCATA.

4.2 NetDevOps for Network Control and Automation

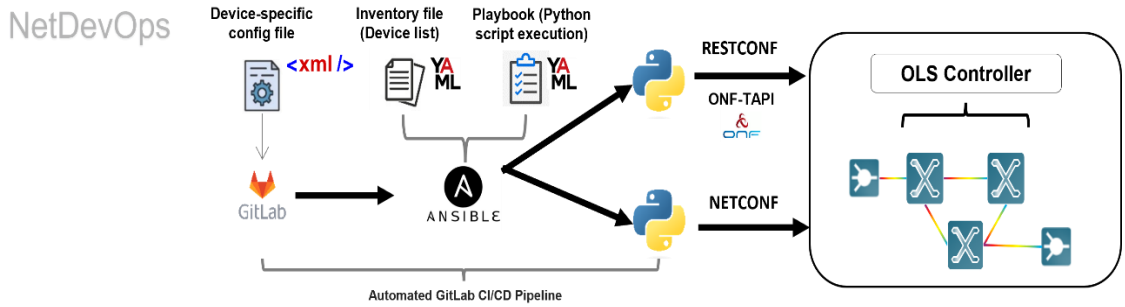


Figure 4.14: NetDevOps pipeline Workflow for device specific configuration and service operations.

The NetDevOps pipeline as shown in Figure 4.14 introduces a streamlined, automated approach to managing network configurations and services, aligning network operations with DevOps best practices. It combines version control, test-driven validation, and programmable deployment workflows to support consistent, scalable, and low-risk changes across diverse network infrastructures—including multi-vendor optical domains.

Through integration with controller APIs and support for standard interfaces like TAPI, the pipeline enables end-to-end service automation and cross-domain orchestration. Features such as digital twin validation, rollback mechanisms, and centralized state tracking ensure operational reliability and traceability.

CI/CD Workflow and Automation Stack

The pipeline leverages a Git-based CI/CD (continuous Integration and Continuous deployment) model to drive automation workflows. Any configuration change or network event—once committed to the repository, automatically initiates the pipeline, which performs the following sequential stages:

- File differencing to isolate modified configurations
- Static validation (syntax, schema, and policy compliance)

- Environment verification, including device/controller reachability
- Deployment using Ansible playbooks and Python automation scripts

This design enables deterministic deployments, versioning via Git commit history, and minimal manual intervention, reducing human error and promoting infrastructure-as-code practices.

Configuration Management and Rollback

Each network element is associated with a discrete configuration artifact. This facilitates device-specific configuration management across heterogeneous vendor ecosystems. Leveraging Git, the system supports automated rollback, allowing reversion to previously known-good states in the event of failure, thereby improving operational stability and fault recovery efficiency.

API Enablement and Optical Controller Integration

To extend NetDevOps workflows into optical transport domains, a dedicated API layer has been designed and deployed. This API serves as an abstraction interface to optical SDN controllers and supports standardized orchestration through TAPI (Transport API).

The API enables (as shown in Figure 4.15):

- Service creation
- Service deletion
- Rollback and recovery logic, tightly coupled with configuration versioning and operational state monitoring

Configuration and Change Management:

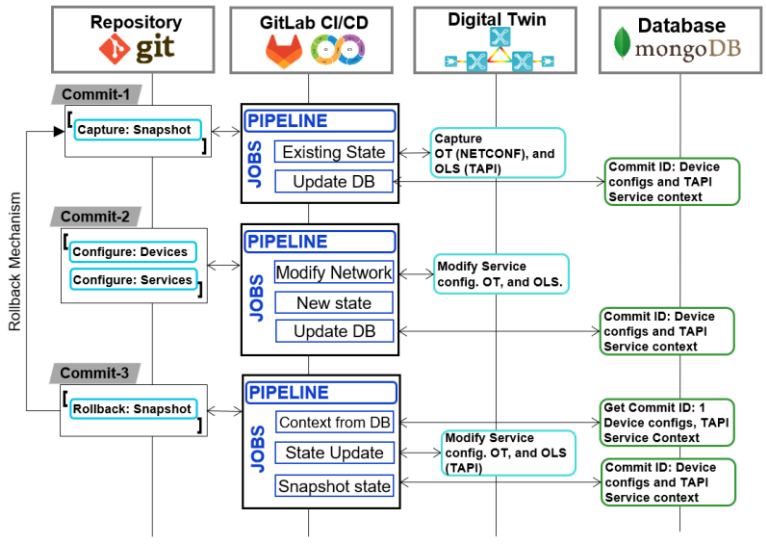


Figure 4.15: Sequence Flow Diagram of Commit and Rollback Mechanisms in NetDevOps.

Dissemination Level	PUB (Public)

Each network device maintains a dedicated configuration file within a Git repository, ensuring version control and easy tracking of modifications. Configuration files are managed in hierarchical formats (e.g., XML, YAML, JSON) and include device-specific parameters. For service provisioning, JSON based files define necessary service parameters, enabling streamlined creation or deletion of services through REST requests complied to ONF-TAPI. Network engineers modify these files as needed, committing updates to the repository. This commit triggers an automated pipeline within GitLab ensuring that changes are processed in a controlled environment.

Automated Pipeline Execution and Deployment:

Upon committing changes, the GitLab CI/CD pipeline initiates a multi-stage process, enhancing workflow efficiency and error handling. Stages include, see Figure 4.16.



Figure 4.16: NetDevOps CI/CD pipeline stages including testing, device configuration and service provisioning.

1. Build Stage: Prepares the necessary environment, either through a virtual environment setup or by pulling a pre-configured Docker image.
2. Change Detection: An Ansible playbook identifies modified files and matches them to specific network devices, allowing targeted configuration updates.
3. Testing Stage: Checks device connectivity and verifies authentication using NETCONF or gNMI protocols.
4. Device Configuration: This stage deploys per-device configuration across network nodes using Ansible playbooks and Python NETCONF client (ncclient).
5. Service Provisioning: This stage automates service creation and deletion via TAPI and RESTCONF APIs, enabling seamless end-to-end service deployment across the optical network.

Security, Rollback and Compliance:

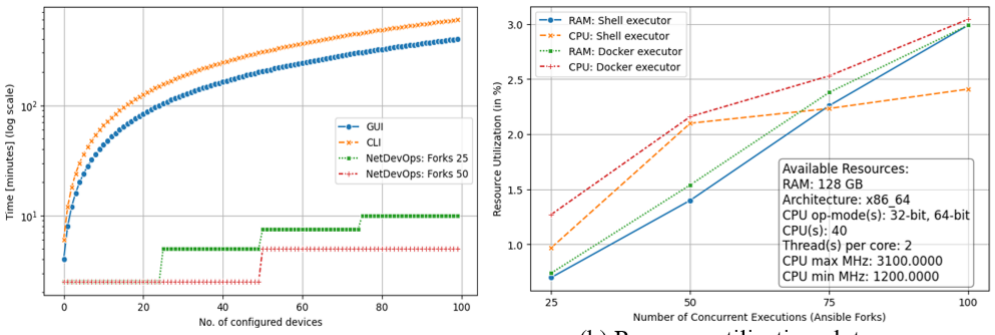
The proposed NetDevOps pipeline automates and validates network changes through a structured workflow. The process begins with committing changes to the repository. For example, in Commit-1 (as shown Figure 3.12), the service configuration (S-1) and device specific configuration files are modified and committed. Upon commit, the pipeline validates the working environment and identifies the affected devices by mapping configuration file name to the corresponding hosts in the global inventory. Next, the pipeline invokes an ansible playbook

that triggers a python script to apply device specific configurations via NETCONF. Simultaneously, service provisioning is initiated through a RESTCONF request to the TAPI interface. This establishes Service 1 (S-1) through the optical line system (OLS), routing traffic via ROADMs R-2 to R-3. In scenarios such as network failures or service upgrades requiring a path change (e.g., routing through R-2 → R-1 → R-3), a delete request is issued for S-1. Subsequently, new service configurations (S-2) are created, and the corresponding per-device configuration files are updated to reflect the new path. To validate the rollback mechanism, a rollback file is edited to reference Commit-1, triggering the pipeline to retrieve and reapply configurations from Commit 1. Updated service data and configurations are stored in a MongoDB database after each pipeline execution, maintaining a complete history for efficient future rollbacks and audits.

Sensitive information, such as credentials, is securely stored in Ansible Vault to prevent unauthorized access. Network communications use secure protocols to protect data integrity. The pipeline includes error handling and rollback mechanisms, reverting to the last stable configuration via Git commit hash ensuring stability. Compliance with industry standards enables interoperability and seamless integration with network controllers.

Execution Performance and Testing

The pipeline achieves an end-to-end execution time of under 10 minutes, from commit or event detection to successful configuration deployment and validation. This represents a significant reduction from traditional workflows (~30 minutes) that relied on manual CLI or GUI methods as shown in Figure 4.17. It is scaled accordingly depending on the number of devices.



(a) Time efficiency observed (b) Resource utilization plot

Figure 4.17: NetDevOps Performance Evaluation (a) Time efficiency comparison (b) Resource utilization.

To ensure deployment safety, all configuration changes undergo digital twin testing in a virtualized emulated environment. This pre-deployment validation ensures policy compliance, functional correctness, and compatibility with the production environment.

All deployment actions, configuration states, and validation results are persistently stored in a centralized MongoDB database, enabling comprehensive audit trails, state introspection, and post-mortem analysis.

Scalability and Parallelism

The architecture supports parallel execution of deployment jobs, allowing simultaneous updates across large-scale, geographically distributed devices. This enables high-throughput, zero-touch configuration management at scale.

4.3 Distributed Intelligence and MAS Control

The combination of highly dynamic network services requiring stringent Quality of Service (QoS), especially in terms of end-to-end (e2e) delay, together with capital and operational cost reduction cannot be faced using centralized Software-Defined Networking (SDN) solutions only. In particular, such expected dynamicity requires autonomous near-real-time operation fed with pervasive telemetry to make per-service decisions that ensure the committed QoS, while reducing overprovisioning as much as possible. In this section, we propose a distributed control architecture based on Multi-Agent Systems (MAS) to assist the SDN controller in the control of network services near-real-time. Per-traffic flow telemetry data is collected from the packet nodes, distributed through the agents in the control plane, and analyzed to assure performance and to anticipate any degradation. Measurements feed flow agents, which are based on Deep Reinforcement Learning (DRL) models, to make routing decisions aiming at ensuring flow performance. In case that QoS degradation is detected, we propose algorithms to analyze its cause, which can be as a result of some bottleneck in the network. We show how the latter is detected and additional capacity is requested to the SDN controller, which in turn creates an optical bypass to provide additional capacity. The proposed solution is demonstrated experimentally on a federated test bed connecting UPC and CNIT premises. Focused first on the control plane, the feasibility of the proposed architecture and workflows is experimentally assessed. After that, the performance of the near-real-time operation is evaluated at the data plane to verify that the maximum e2e delay is not exceeded for multiple flows, showing the effectiveness of predictive QoS evaluation together with infrastructure and service reconfiguration.

4.3.1 Autonomous Flow Routing Operation

The proposed autonomous flow routing architecture is presented in Figure 4.18a. At the control plane, an SDN controller is in charge of the network; specifically, during flow provisioning, the SDN controller computes a set of allowable routes P_i for every flow f_i, i.e., those that can potentially meet the required QoS in terms of predictable delay components only. We assume that every individual flow $f_i \in F$ has a particular maximum e2e delay (d^{max}_i) that needs to be

satisfied. Figure 4.18a shows two packet switches that are ingress (R1) and egress (R5) of a set of flows $F_{1\text{-}5}$. Static routing policies may not provide the needed flexibility under dynamic traffic conditions, so its application could result in delays above the maximum for the flow in the case of congestion. In view of that, once each flow is established, the SDN controller hands over its near-real-time control to a *multi-flow agent*, running on top of ingress node R1. The multi-flow agent implements *flow agents* that are deployed by the MLFO, where a specific *multi-flow MAS supervisor* oversees their performance and can tune their parameters and models. Each individual *flow agent* makes dynamic decisions for the flow under control on the amount of flow traffic to be sent through each of the output interfaces related to the allowable routes (*routing policies*). Flow agents are continuously fed with per-flow traffic measurements collected by the local *node agent* and with telemetry data provided by a *telemetry agent* deployed at the egress packet node R5, which measures (per-flow and aggregated) e2e delay and jitter for each of the allowed routes.

Armed with such telemetry data, intelligent and accurate actions can be performed to meet complex operational objectives like ensuring per-flow maximum e2e delay with overall minimum routing cost. For illustrative purposes, Figure 4.18b shows a small 5-node network, a set of flows $F_{1\text{-}5}$ with common ingress and egress nodes R1 and R5 (in general, R1 and R5 do not need to be in the border of the network), as well as other flows (R1-R3 and R2-R5) on the network (*background traffic*). In the example, for every f_i, the SDN controller computes two routes through the packet network: *i*) route R1-R2-R5 (*p1*); and *ii*) R1-R3-R4-R5 (*p2*), which can be used at any time. The SDN controller assigns different costs to each route (e.g., as a function of the number of hops), so as to give priority to traffic routing.

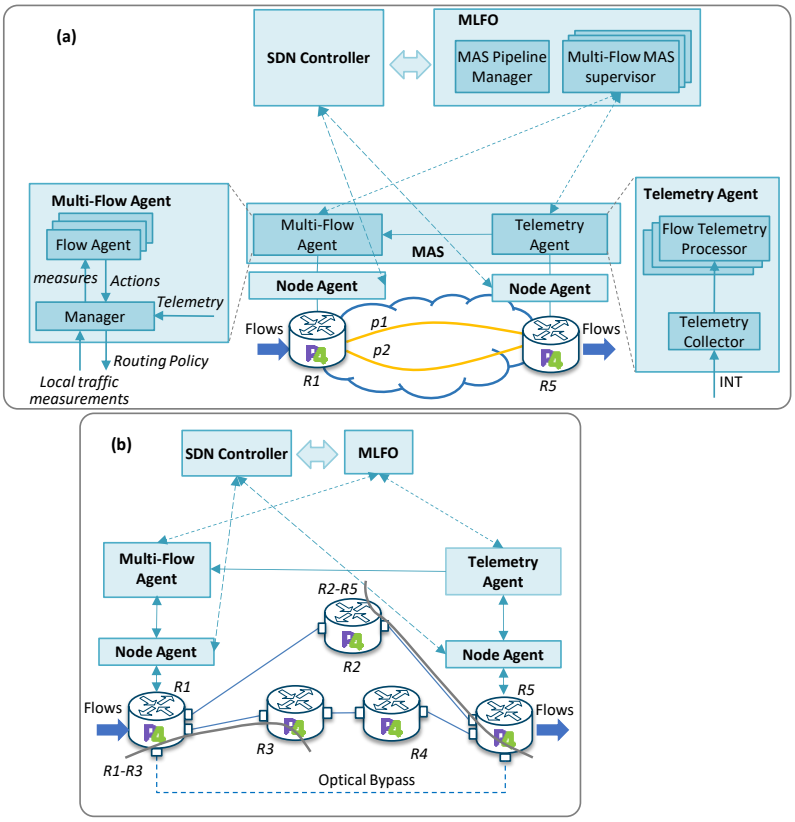


Figure 4.18: Proposed architecture for autonomous multi-flow routing (a) and illustrative scenario (b).

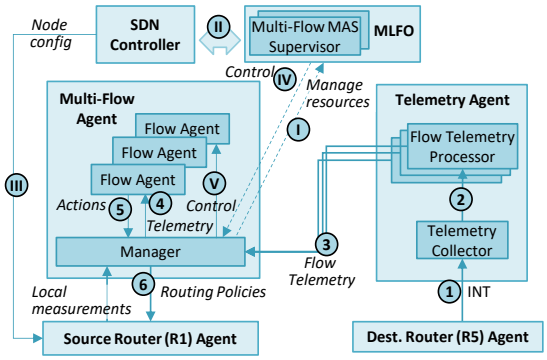


Figure 4.19: Flow control and network resource management.

Figure 4.19 details the workflow of autonomous multi-flow routing operation, where every flow agent defines routing policies independently. The destination node R5 collects INT data and reports those records to the collector in the telemetry agent (step 1 in Figure 4.19). Then, per-flow telemetry data is managed by individual processors (step 2) and sent to the multi-flow agent manager in the ingress (step 3). The manager sends telemetry data, local traffic measurements and some other parameters to the specific flow agent (step 4). Each flow agent computes the routing policies to be enforced next for traffic flow f_i (step 5) and, the manager

eventually forwards the policies to the node agent that translates them into actions in the packet node (e.g., routing table updates) (step 6).

In our implementation, the flow agents are based on DRL, so routing policies are decided by DRL models. At the flow provisioning time, the multi-flow MAS supervisor selects the most appropriate model for operation from a pre-trained models data base (DB). Models in the DB are pre-trained in a sandbox domain and classified by their characteristics, such as d^{max} and number of routes available. Because the flow agents are network topology agnostic, pre-training can be performed on a sample network to obtain a general model that operates within the specified characteristics. A complete DB can then be built by considering various d^{max} requirements and total routes. In consequence, the multi-flow MAS supervisor needs to know d^{max} and the number of routes available for each flow to select the model that best fit them and send it to the flow agent to immediately start flow operation.

Once in operation, the flow agent selects the most appropriate action and measures the resulting QoS. Note that the QoS of each flow f_i depends on its input traffic and routing policies, as well as on that of other flows in the network that use common network resources, e.g., flows R1-R3 and R2-R5 in Figure 4.18b. Therefore, even though flow agents try to balance the traffic through the allowable routes to meet d^{max}, it might happen that delay constraints cannot be meet for all the flows during some time periods because of congestion in some intermediate packet node. For that reason, the connectivity of R1 and R5 can be increased by adding additional connectivity that increase the current capacity. In particular, we can consider a multi-layer scenario so, in case of congestion, R1 and R5 can be connected using an *optical bypass*, i.e., a (virtual) link in the packet network that is supported by an optical connection, which can be activated on-demand.

In the architecture in Figure 4.19, the manager provides periodical updates on the performance of the flow agents in meeting the QoS requirements of each flow (step I in Figure 4.19). In case that difficulties are predicted, the multi-flow MAS supervisor in the MLFO requests the SDN controller to add more resources, e.g., increase the capacity of the existing routes or add additional ones (II). The SDN controller manages the resources in the network and, depending on the capacity that is requested, it can decide to establish and activate an optical bypass connecting R1 and R5 (III). Once the optical bypass is available, the multi-flow MAS supervisor advertises its activation. In addition, the flow agent may require new models to properly operate with a different number of routes, so the MAS supervisor provides pre-trained models for the flow agents to operate (IV). The manager in the multi-flow agent distributes the models to the related agents (V).

However, network reconfiguration, particularly the activation of an optical bypass, takes time, e.g., 1 min, and this time might result in a QoS degradation of the flows. Another issue is that, once the bypass is activated, it could be used by other flows in $F_{1\text{-}5}$ without any additional cost, which might result in excessive delay if large amount of traffic uses the bypass. For this reason,

if the objective is to allow the use of the bypass for flow routing only when it is strictly needed to meet the flow QoS requirements, then the use of the bypass needs to be somehow penalized, so the SDN controller assigns its use with a cost higher than that of routes *p1* and *p2.* All these together impose the need of some coordination among the flow agents. In consequence, algorithms are required at both, the multi-flow agent manager and the multi-flow MAS supervisor, which is in charge of managing the network resources allocated for the flows $F_{1\text{-}5}$. To that end, the multi-flow MAS supervisor receives collected e2e telemetry data that includes the QoS of the flows and the load of the routes together with the aggregated total traffic. After analysis, if the need for additional network resources is identified, it coordinates with the SDN controller via the MLFO and informs the flow agents about the availability of new routes.

4.3.2 Models and Algorithms

This section describes the key models and algorithms involved in the proposed autonomous multi-flow routing operation, specifically during: *i*) *flow provisioning phase*, where the SDN controller finds resources that will support the packet flow; *ii*) *near-real time operation*, managed by flow agents that ensure the committed QoS, and *iii*) *resource reconfiguration*, where the load manager operation ensures the availability of just enough network resources to guarantee the committed QoS. Table 4-2 summarizes the used notation.

Table 4-2: Notation.

Flow provisioning	
G	Graph $G(N, E)$ representing the network topology. N is the set of nodes and E is the set of links.
d^{max}	maximum delay (QoS) to guarantee for traffic flow f.
x^{max}	maximum expected traffic for traffic flow f.
p^{max}	maximum number of allowed routes for traffic flow f.
q^{min}	minimum queuing delay introduced by any route
F	set of flows between a pair of source and target (s, t) nodes. f denotes an individual flow.
P	set of routes computed between (s, t). p denotes an individual route.
Near-real time operation	
$s(t)$	state at time t
$x(t)$	amount of input traffic at time t.
k_p	allowed capacity of route p.
c_p	cost of route p.
$a_p(t)$	percentage of input traffic to be routed through route p at time t
$y_p(t)$	amount of traffic directed through route p at time t.
$r(t)$	reward at time t.
$r_{delay}(t)$	reward by the measured delay

$r_{cost}(t)$	reward by routing cost
α_{delay}, α_{cost}	Constants for reward components
β	penalty when the maximum delay is exceeded
Resource reconfiguration	
$P(f)$	set of routes for flow f.
ℓ_{fp}	maximum normalized load measured along any interface of path p of flow f.
Z	summary of current and predicted values of traffic, load, and e2e delay.

Algorithm 4-3. Route computation algorithm.

INPUT: $G=\{V,E\}$, $req=<s, t, x^{max}, d^{max}, p^{max}>$, τ
OUTPUT: P

1: $G_{aux} \leftarrow G$
2: **for each** $e \in G_{aux}$.E **do**
3: **if** e.cap $< \tau \cdot x^{max}$ **then** G_{aux}.E.pop(e)
4: $P \leftarrow k\text{-}SP(G_{aux}, s, t)$
5: **for each** $p \in P$ **do**
6: p.d$^{\min} \leftarrow transmDelay(p) + q^{min}$
7: **if** p.d$^{\min} > d^{max}$ **then** P.pop(p)
8: sort(P, <'d^{min}','*similarity*'>, ASC)
9: **return** $P[1..p^{max}]$

E) Flow Provisioning Phase

Upon the reception of a new flow provisioning request, *req*, the SDN controller runs Algorithm 4-3 to compute the set of allowable routes P for the flow. Algorithm 4-3 receives the current network state, summarized by graph G, and the flow request including ingress s and egress t nodes, maximum traffic x^{max} and delay d^{max} to be guaranteed, and maximum number of allowed routes p^{max}. In addition, parameter τ specifies the minimum percentage of x^{max} that every individual rote has to support. After some initialization (line 1 in Algorithm 4-3), the algorithm discards those links with residual capacity below $\tau \cdot x^{max}$ (lines 2-3) and uses the k-shortest path algorithm to compute the routes on the resulting graph (line 4). For each route, the minimum expected delay d^{min} is computed considering its total distance and a minimum value of queuing delay. The expected delay is used to discard those routes that cannot meet d^{max} (lines 5-7). Finally, the remaining routes are sorted by multiple criteria, like cost and capacity (line 8) and returned (line 9). The complexity of the is related to that of computing k-shortest paths, e.g., using Yen's algorithm, which is $O(k^*|N|^*(|E| + |N| \log |N|))$.

F) Near-real time operation

The flow routing agents are based on DRL and thus, include: *i*) an *DRL agent* in charge of learning the best actions to be taken based on the current routing state and the received reward; and *ii*) an *environment* that computes the state and a reward for the action taken. From the various DRL methods available in the literature, we rely on Twin Delayed Deep Deterministic Policy

Gradients (TD3) for the flow agents. TD3 is an off-policy DRL technique employing a dual set of *critic* Deep Neural Networks (DNN) alongside with an *actor* DNN, which is updated with a certain delay. We denote the ensemble of critic and actor DNNs operating within the DRL framework as the *model*, which is responsible for dynamically determining the best routing policy for the flow.

The formulation of the DRL model is as follows:

$$s(t) = x(t) \cdot \frac{\sum_{p \in P} k_p}{|P|} \tag{28}$$

$$y_p(t) = a_p(t) \cdot x(t) \tag{29}$$

$$r(t) = \alpha_{delay} \cdot r_{delay}(t) + \alpha_{cost} \cdot r_{cost}(t) \tag{30}$$

$$r_{delay}(t) = \begin{cases} -\beta - \dfrac{d(t)}{dmax}, & d(t) > dmax \\ 0, & d(t) \leq dmax \end{cases} \tag{31}$$

$$r_{cost}(t) = -x(t) \cdot \sum_{p \in P} a_p(t) \cdot \frac{c_p}{k_p} \tag{32}$$

We define the state $s(t) \in \mathbb{R}^+$ as the input traffic $x(t)$ scaled by the average capacity of the available routes P, where k_p represents the capacity of route $p \in P$ (eq. (28)). Every element in action $a(t) \in [0,1]^{|P|}$ specifies the proportion of input traffic to route through route p, and thus $a(t)$ represents the routing policy to be applied. Then, the amount of traffic directed through route p is $y_p(t)$, defined in eq. (29). The reward function $r(t)$ aims to penalize actions that lead to poor QoS or increased network cost. Consequently, the multi-objective reward function in eq. (30) considers a reward based on the measured e2e delay (r_{delay}), as well as on the routing cost (r_{cost}); constants α_{delay} and α_{cost} weight each reward component. The r_{delay} component is defined in eq. (31), where β is a predetermined penalty applied when the maximum delay is exceeded. Finally, the r_{cost} component (eq. (32)) computes the cost of the distribution of traffic on the routes, based on the cost (c_p) and the capacity of each route.

Algorithm 4-4. Telemetry Data Analysis.

INPUT: $\omega = \{< x_{fp}, \ell_{fp}, d_{fp}, a_{fp} >, \forall f \in F, p \in P(f)\}, t$
OUTPUT: Z

1: DB.update(ω, t)
2: $Z \leftarrow \emptyset$
3: **for each** $f \in F$ **do**
4: $Y \leftarrow$ select(DB, "d_f", [t-w, t])
5: $d_{cur} \leftarrow Y[w]$
6: $d_{pred} \leftarrow$ forecast(Y, t+m)
7: $x \leftarrow \emptyset$; $\ell \leftarrow \emptyset$
8: **for each** $p \in P(f)$ **do**
9: $Y \leftarrow$ select(DB, "x_{fp}", [t-w, t])
10: $x_{cur}(p) \leftarrow Y[w]$
11: $x_{pred}(p) \leftarrow$ forecast(Y, t+m)
12: $Y \leftarrow$ select(DB, "π_{fp}", [t-w, t])
13: $\ell_{cur}(p) \leftarrow Y[w]$
14: $\ell_{pred}(p) \leftarrow$ forecast(Y, t+m)
15:

16: $Z(f) \leftarrow <d_{cur}, d_{pred}, x_{cur}, x_{pred}, \ell_{cur}, \ell_{pred}>$
return Z

When the flow agents are deployed, they are configured with a pre-trained model able to tackle the uncertainty of input traffic. Then, the manager in the multi-flow agent collects measurements of the flow traffic $x(t)$ from the local node agent and e2e delay $d(t)$ from the telemetry agent and sends them to the specific flow routing agent for flow f_i (see Figure 4.19). The measurements are received by the flow agent environment that computes the current state $s(t)$ and reward $r(t)$ and sends them to the DRL agent to determine the best action $a(t)$. The action is sent to the manager who is in charge of sending the routing policies to the local node agent.

G) Resource reconfiguration

The operation of the flow agents needs to be continuously monitored and analyzed to prevent degradations in the performance of the flows. To that end, telemetry data is first analyzed at the multi-flow agent and a summary is sent to the multi-flow MAS supervisor for flow performance analysis.

Algorithm 4-4 shows the telemetry data analysis that is performed at time t, when a data sample ω is received. The sample contains the current routing policies a_{fp} that are configured in the local node, together with three features measured for every flow f and path p from the local and the telemetry agents: *i*) the traffic count x_{fp}, *ii)* the maximum load ℓ_{fp} computed as the maximum normalized load measured along any interface of the path; and *iii*) the e2e delay d_{fp}. The purpose of the telemetry data analysis is to return the summary Z that contains the current and predicted values of each of the aforementioned telemetry features. In particular, predictions aim at providing estimation of the features at time $t+m$, being m sufficiently large to include both the time needed by MAS supervisor to carry out some performance evaluation, as well as for the SDN controller to implement some resource reconfiguration (e.g., optical bypass activation) in response to multi-flow MAS supervisor notifications. The multi-flow MAS supervisor uses Z for several purposes apart of flow performance evaluation, such as model tuning and retraining.

Algorithm 4-5. Flow Performance Analysis.

INPUT: $Z, d_{high}, \ell_{low}, \ell_{high}$
OUTPUT: W

1: $W \leftarrow \emptyset$
2: **for each** $f \in F$ **do**
3: $<d_{cur}, d_{pred}, \ell_{cur}, \ell_{pred}> \leftarrow Z(f)$
4: **if** $\max(d_{cur}, d_{pred}) < d_{high}$ **then continue**
5: **if** $\min(\ell_{cur}, \ell_{pred}) < \ell_{high}$ **then**
6: $W \leftarrow W \cup \{<f, \text{“model_reconf”}>\}$
7: **else for each** $p \in P(f)$ **do**
8: $\ell' = \max(\ell_{cur}(p), \ell_{pred}(p))$

```
9:          if ℓ' > ℓ_high then
10:             Δk ← k_p·(ℓ'-ℓ_low)/ ℓ_low
11:             W ← W ∪ <f, p, "cap_inc", Δk>
12:     return W
```

Algorithm 4-4 starts updating its internal database DB with the received telemetry data sample ω (line 1). Then, after initializing the summary data Z, the algorithm performs delay forecasting per flow using the last w aggregated delay measurements in the DB and estimating the expected delay at time $t+m$ using polynomial extrapolation (lines 2-6). In particular, the i-th degree polynomial that maximizes the Akaike Information Criteria (AIC) is chosen as predictor.

A similar procedure is used to estimate traffic and load, but independently for each path of a flow (lines 7-14). Finally, current and predicted values are populated in the summary data Z and returned (lines 15-16). The complexity of the algorithm is $O(k^*|F|)$.

Algorithm 4-5 in the multi-flow MAS supervisor receives the data summary Z generated by the telemetry data analysis process at every time t. The purpose of this algorithm is to return a diagnosis set W with a list of notifications related to the flows performance. The method follows a threshold-based approach where the received current and predicted features are compared against reference thresholds for delay (d_{high}) and load (ℓ_{low}, ℓ_{high}). After initializing the diagnosis set W (line 1 in Algorithm 4-5), every flow is processed independently (lines 2-11). Delay and load features for the current flow are extracted from Z (line 3). Without loss of generality, we consider that if both current and predicted delay are under the threshold d_{high}, flow performance is good, i.e., flow agent is performing as expected (line 4). Otherwise, delay is (or will be) high and hence, some actions must be taken to mitigate such performance degradation. Two performance degradation causes can be distinguished: *i*) if the minimum load of the flow is below ℓ_{high}, we assume that the model running in the multi-flow agent is not taking advantage of unused capacity and, therefore, a model reconfiguration for that flow should be triggered (lines 5-6); and *ii*) if the load is high, additional capacity might be needed (lines 7-11). In such case, capacity increment Δk is requested for each path whose load exceeds ℓ_{high}, in order to make its load equal to ℓ_{low}. Such additional capacity is computed with respect to the allowed capacity k_p of the evaluated path. The complete diagnosis set is eventually returned (line 12), which will be processed by the MAS supervisor to trigger model reconfiguration and/or request additional capacity to the SDN controller.

4.3.3 Description of Workflows

This section describes in depth the workflows executed during: *i*) flow provisioning (WF1); *ii*) near-real time operation (WF2); and *iii*) resource reconfiguration (WF3). We assume the scenario in Figure 4.18.

A) Flow provisioning phase

Recall that the information needed for flow provisioning includes the ingress and egress nodes, as well as the maximum traffic and delay to be guaranteed. The SDN controller uses Algorithm 4-3 to compute a set of routes and selects a subset of them with enough capacity to meet the maximum traffic of the flow, and configures the routing tables in the nodes in each route, without making any resource reservation for the flow, so as to leave the freedom to operate to the MAS. Then, the MAS pipeline is deployed and configured for near-real-time operation based on the received telemetry that needs to be configured as specified by the MAS. The workflow carried out during flow provisioning is shown in Figure 4.20. Details of some selected messages for this and the rest of the workflows (identified with the labels in their respective workflow) are presented in Table 4-3.

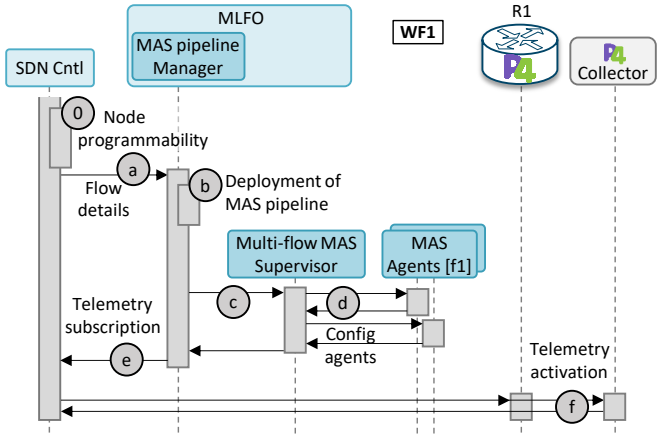


Figure 4.20: WF1: Flow provisioning.

The workflow starts after the packet nodes along the routes supporting in the flow (see Subsection 4.3.2.E)) have been properly configured (step 0 in Figure 4.20). The SDN controller sends the flow details (message *a*) to the MLFO. Among other information, message *a* includes: *i*) the QoS parameters to be guaranteed for the flow; *ii*) identification of the ingress and egress nodes and ports; *iii*) the node agents for configuration purposes; *iv*) the set of metrics and other capabilities (e.g., periodicity, aggregation, etc.) for telemetry; and *v*) the path list with details of the ports in ingress and egress nodes, including their maximum usable capacity, and their cost. Based on that information, the MAS pipeline manager in the MLFO decides the locations where agents are needed to operate the flow and deploys them together with the necessary connectivity. The designed solution allows several flows to be controlled between the same ingress and egress nodes, so the MAS pipeline includes the MAS agents, as well as the multi-flow MAS supervisor, which is only deployed if no other flows between those nodes exist (step b). Once the deployment is completed, the MAS pipeline Manager configures the multi-flow MAS Supervisor and provides the configuration received from the SDN controller (message c).

The multi-flow MAS Supervisor is now in charge of configuring the agents. The configuration includes the DRL model to run in the flow agent and the details of the telemetry that both the flow agent and the telemetry processor will receive (message d). Finally, the multi-flow MAS Supervisor sends the selected telemetry configuration to the MAS pipeline manager, which sends it back as a response to the SDN controller (message e). The details of message *e* are summarized in Table 4-3. The SDN controller uses the telemetry configuration to activate the telemetry in the nodes along the paths, as well as in the telemetry collector (message f), so the details of the MAS agent (IP address, port, etc.) where the data has to be sent are included in message *e*.

Table 4-3: Information in selected messages.

Msg Id	Information
a **(WF1)**	Create flow. Flow details: • Flow Id • QoS parameters (d^{max}, x^{max}) • Ingress (node/port, agent details) • Egress (node/port) • INT collector (telemetry capabilities) • Route list (path Id, src/dst ports, k_p, c_p)
e **(WF1)**	Telemetry configuration for ingress node and collector: • MAS agent details (IP, port, endpoint) • Metrics to collect • Specific configuration (period, aggregation, etc)
1 **(WF2)**	Traffic telemetry for every flow: • Flow Id • List of metrics and measured values during the specified period.
5 **(WF2)**	E2e telemetry for every flow: • Flow Id • List of metrics and aggregated values during the specified period disaggregated by route.
8 **(WF2)**	Routing policy for every flow: • Flow Id • a_p.
10 **(WF3)**	Required reconfiguration of resources: • List of flow Ids • Capacity and delay constraints
13 **(WF3)**	Reconfiguration of resources for every flow: • Route list (path Id, src/dst ports, k_p, c_p)

B) Near-real time operation

The workflow for near-real time operation of the traffic flows is outlined in Figure 4.21. Let us assume that flow routing policies are configured at node R1 (message 0 in Figure 4.21) to enable multi-path routing. Input flow traffic measurements are collected and sent periodically to the local multi-flow agent for every traffic flow entering in the network through that node (message 1); the messages contain the selected metrics, e.g., packet and bit count for every traffic flow (see Table 4-3). Packet delay and other metrics are measured along the path and reported though INT messages to the P4 collector (message 2), which performs the aggregations specified in WF1 and computes statistics, e.g., min, average, and max of delay, jitter, and load. Periodically, the P4 collector sends the computed statistics to the collector in the local telemetry agent (message 3). The aggregated QoS telemetry message includes per-path statistics for every flow. The received statistics are then processed by each of the flow processors (message 4) that compute per-flow QoS telemetry measurements and send them to the corresponding flow agent (message 5). When flow QoS statistics are received in the multi-flow agent, the manager pushes both traffic and QoS measurements to the corresponding flow agent (message 6), which uses the current DRL model (see Subsection 4.3.2.F)) to take an action on the routing for the flow. Routing decisions are gathered by the manager (message 7) who eventually sends them to the P4 switch (message 8). The routing policies in message 8 are defined as the percentages of input traffic to be routed through each of the routes (see Table 4-3). Internally, the P4 switch translates this policy into flow rules to efficiently perform packet forwarding according to defined percentages.

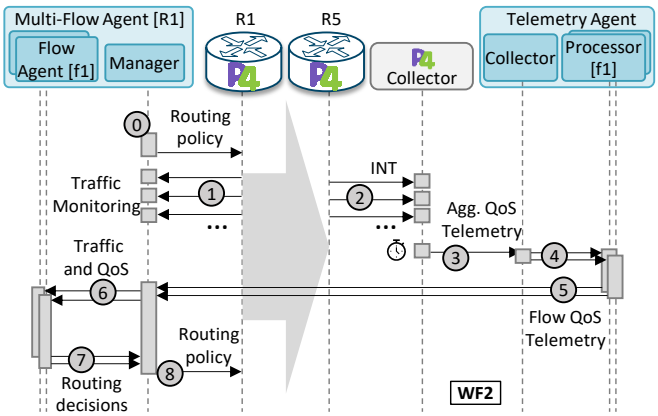


Figure 4.21: WF2: Near-real-time operation.

C) Resource reconfiguration

As defined in Subsection 4.3.2.G), the near-real-time operation of the flows between the same ingress and egress is continuously monitored by the manager in the *multi-flow agent* and summarized to the *multi-flow MAS supervisor* in the MLFO (message 9 in Figure 4.22). In the

case of detection of expected performance degradation due to the lack of networking resources, the *multi-flow MAS supervisor* computes the required capacity that needs to be added, the list of flows that will use it, and other requirements, like delay constraints, and sends a request to the SDN controller (message 10) (see Table 4-3). The SDN controller decides the action to be taken, e.g., increase the capacity of one of the existing paths or add a new path to the set of allowed paths. In this example, let us assume that the SDN controller decides to create an optical bypass connecting nodes R1 and R5. To that end, the SDN controller coordinates the setup of the new lightpath at the optical layer (message 11) and configures a new virtual link between the nodes and the new path (message 12). Finally, the SDN controller replies to the *multi-flow MAS supervisor*, via the MLFO, with the details of the action taken (message 13). For instance, the availability of a new path (see Table 4-3). In that case, the *multi-flow MAS supervisor* finds the most appropriate DRL model to operate each of the flows and sends the details of the new path together with the new models for the affected flows to the manager in the *multi-flow agent* (message 14), as well as the details of the new paths to the telemetry agent (message 15). Note that during this whole process, the flows were operated with the models in place and available resources, which were expected to produce performance degradation. In consequence, it is of paramount importance that the detection occurs with as much anticipation as possible to leave time for the new resources and models to be available before such degradation has any impact of the performance of the flows.

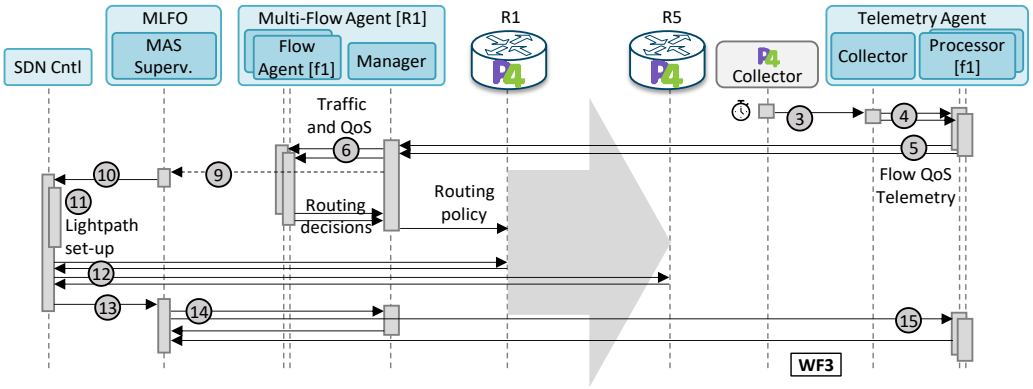


Figure 4.22: WF3: Resource reconfiguration during flow operation.

4.3.4 Experimental Results

In this section, we experimentally demonstrate the proposed distributed MAS fed with telemetry measurements to perform near-real time multi-flow operation on a federated testbed. The experimental set up, that reproduces the architecture and scenario depicted in Figure 4.18, is first described in detail. Next, the algorithms and workflows defined in Sections 4.3.2 and 4.3.3 are experimentally assessed, and finally, the performance of the proposed near-real time operation is experimentally evaluated.

A) Experimental Setup

Figure 4.23 presents the federated testbed and the experimental deployment, which reproduces a 5-node network scenario similar to that in Figure 4.18b. The packet- optical network is deployed at the CNIT/SSSA ARNO testbed, whereas the distributed MAS is deployed at UPC premises. Five native in-kernel P4-programmable software switches (P4-NIKSS) are deployed and connected to create the packet network. P4 switches are connected among them using 10G interfaces. P4 switches run on a separate server to maximize performance. Specifically, an Intel(R) Xeon(R) Gold 6238R CPU @2.20GHz server with 256GB RAM is used. An additional P4-based software process is deployed as telemetry data collector, which receives INT data from the P4 switches.

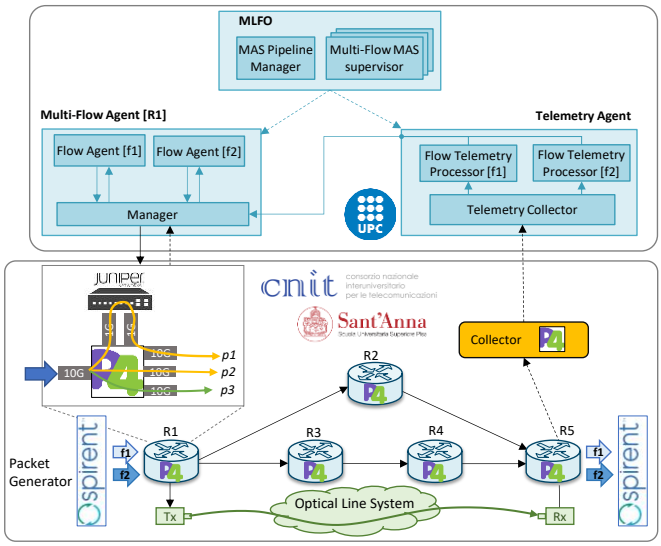


Figure 4.23: Federated testbed setup for the experiments.

In order to reproduce network congestion and selectively introduce delay in some of the paths of the traffic flows, a Juniper M10i router equipped with 1G interfaces is connected to switch R1, as depicted in the inner graph in Figure 4.23. Selected flows are forced to traverse the Juniper router before route switching, thus generating different congestion scenarios. Regarding the optical layer, the optical bypass between R1 and R5 is based on a pre-programmed packet-optical whitebox employing 10G and 100G optical pluggables. The pluggables are connected to an Optical Line System, with three 80km fiber spans and line optical amplifiers, by means of an arrayed waveguide gratings multiplexer. The optical connection between the two pluggables is established and enforced locally by OpenConfig-enabled SDN agents co-located with the SONiC node operating system. Finally, a Spirent SPT N4U equipped with 10G ports is used as packet generator for two different traffic flows (*f1* and *f2*), both from R1 to R5. In our case, flows are identified by source and destination IP. For validation purposes, the Spirent tool is also used as flow traffic sink. In the experiments, R1 measures per-flow traffic entering the switch, before

being forwarded through the different paths. In particular, three alternative paths are considered: *p1* (R1-R2-R5), *p2* (R1-R3-R4-R5), and *p3* (R1-R5, using the optical bypass active on demand).

As illustrated in Figure 4.19, the distributed system consists of several interconnected software components. Specifically, the *telemetry agent*, which is part of the distributed telemetry architecture, includes: *i*) a telemetry collector that periodically receives telemetry data from the P4 collector; and *ii*) a per-flow telemetry processor. In addition, *flow agents* running inside the *multi-flow agent* make routing decisions for each traffic flow. In this case, the multi-flow agent at R1 includes flow agents for both *f1* and *f2*. REST API interfaces have been implemented between agents and P4 systems. MAS agents and the MLFO, including all the components and interfaces have been implemented in Python 3.10.4 and run inside Docker containers. Three separated virtual machines (VM) for the agents and the MLFO, with Ubuntu Server 22.04 LTS as operating system, have been deployed to emulate a distributed environment.

B) Experimental assessment of the workflows

This section is devoted to the experimental demonstration of the workflows defined in Section 4.3.3: *i*) flow provisioning (WF1); *ii*) near-real time operation (WF2); and *iii*) resource reconfiguration (WF3). The captures in this section use the labels for the exchanges messages as defined in the related workflow in Section 4.3.3. The information in most of the messages (see Table 4-3) is encoded as a JSON object. Note that messages exchanged between functional blocks in the same system, e.g., inside the multi-flow agent, are not captured.

WF1 is triggered after a new flow request has arrived to the SDN controller in the control plane, which has run its provisioning procedure to properly configure the nodes in the data plane. Figure 4.24 presents the exchanged HTTP messages captured with Wireshark (see WF1 in Figure 4.24 for details). The SDN controller sends a request to the MLFO to deploy and configure the MAS in charge of near-real time control of that flow (message *a*). Figure 4.24 shows the details of message *a* that includes the id of the flow, the QoS parameters for the flow, the details of the ingress and egress nodes and the list of paths that can be used for the flow (*p1* and *p2*). Once the agents are deployed in the selected locations, the MLFO deploys a multi-flow MAS supervisor for the new MAS, which uses the received information to create descriptors to configure the multi-flow agent collocated with R1 and the telemetry agent collocated with the P4 collector (messages *d*). Finally, the MLFO replies the SDN controller (*e*); the details of message *e* are shown in Figure 4.24, which include the reference of the agents for the P4 agents of R1 and collector to communicate with them, the metrics to be collected from the nodes and the configuration, which includes the collection period and the activation of data aggregation for that period. The SDN controller configures the agents in the P4 nodes supporting the flow, as well the P4 collector and activate the telemetry with the received configuration (*f*) and WF2 for near real time flow routing operation starts.

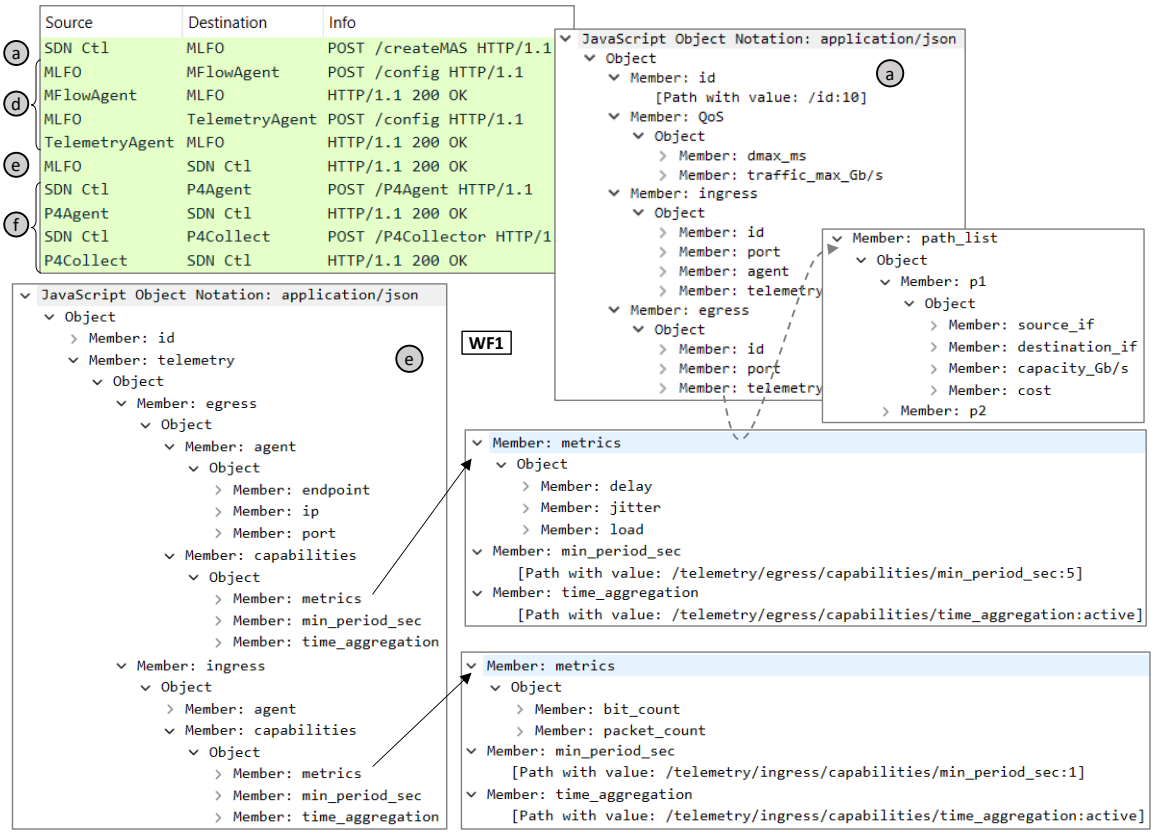


Figure 4.24: WF1: Capture and details of messages.

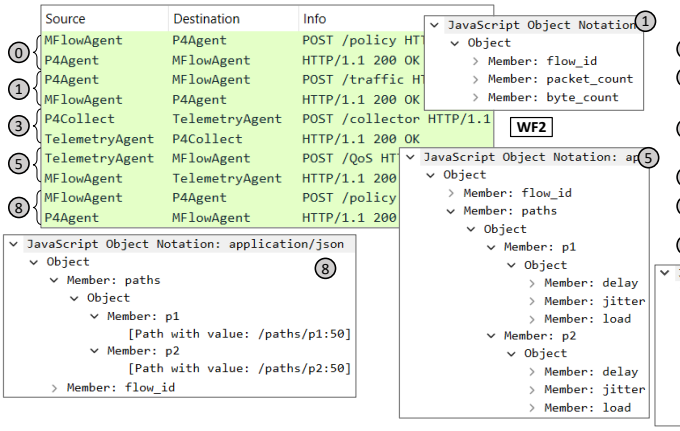


Figure 4.25: WF3: Capture and details of messages.

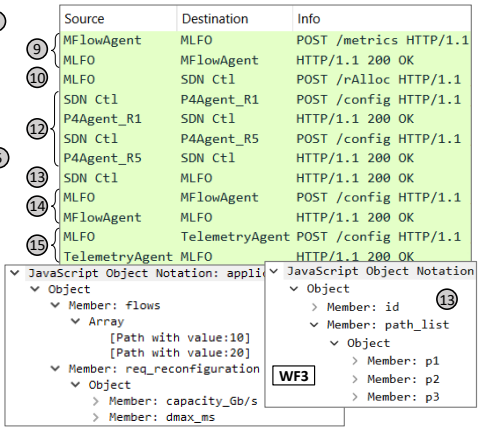


Figure 4.26 WF2: Capture and details of messages.

Figure 4.25 presents the capture of the HTTP messages exchanged during the first iteration in WF2 (see WF2 in Figure 4.21 for details). The configuration of flow agent includes the initial routing policy that is applied when the flow operation starts (message 0); new policies will be defined based on the subsequent measurements. Traffic volume is measured by the agent in the ingress P4 node and sent to the manager in the multi-flow agent (1). The details of message

1 are presented in Figure 4.25 and include the counters of packets and bytes for the flow. The P4 collector receives and aggregates INT measurements for the flow and sends them to the collector in the telemetry agent (3), which processes them and sends to the manager in the multi-flow agent (5). Message 5 includes measurements per path of the required QoS parameters (delay, jitter and load in our case) during the collection period (see Figure 4.25). The manager, takes both, traffic and e2e measurements and sends them to the flow agent for decision making. The new routing policy is then sent to the ingress P4 node agent (8). An example of routing policy in message 8 is shown in Figure 4.25, where the traffic of the flow is routed evenly on the available paths *p1* and *p2*.

WF2 continues and policies are computed based on the received measurements. The operation of this flow and other flows between the same ingress and egress nodes is continuously overseen by a common multi-flow MAS supervisor in the MLFO (message 9 in Figure 4.26). In the case that degradation because of lack of capacity is anticipated, more resources need to be added and WF3 formally starts to reconfigure the involved flows (see WF3 in Figure 4.22 for details). In such case, the multi-flow MAS supervisor sends a request to the SDN controller specifying the required increment of capacity and delay constraints of such capacity (10); see the details of message 10 in Figure 4.26. In the experiments, the SDN controller decides to create an optical bypass connecting R1 and R5 directly, and the controller configures the virtual link in the two P4 switches (12). Once the optical bypass is configured, the SDN controller informs the MLFO (13) and includes the list of flows and the paths that can be used for their operation (see Figure 4.26). Finally, the MLFO updates the configuration of both the multi-flow and the telemetry agents (14-15). The new configuration entails deploying new models for the agents so they can make routing decisions on three paths.

C) Performance evaluation

The performance of the proposed near-real time operation was evaluated on the federated testbed in Figure 4.23. The Spirent packet generator was configured to reproduce traffic for flows *f1* and *f2*, while paths *p1* and *p2* through the packet network were initially available for supporting the flows. Additional traffic was generated between R3 and R4 with a constant pattern very close to 10Gb/s, which made that routing traffic through *p2* resulted in very high delay, which in practice forced to use *p1* only. The e2e delay was measured by the Spirent connected at R5. Figure 4.27a (up) shows the traffic generated (48 hours) for flows *f1* and *f2*, as well as the aggregated traffic (*f1*+*f2*). The traffic was intentionally generated with the aim to generate daily variation with different peaks, being the highest one at 16h with values close to 1Gb/s, which is the capacity of the interfaces in the Juniper router. Note that the traffic of the individual flows follows a clearer and sharper periodic pattern than the aggregated one, that presents more differences between consecutive days.

In order to illustrate the proposed near-real-time operation, we setup a maximum delay to be assured (d^{max}) for the agents to make the appropriate flow routing actions. To this end, we conducted a preliminary test where optical bypass is never active. Figure 4.27a (down) shows the obtained e2e delay for both flows with a very high peak due to congestion in the input interface to the Juniper router. Note also that the e2e delay is under 0.6ms (with some variation) during the rest of the analyzed period. In consequence, we configured d^{max} = 0.6ms during flow deployment in WF1.

Figure 4.27b presents the obtained results when the architecture proposed in Section 4.3.1 was deployed and d^{max} is set to 0.6ms. Figure 4.27b (up) shows the aggregated traffic (*f1+f2*), and the traffic through the packet network (*p1+p2*) and through the optical bypass (*p3*) when it is active. We observe that only *f1* used the optical bypass, whereas *f2* was always routed through the packet network. Figure 4.27b (down) presents the obtained delay, where we observe that d^{max} was never exceeded. In addition, during the periods of high traffic, the optical bypass was activated since the flow supervisor anticipated the possibility of excessive delay. It is worth noting that during the second traffic peak, at around 42h, the bypass was activated although the e2e delay did not exceed d^{max}. This was as a result of the anticipation for the prediction in Algorithm 4-4 (parameter *m*). Such anticipation needs be configured as a trade-off of the risk of providing episodes with excessive delay (if *m* is short) and the risk of activating the optical bypass unnecessarily (if *m* is long).

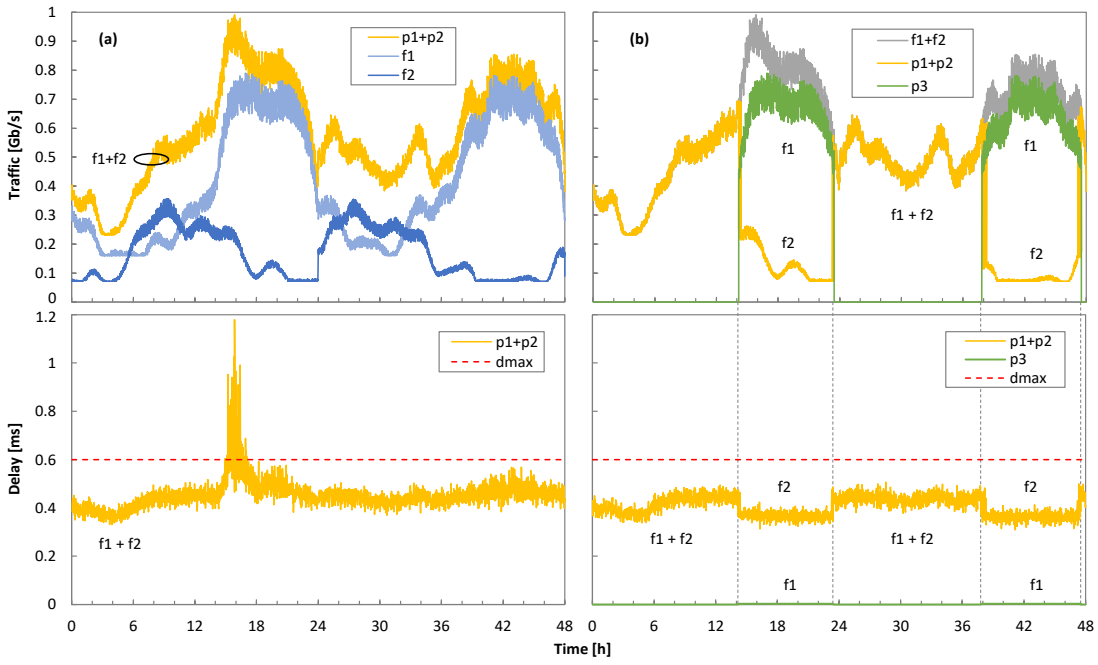


Figure 4.27: Experimental measurements. Flows use the packet network only (a); the optical bypass is used to reduce delay (b).

Dissemination Level	PUB (Public)

4.3.5 Concluding Remarks

A distributed control architecture based on MAS to assist the SDN controller to make autonomous near-real-time decisions for QoS assessment has been proposed and experimentally assessed. In particular, autonomous flow routing was demonstrated focused on ensuring e2e delay.

Flow operation was modeled with three workflows for flow provisioning, near-real-time operation, and resource reconfiguration. During flow provisioning, the SDN controller selects available network resources for the flow and coordinates with the MLFO to deploy a MAS pipeline in charge of its near-real-time control. The proposed solution considers coordination among multiple traffic between the same ingress and egress nodes, which can share resources in the network. During flow operation, DRL-based flow agents make near-real-time routing actions to provide the required performance with the minimum cost based on the collected telemetry. A multi-flow MAS supervisor was proposed to track the performance of the flow agents. In the case of QoS degradation is anticipated, the required increment of capacity computed and requested to the SDN controller.

The proposed distributed control architecture, algorithms and workflows were experimentally assessed in a federated testbed spanning UPC and CNIT premises. Details of the messages exchanged highlighting the information exchanged. The performance was experimentally evaluated and show that the obtained e2e delay was kept under the maximum. Anticipation of performance degradation was demonstrated which triggered the activation of an optical bypass. This anticipatory activation of the optical bypass underscores the balance between preventing excessive delays and avoiding unnecessary bypass activation which may increase both capital and operational costs of the network.

4.4 Optimizing Energy-Efficiency with AI-Assisted SDN Controller in Integrated Packet Optical Transport Network

4.4.1 Description

In [D42], a scenario was presented in which packet nodes equipped with small form factors DWDM coherent transceivers (i.e., *pluggables*) are integrated into an Optical Line System (OLS) based on an Elastic Optical Network (EON). This integrated packet–optical architecture, commonly referred to as IP-over-WDM (IPoWDM) [Gum22], is regarded by operators as a highly promising solution for future metro and core network infrastructures. By deploying different generations of coherent pluggable transceivers (e.g., 400/800 ZR+), the transport capacity of the IPoWDM infrastructure can be flexibly and in a scalable way adapted to evolving traffic demands. Additionally, traditional external multi-rate transponder providing the transmission

between packet "grey" ports and OLS are removed. This does reduce the cost, footprint and power consumption. The latter is related to enhancing sustainability in transport networks, which is becoming a crucial goal for operators. The macroscopic objective is to improve the balance between average network throughput (Gb/s) and the network power consumption of considering all active devices such as ROADMs, routers, optical line amplifiers, etc. In D42, Section 3.4.4.1 presents the analytical expressions used to model power consumption associated with activating various network elements and devices. Notably, the power consumption of coherent transceivers varies depending on their selected operational mode. These transceivers can be programmed with different symbol rates (e.g., 30 or 60 Gbaud) and modulation formats (e.g., DP-QPSK, DP-8QAM, DP-16QAM), offering a range of net bitrates, transmission distances, and corresponding power requirements.

The proposed power model serves as an estimation tool integrated into energy-aware routing and resource selection algorithms [Zha22]. This allows the algorithm, when provisioning a connectivity service request (*D*)-defined by source, destination, and requested bandwidth-to not only satisfy service requirements (e.g., bandwidth) but also minimize the activation of network components, thereby improving energy efficiency. In a nutshell these algorithms modify/enhance traditional Routing Modulation and Spectrum Assignment (RMSA) algorithms to output solutions that lead to minimize the power consumption when accommodating the connectivity service. These algorithms are referred to as EA-RMSA. In D42 [D42], three candidate EA-RMSA algorithms were presented based on either heuristic approach or AI/ML trained strategies (based on Deep Reinforcement Learning, DRL):

- **KSP-FF:** The K-Shortest Path First-Fit algorithm computes up to *K* paths between the source and the destination nodes for request *D*. It prioritizes that the transceivers at the source and destination packet nodes are configured with the highest operational mode (OMs). Highest operational modes (e.g., 60 Gbaud and DP-16QAM) are associated to those offering maximum bitrate (e.g., 400 Gb/s) if the maximum path distance is not over exceeded. For that OM_i, it is determined the number (*m*) of transceivers and optical flows to be allocated across every feasible k^{th} path. For each *m* optical flow, the algorithm selects the central frequency and slot that satisfy the end-to-end optical continuity and contiguity constraint. In other words, a connection service demanding *b* data rate can be accommodated over a set of *m* optical flows allocating diverse central frequencies but routed over the same spatial path, i.e., across the same nodes and links. Observe that this algorithm does not tackle any objective on power savings. This is here used as a benchmark for network performance parameters.
- **EA-KSP-FF**: extends KSP-FF to further minimize network power consumption by computing power increases for each *k*-path using the presented power model. Feasible K paths are sorted by power consumption and hops, with EA-KSP-FF favoring paths through already active devices to avoid powering on new elements.

- **DRL EA-KSP:** leverages the training of a DRL agent through interaction with network environments to learn routing and resource allocation decisions that optimize key performance metrics such as energy efficiency. A key advantage of DRL lies in its ability to dynamically adapt to varying network conditions and service demands without requiring explicit rule-based programming, making it a powerful tool for enhancing EA-RMSA decisions. Details of the observation and action spaces and reward are provided in D42 [D42].

In the following section, the performance evaluation discussed in [D42] is complemented addressing two selected network scenarios: Scenario A where packet nodes are equipped with limited number of transceivers; and Scenario B where packet nodes have large number of transceivers. The performance discussions are done in terms of different metrics such as Blocked Bandwidth Ratio (BBR), power consumption, energy efficiency, transceiver and operational modes utilization.

4.4.2 Complete Performance Evaluation

The considered IPoWDM network topology is shown in Figure 4.28 which comprises 9 terminal nodes (IP routers) and an OLS with 14 ROADMs connected by 21 bidirectional optical links. Optical Line Amplifiers (OLAs) are deployed along the fiber links at fixed intervals of 80 km. The optical spectrum spans from 191.3 THz to 196.1 THz, corresponding to the supported channel frequency range of the DWDM ZR+ transceivers. This range yields 768 Nominal Central Frequencies (NCFs), spaced at 6.25 GHz. Each provisioned optical flow used by a request D is associated with a coherent DWDM transceiver pluggable and occupies a fixed slot width of 50 GHz (i.e., 8 NCFs). Each terminal IP router connected to a ROADM is equipped with *N* DWDM coherent transceivers. Two scenarios are considered by varying the number of transceivers *N* per terminal node: Scenario-A assumes limited transceiver availability with *N=5*, while Scenario-B represents a more provisioned network with *N=15* transceivers installed at each terminal node. The *D* requests are generated following a Poisson arrival process with an inter-arrival time of 50 seconds. Each request's holding time (HT) is modeled as an exponential distribution, with the mean HT ranging between 100–600 seconds for scenario-A, and 2000–3500 seconds for scenario-B. Requested data rate (b) is randomly selected from the set {100, 200, 300, 400} Gb/s. Source and destination nodes are also randomly selected among the 9 terminal IP routers depicted in Figure 4.28. The number of candidate paths *K* is set to 3 for all the algorithms.

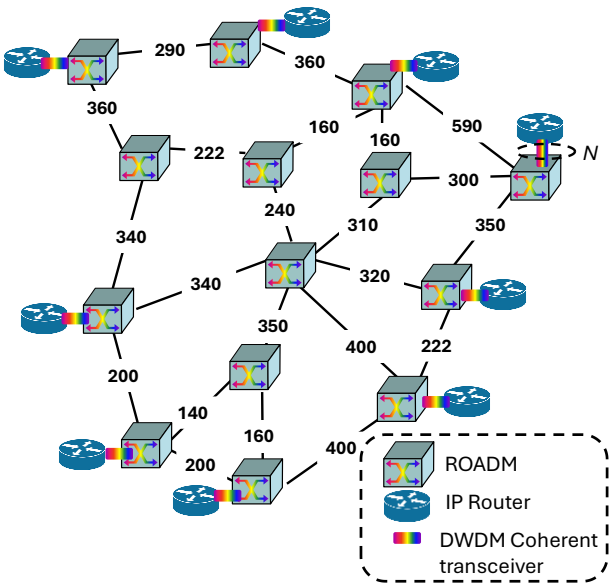


Figure 4.28: IPoWDM Transport Network Topology – Routers equipped with N 400G ZR+ DWDM transceivers.

Bandwidth Blocked Ratio (BBR) and Energy Efficiency

Figure 4.29 shows the Bandwidth Blocking Ratio (BBR) as a function of the holding time (HT) for the KSP-FF, EA-KSP-FF, and DRL-based EA-KSP RMSA strategies under Scenario-A. A D requests is blocked if either: (1) the RMSA algorithm fails to find sufficient contiguous and continuous spectrum to accommodate all *m* optical flows required by *D*, or (2) there are fewer than m unused transceivers available at either the source or destination node.

The results obtained show that KSP-FF approach consistently achieves the lowest (i.e., best) BBR across all traffic loads. This is expected, as KSP-FF targets maximizing resource utilization—such as optical spectrum and transceivers—to improve connectivity service provisioning and throughput. However, it does so without considering the implications for power consumption. That is, KSP-FF does not attempt to reduce the number of active ROADMs and OLAs, nor does it promote the use of lower-power transceiver operational modes.

The EA-KSP-FF strategy is specifically designed to improve the energy efficiency of the baseline KSP-FF algorithm. Energy efficiency is defined as the average network power consumption (in kW) per unit of served throughput (in Gb/s), where lower values indicate better performance. A common way to enhance this metric is to significantly reduce power consumption while maintaining— or only slightly decreasing— the overall served throughput. As illustrated in Figure 4.29b), the EA-KSP-FF strategy achieves energy efficiency improvements ranging from 1.6% to 5.6% over KSP-FF, depending on the traffic load. This improvement stems mainly from the EA-KSP-FF's ability to prioritize the reuse of already active OLS components—such as ROADMs and OLAs along the links—when provisioning *D* services. Specifically, EA-KSP-FF reduces overall network power consumption by 3.1% to 8.6% compared to KSP-FF, as reported in Table 4-4.

However, these power savings come at the cost of reduced network throughput, which in turn leads to a higher BBR. For example, at HT = 600s in Scenario-A, EA-KSP-FF achieves an average throughput of 2656.6 Gb/s, compared to 2756.9 Gb/s with KSP-FF (see Table 4-4). Consequently, the BBR increases from 10.2% with KSP-FF to 13.5% with EA-KSP-FF. The BBR degradation stems from EA-KSP-FF's tendency to route service requests over paths that traverse a higher number of already active network elements to minimize the power consumption which in turn, results in longer paths with more hops. At HT = 600s, the average number of hops is 4.53 and 4.76 for KSP-FF and EA-KSP-FF, respectively (Table 4-4). *D* services established over longer paths lead to poorer resource utilization, making it more difficult to satisfy constraints such as spectrum continuity and contiguity.

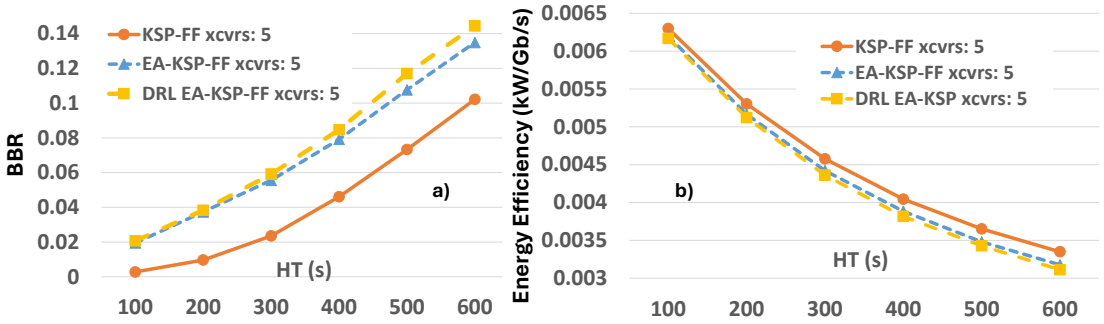


Figure 4.29: Scenario-A [N = 5]: a) Blocked Bandwidth Ratio (BBR) vs HT (s); b) Energy Efficiency (kw/Gb/s) vs HT (s).

The DRL EA-KSP strategy is trained on diverse network states to learn optimized routing policies aimed at further improving the energy efficiency achieved by the EA-KSP-FF strategy. In Scenario-A, Figure 4.29b shows that this improvement becomes most evident under high traffic loads (HT = 500 or 600), with gains ranging from 1.5% to 2%. At these traffic loads, as more network resources (e.g., ROADMs and link OLAs) are active, the trained DRL model can guide RMSA decisions that further reduce power consumption, thereby enhancing energy efficiency. At HT = 600s, DRL EA-KSP and EA-KSP-FF attain network power consumptions of 8.1 kW and 8.4 kW, respectively. However, as mentioned before, reducing power consumption may lead to worsening both BBR and throughput performance due to routing connections over longer paths. Notably, the DRL EA-KSP approach tends to route services along paths with the highest number of hops.

Table 4-4: Average number of hops, average network power consumption (kW), and average network throughput (Gb/s) for Scenario-A [N = 5] and Scenario-B [N = 15].

Scenario	HT(s)	Algorithm	Hops	Power (kW)	Throughput (Gb/s)
A [N=5]	100	KSP-FF	4.55	3.41	541.9
		EA-KSP-FF	4.68	3.30	533.7
		DRL EA-KSP	4.80	3.27	533.1

	600	KSP-FF	4.53	9.23	2756.9
		EA-KSP-FF	4.76	8.4	2656.6
		DRL EA-KSP	5.12	8.1	2621.5
B [N=15]	2000	KSP-FF	4.55	12.7	10057.1
		EA-KSP-FF	4.78	11.4	9939.3
		DRL EA-KSP	5.19	10.9	9879.2
	3500	KSP-FF	4.53	13.9	15932.8
		EA-KSP-FF	4.81	12.7	15720.5
		DRL EA-KSP	4.96	12.0	15425.1

Scenario-B evaluates how increasing the number of coherent transceivers at the packet nodes (N = 15) affects the overall performance of the different RMSA approaches. Figure 4.30a) and b) depict the BBR and energy efficiency for HT values ranging from 2000 to 3500 seconds, respectively. Note that HT values differ from those used in Scenario-A and are chosen to attain BBR levels comparable to those in Scenario-A.

Notably, the trained DRL EA-KSP model achieves higher energy efficiency gains compared to the EA-KSP-FF strategy, with improvements ranging from 3.2% to 6.6%. These gains result from both a reduction in the network power consumption by routing through already active ROADM and OLA devices (see Table 4-4), and by favoring the configuration of lower-power transceiver OMs. Figure 4.31 illustrates the power consumption distribution across different network elements (i.e., ROADMs, OLAs, routers, and transceivers). Clearly, both EA-KSP-FF and DRL EA-KSP strategies lead to significant reductions in the power consumed by ROADMs and OLAs. Adopting the trained DRL model yields additional power savings due to its ability to select lower-power operational modes for the transceivers. However, like Scenario-A, these power savings are achieved by routing D services over longer paths with increased hop counts, which ultimately degrades key network performance metrics such as BBR and throughput.

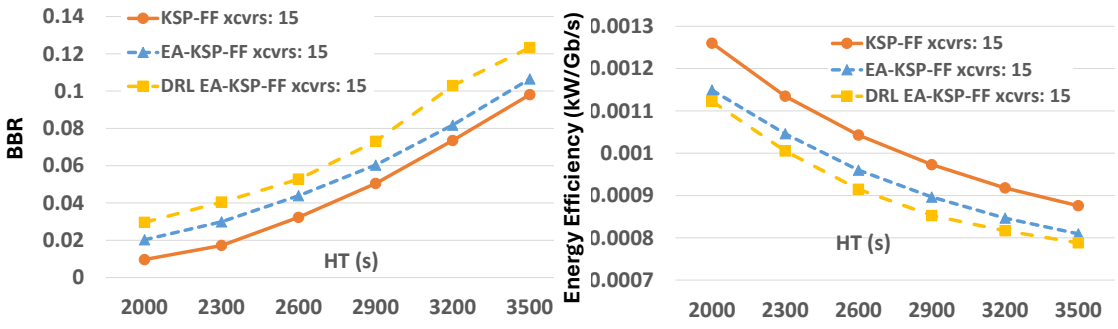


Figure 4.30: Scenario-b [N = 15]: a) Blocked Bandwidth Ratio (BBR) vs HT (s); b) Energy Efficiency (kw/Gb/s) vs HT.

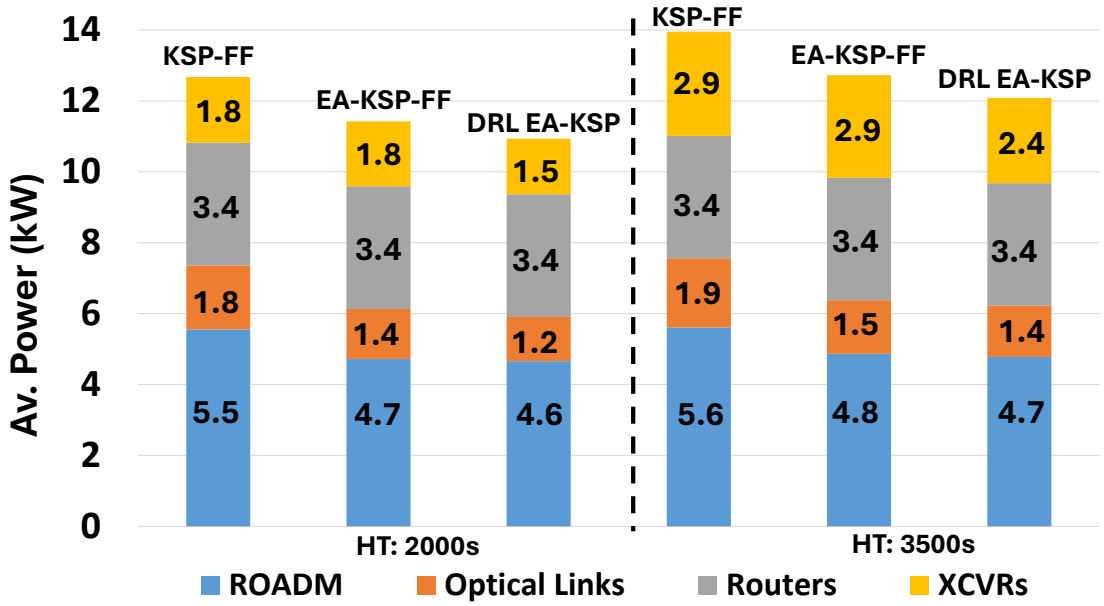


Figure 4.31: Average power consumption (kW) for HT = 2000s and 3500s per network device type for Scenario-B.

Transceiver utilization and operational mode selection

Table 4-5 presents, for both Scenario-A and Scenario-B, the average transceiver utilization (%) and the distribution of selected operational modes (OMs) across varying traffic loads for each RMSA strategy under consideration. These metrics complement the previously discussed network and sustainability performance indicators, offering further insights into transceiver resource usage patterns.

Table 4-5: Average transceiver and operational mode (OM) utilization (%) for Scenario-A [N = 5] and Scenario-B [N =15].

Scenario	HT(s)	Algorithm	Transceiver utilization (%)	Operational Modes Utilization %			
				30Gbaud DP-QPSK	60Gbaud DP-QPSK	60 Gbaud DP-8QAM	60GBaud DP-16QAM
A [N=5]	100	KSP-FF	21.9	0	25.2	44.3	30.5
		EA-KSP-FF	22.2	1.3	25.8	41.9	30.6
		DRL EA-KSP	22.7	26.4	43.1	23.2	7.3
	600	KSP-FF	35.27	0	24.1	44.3	31.6
		EA-KSP-FF	34.8	1.8	25.4	40.3	32.4
		DRL EA-KSP	36.0	27.7	46.5	17.9	7.8
B [N=15]	2000	KSP-FF	37.9	0	25.1	44.2	30.7
		EA-KSP-FF	37.9	0.4	28.4	40.4	30.8
		DRL EA-KSP	40.3	26.7	52.2	16.2	4.9

	3500	KSP-FF	59.7	0	24.2	44.5	31.3
		EA-KSP-FF	59.6	0.07	30.0	38.7	31.2
		DRL EA-KSP	60.9	26.7	49.3	17.3	6.7

Regardless of the scenario or traffic load, the DRL EA-KSP strategy consistently exhibits the highest average transceiver utilization. Recall that provisioning a request *D* under a given operational mode OM_i may require the allocation of *m* transceivers along with *m* corresponding optical flows. Therefore, if DRL EA-KSP is the strategy occupying the most transceivers, it also results in the highest number of optical flows being established. In other words, applying DRL EA-KSP leads to increased transceiver usage and higher optical resource occupancy along the path links. Such a high transceiver utilization in the DRL-trained RMSA solution is primarily due to its frequent selection of lower-power OMs, compared to both EA-KSP-FF and KSP-FF strategies. By favoring such configurations, the DRL EA-KSP approach aims to further reduce power consumption associated with coherent transceivers. Notably, operational modes 1 and 2—corresponding to the DP-QPSK modulation format with symbol rates of 30 and 60 Gbaud, respectively—exhibit a combined utilization rate between 76% and 79%. In contrast, the EA-KSP-FF strategy utilizes these low-power OMs in only about 30% of cases.

In its effort to reduce power consumption and improve energy efficiency, the DRL EA-KSP algorithm increases path lengths (i.e., hops) to exploit the reuse of already active devices, as previously discussed. Additionally, DRL EA-KSP tends to select a greater number of lower-power transceivers, thereby increasing the number of required optical flows. When combined, these two effects lead to a more intensive utilization of spectral resources in the IPoWDM network. As a result, the performance of DRL EA-KSP—and to a lesser extent EA-KSP-FF—in terms of BBR and network throughput is significantly degraded compared to non-energy-aware strategies such as KSP-FF.

4.5 DRL-BASED ROUTING AND SPECTRUM ASSIGNMENT FOR ENHANCED NETWORK PERFORMANCE

4.5.1 Latency-aware RSA based on DRL

The fundamental problem of service provisioning in EONs is the routing and spectrum assignment (RSA). Traditional heuristic RSA algorithms often struggle to achieve optimal solutions due to the dynamic nature and inherent complexity of EONs, especially when considering strict requirements like high bandwidth and low latency for future 5G and 6G communications. To address these limitations. Section 3.4.5 of [D42] introduces a DRL-based RSA solution that tackles these issues by incorporating latency constraints alongside spectrum allocation.

This approach integrates a DRL agent within the SDN control plane, where the path computation function is externalized to a dedicated ML-assisted Path Computation Element (PCE). As outlined in [D42], the DRL agent is trained using a network state representation that captures spectrum utilization and end-to-end latency information for candidate paths. Its discrete action space enables the selection of one of the candidate paths, guided by a reward function designed to incentivize successful allocations by rewarding higher bandwidth demands and lower latency, and penalizing failed attempts with a negative value. This design enables the agent to dynamically adapt to network conditions and traffic patterns, enhancing spectrum efficiency and overall network performance.

4.5.1.1 Performance Evaluation

To validate its effectiveness, performance evaluations were conducted through extensive simulations on two distinct EON topologies, shown in Figure 4.32: the Barcelona City Network (BCNNet), a metropolitan network with 14 nodes and 42 bidirectional links, and the Great Britain's British Telecom Network (BTNet), a national core network comprising 22 nodes and 70 links. Both networks were modeled with 100 Frequency Slots (FSs) per link, and connection requests followed a Poisson process with varying bandwidth (2, 4, or 8 FSs) and latency requirements (0.30-0.60 ms for BCNNet, 1.25-2.5 ms for BTNet). The DRL approach was rigorously compared against a traditional heuristic called the k-shortest path routing and first-fit spectrum allocation (kSPFF) algorithm, which selects paths based on lowest latency among candidates.

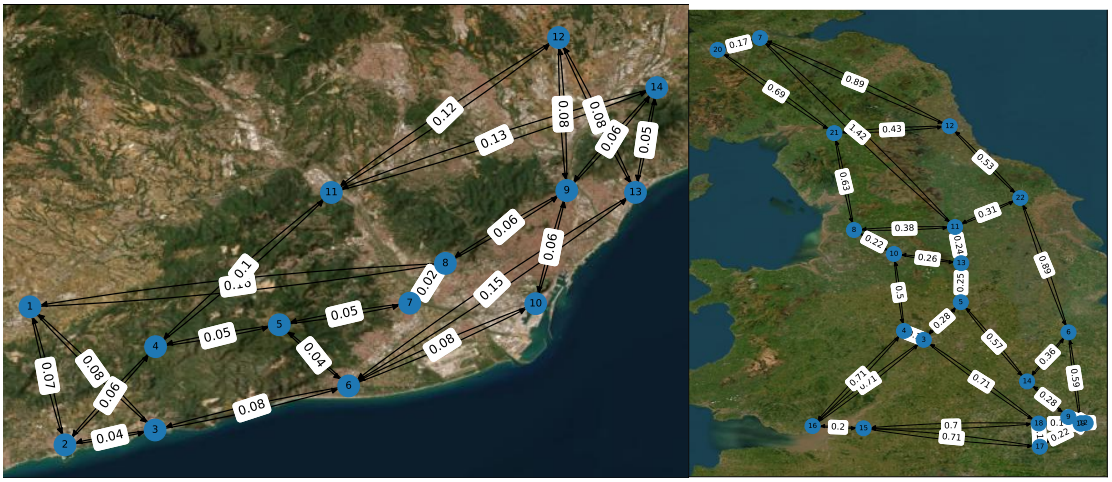


Figure 4.32: BCNNet and BTNet topologies used for performance evaluation.

The simulations provided compelling evidence of the DRL approach's superior performance across several key metrics:

Bandwidth Blocking Ratio (BBR)

As illustrated in Figure 4.33, the DRL agent demonstrated a notable reduction in BBR compared to kSPFF, especially under increasing traffic loads. For instance, at a moderate traffic load of 150 Erlangs in the BCNNet, the DRL agent achieved approximately a 16.20% reduction in BBR, an improvement that further increased to around 18.35% at a high traffic load of 250 Erlangs. This significant improvement underscores the DRL agent's ability to learn and adapt to diverse network conditions, making more optimal path and resource selections even when optical spectrum resources become scarce. Similar positive trends were observed in the larger BTNet topology, indicating the scalability of the DRL-based approach regardless of network size.

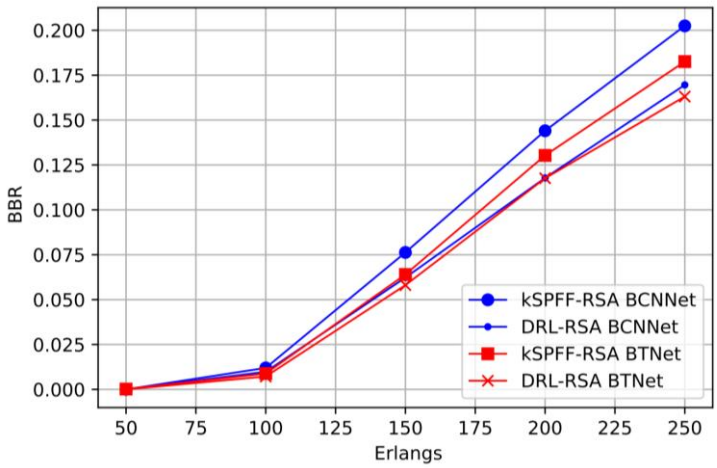


Figure 4.33: BBR benchmarking of kSPFF and DRL RSA.

Average Network Throughput

The DRL agent consistently led to a greater average network throughput across both evaluated topologies, as shown in Figure 4.34. In the BCNNet, the DRL agent achieved an average throughput of 10.4 Tb/s, surpassing kSPFF's 9.9 Tb/s, which represents a significant 5.1% improvement. Similarly, for the BTNet, the DRL agent reached 10.3 Tb/s, outperforming kSPFF's 10 Tb/s by 3.1%. This enhanced throughput is attributed to the DRL agent's superior optical resource utilization and its improved capability to fulfill latency restrictions simultaneously.

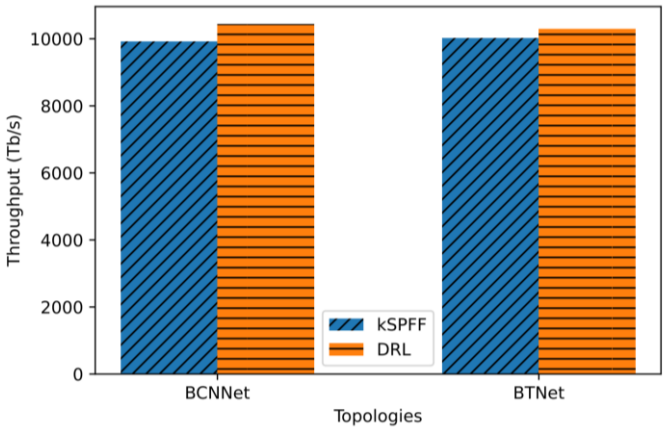


Figure 4.34: Network throughput attained by kSPFF and DRL RSA in BCNNet andBTNet topologies.

Path Computation Time

The Cumulative Distribution Function (CDF) plot in Figure 4.35 shows that the DRL approach exhibited a significantly lower mean execution time for path computation. It achieved a mean execution time of 1.04 ms with a standard deviation of 0.17 ms, whereas kSPFF recorded a mean of 1.91 ms with a standard deviation of 0.28 ms. This efficiency stems from the DRL approach leveraging a pre-trained ML model for inference, which rapidly returns a selected path based on its learned features. In contrast, kSPFF must iteratively check each of the k candidate paths against all connection requirements, leading to longer computation times.

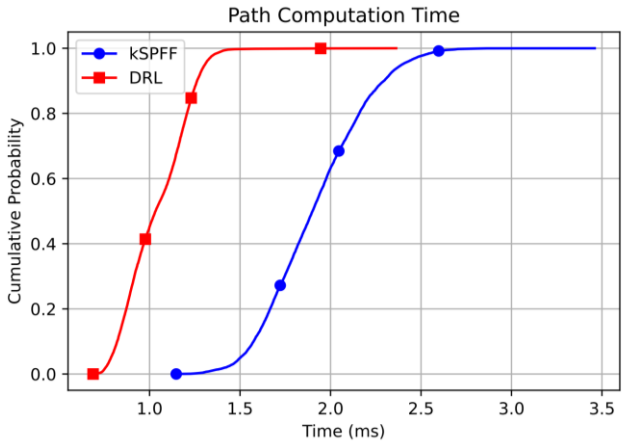


Figure 4.35: CDF of the path computation time for kSPFF and DRL-agent mechanisms in BCNNet.

Fulfillment of Latency Requirements

Figure 4.36 illustrates the Probability Density Function (PDF) of service of service failure attributed to unmet latency needs within the BCNNet topology. The results show that the DRL approach successfully accommodates a larger number of connectivity services that have

stringent (i.e., lower) end-to-end latency demands when compared to the kSPFF algorithm. A significant observation is that with the DRL agent, services requiring lower latency experience fewer connection failures (lower blocking probability) than those connections that request more flexible (higher) latencies. This performance advantage is directly attributable to the design of the DRL training process: the end-to-end service latency requirement is a key performance objective explicitly optimized through the carefully crafted reward function. Consequently, the DRL agent is inherently tailored to handle latency-sensitive connections more effectively, resulting in a reduced probability of service failure for such critical requests.

In contrast, the kSPFF algorithm exhibits a relatively uniform rejection probability across all latency requirements. This behavior stems from kSPFF's primary objective when processing a new connection request, which is to optimize the overall utilization of spectral resources by simply selecting the first shortest path (from a set of k candidates) that satisfies the specified latency requirement, without explicitly targeting or optimizing for minimal end-to-end latency. Overall, these findings underscore the importance of incorporating latency awareness in RSA algorithms for EONs, with the DRL-based approach proving highly beneficial for latency-sensitive applications.

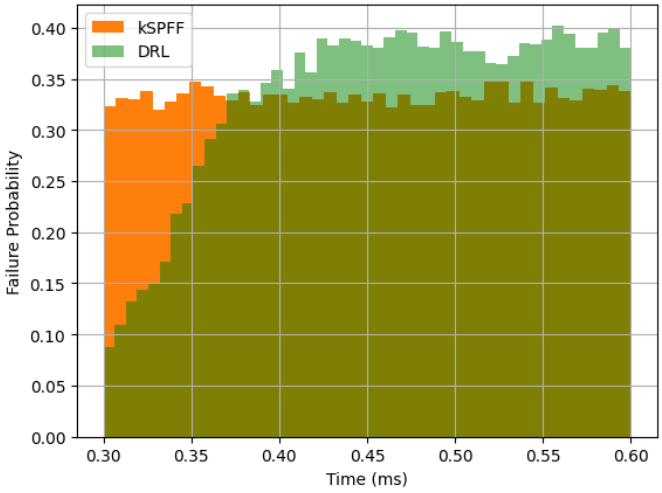


Figure 4.36: PDF of failed services with respect to end-to-end latency demands in the BCNNet.

4.5.2 QoT-Aware DRL-based RMBSA in MultiBand Elastic Optical Networks

Building upon the previously reported work, which primarily focused on EONs operating in the conventional C-band, this section presents an extension of this DRL-based approach to MultiBand Elastic Optical Networks (MB-EONs), specifically incorporating Quality of Transmission (QoT) awareness. This development addresses the need to scale network capacity

beyond the C-band by utilizing additional bands like L and S, which are critical for meeting the continuous growth in global data traffic.

The continuous growth in global data traffic necessitates innovative approaches to increase network capacity beyond the conventional C-band, leading to the development of MB-EONs that incorporate additional bands like L, S, U, E, and O. While extending transmission spectrum is cost-effective, managing spectrum resources dynamically in MB-EONs, especially while ensuring Quality of Transmission (QoT) for each connection, presents significant challenges due to the diverse physical layer characteristics across bands and dynamic traffic patterns. QoT is influenced by Amplified Spontaneous Emission (ASE) generated by optical amplifiers and Non-Linear Interference (NLI) noise (including Self-Phase Modulation, Cross-Phase Modulation, Multi-Channel Interference, and critically, Inter-Channel Stimulated Raman Scattering (ISRS). QoT is typically quantified by metrics like Optical Signal-to-Noise Ratio (OSNR) or Generalized Signal-to-Noise Ratio (GSNR).

Traditional network control methods, often reliant on rule-based heuristics, are generally not optimally designed for dynamic intricacies and interactions specific to multiband systems, especially when fully accounting for physical layer impairments. DRL has emerged as a promising paradigm to address the complex RMBSA problem in MB-EONs. Previous DRL approaches for EONs, however, are often simplified QoT models or primarily focused on spectrum utilization.

4.5.2.1 QoT-Aware DRL Framework

To overcome these limitations, a DRL framework must effectively incorporate detailed physical layer information into its decision-making process. This involves designing a DRL agent that interacts with an environment modeling MB-EON dynamics and includes a QoT estimation unit capable of providing accurate lightpath feasibility based on physical layer impairments.

State Representation

For QoT-aware RMBSA, the DRL agent's state representation is crucial. It includes network status, traffic information, spectrum utilization, and critically, QoT-level information for candidate lightpaths (e.g., estimated GSNR, maximum bit rate supported) and multiband specific features like band utilization.

Action Space:

The agent's action space typically involves selecting a route (from candidate paths), a frequency band, and a starting spectrum slot. Action masking techniques can be employed to filter out invalid actions, such as paths lacking sufficient spectrum or failing QoT checks.

Reward Function:

The reward function for QoT-aware provisioning explicitly incorporates QoT metrics to encourage the agent to select feasible lightpaths that meet QoT demands. For instance, the

reward can be the GSNR value of a selected lightpath if the traffic request is accepted, or a negative GSNR value if the chosen path/resources fail QoT requirements.

4.5.2.2 Control Plane Architecture

The overall architecture, depicted in Figure 4.37, integrates an intelligent DRL agent and a QoT estimation unit within an SDN control plane, specifically showcasing interaction with the CTTC FlexOpt SDN controller. This architecture builds upon previous setups reported in [D42] by extending to a multiband environment (C, L, and S bands) and introducing explicit QoT awareness. The SDN controller is responsible for receiving traffic requests and initiating service provisioning, delegating the RMBSA task to an external ML-assisted PCE. The PCE hosts the DRL agent and the QoT estimation unit. The QoT estimation unit can leverage analytical physical layer models (GN/EGN/GGN), DNN models, or open-source tools like GNPy to estimate QoT/GSNR for candidate lightpaths. This integration allows the DRL agent to make informed decisions about path, band, and spectrum selection that explicitly account for QoT.

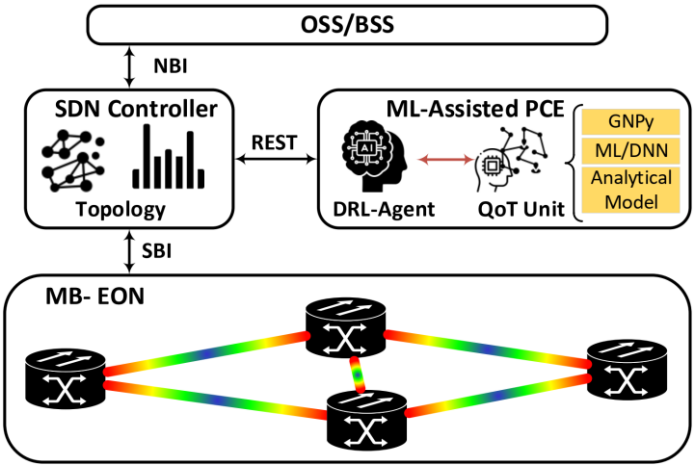


Figure 4.37: Architecture of the QoT-aware dynamic RMBSA in MB-EONs.

Provisioning Workflow

The following steps, shown in Figure 4.38, outline the end-to-end workflow for QoT aware service provisioning in a MB-EON using a DRL-enabled framework:

1. An incoming traffic request (source, destination, bit rate) is received by the SDN controller.
2. The controller delegates the RMBSA task to the ML-assisted PCE via a TAPI implementation.
3. The PCE generates a state representation by retrieving network information (topology, link spectrum status) from the SDN controller and querying the QoT estimation unit for QoT (e.g., GSNR or OSNR) for potential candidate paths.

4. This enriched state, augmented with QoT-level information, is fed to the DRL agent.
5. Based on its learned policy, the agent selects an action (route, band, and contiguous spectrum slots).
6. The QoT estimation unit validates if the chosen lightpath meets the required QoT threshold and the decision is communicated back to the SDN controller.
7. The controller then configures the relevant network elements (ROADMs, transceivers).

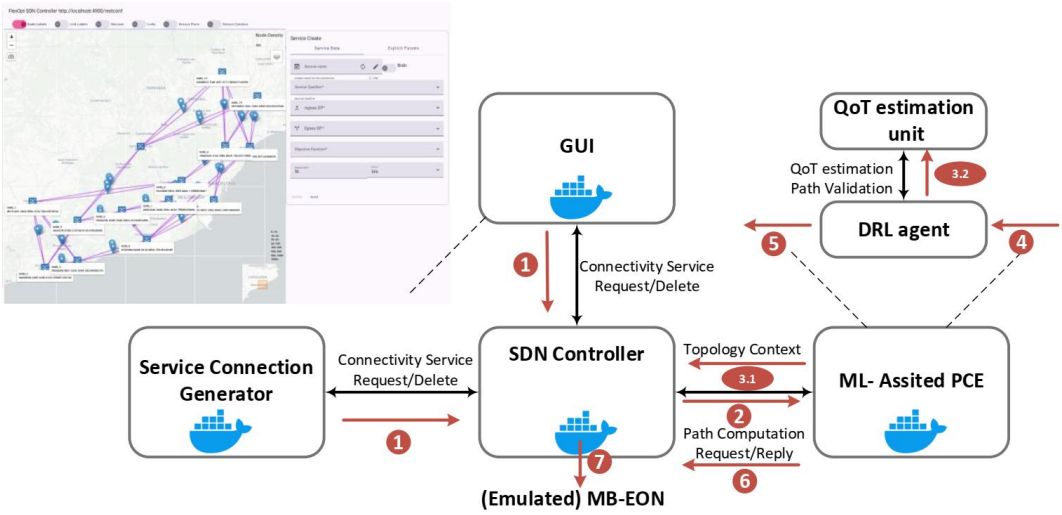


Figure 4.38: Provisioning Workflow for DRL-based QoT-aware RMBSA in MB-EONs.

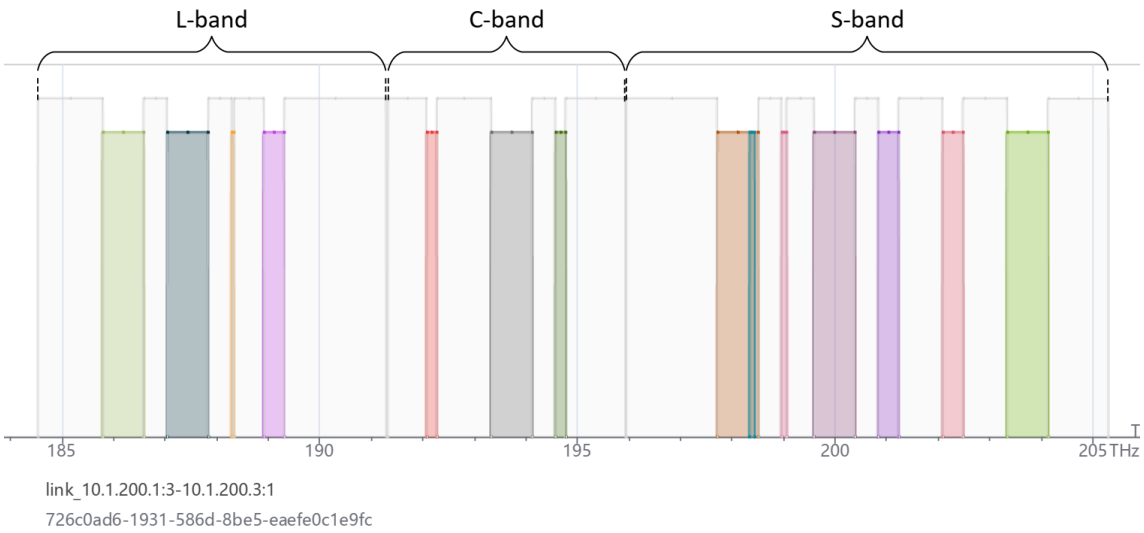


Figure 4.39 provides a visual representation of spectrum usage across the L, C, and S bands within the fiber links of the MB-EON. This illustration highlights the DRL agent's capability to effectively manage and allocate spectral resources. It showcases the system's ability to ensure

successful service provisioning while simultaneously maintaining the required QoT for lightpaths in a dynamic multiband environment.

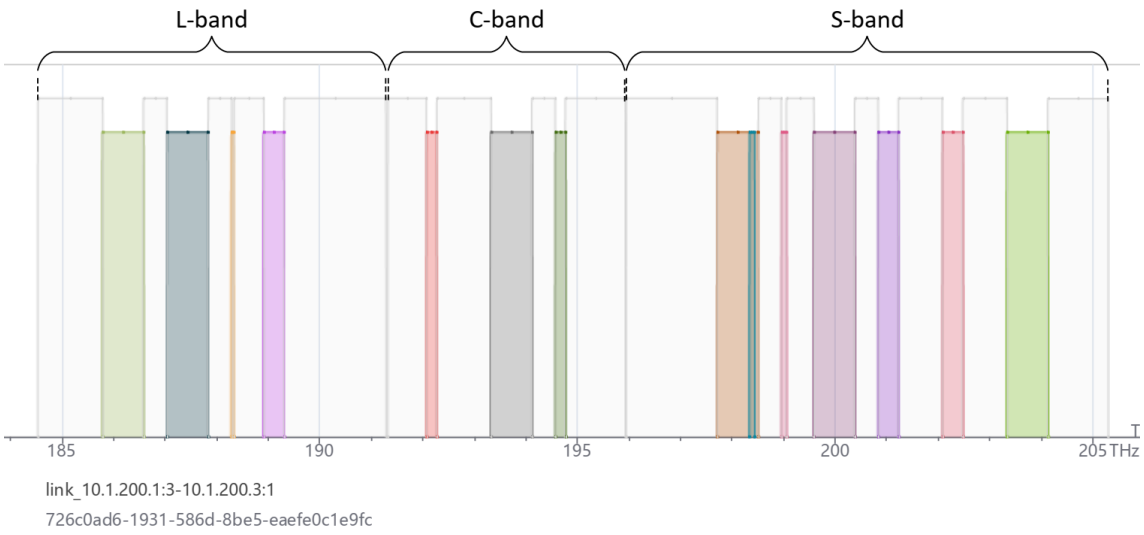


Figure 4.39: Spectrum Utilization in L+C+S-bands.

4.5.2.3 Concluding Remarks

This setup demonstrates a closed-loop control system where the SDN controller provides network abstraction and execution, and the ML-assisted PCE provides QoT-aware RMBSA decision-making specifically trained for a multiband environment. This setup provides a tangible illustration of how AI/ML can contribute to the automation and optimization of future high-capacity optical networks, paving the way for more efficient and reliable service delivery in dynamic multiband environments. This work has been published in [Her25-1].

5 TELEMETRY SYSTEM

5.1 TELEMETRY SYSTEM BASED ON MQTT

In Section 4.1 of [D42], the development and initial validation of an MQTT-based telemetry system was reported. This system was integrated with the CTTC FlexOpt SDN controller and AI/ML Path Computation Server into a proof-of-concept.

5.1.1 Telemetry System for Network Topology Management

The Telemetry System leverages an event-driven approach where SDN controllers generate events corresponding to specific changes in network configurations. This means only information about updated network elements is transmitted, significantly reducing data exchange. This event-driven nature ensures updates are transmitted immediately, providing a real-time view of network configurations, and focusing on specific changes allows for more efficient data processing and analysis, leading to quicker responses.

5.1.2 System Architecture

Figure 5.1 illustrates the integrated architecture of the telemetry system based on MQTT, integrating the CTTC FlexOpt SDN Controller, a lightweight MQTT subscriber and an external AI/ML Path Computation Server.

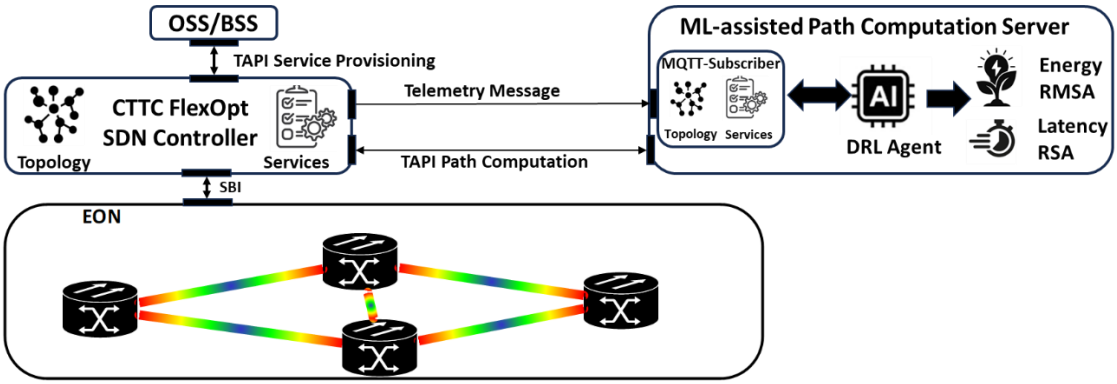


Figure 5.1: MQTT-based Telemetry.

CTTC FlexOpt SDN Controller Integration

The CTTC FlexOpt SDN controller is capable of exporting telemetry data using MQTT for simple topology synchronization. For instance, when CTTC FlexOpt SDN controller creates or deletes connections, it generates specific events describing changes like new connection establishments

affecting nodes, links and spectrum resources. This event data is transmitted to telemetry subscribers, who monitor and maintain an updated status of the network topology, network elements (ROADM) and resources (spectrum), based solely on these telemetry messages.

MQTT Subscriber Implementation

The MQTT subscriber takes advantage of the controller's MQTT export capability. The subscriber's primary function is to process received MQTT messages from the controller. It uses telemetry messages to build and dynamically update a network topology database (TED) and database of active connectivity services, providing the control plane with a more holistic view for optimizing network performance and resource allocation. **T**he MQTT subscriber has been successfully developed to process telemetry messages exported from the CTTC FlexOpt SDN controller. The subscriber processes messages based on their content, including LINK, NODE, NEP, CONNECTION, and CONNECTIVITY_SERVICE records (see Figure 5.2), to build and maintain the updated topology and list of active services.

```
09671717 UTC,8ddcc1a9-2286-5486-876e-0c6e2317557a,RECORD_TYPE_CREATE_UPDATE,2025-05-24 18:13:23.609681667 UTC,LINK
10204379 UTC,8282c0d1-9726-586e-b405-5cb9509dba85,RECORD_TYPE_CREATE_UPDATE,2025-05-24 18:13:23.610210059 UTC,NEP
12061837 UTC,5c9809bc-33a9-5214-ab03-b256457b4b46,RECORD_TYPE_CREATE_UPDATE,2025-05-24 18:13:23.612068177 UTC,NODE
13684743 UTC,d5c8aa78-c102-4f59-b9dc-f8d8112e7b01,RECORD_TYPE_CREATE_UPDATE,2025-05-24 18:13:23.613690903 UTC,CONNECTIVITY_SERVICE
14553334 UTC,f2ba98ca-986b-5e6a-b449-b3e5e84ad482,RECORD_TYPE_CREATE_UPDATE,2025-05-24 18:13:23.614559014 UTC,CONNECTION
14684260 UTC,d553c4be-6416-5895-8636-b8ee7a2c3fcf,RECORD_TYPE_CREATE_UPDATE,2025-05-24 18:13:23.614690820 UTC,CONNECTION
14763037 UTC,94e3e8ca-c836-5fe0-93ed-ab90f8b1c89a,RECORD_TYPE_CREATE_UPDATE,2025-05-24 18:13:23.614770067 UTC,CONNECTION
```

Figure 5.2: Telemetry Messages.

Integration with AI/ML Path Computation Server

The dynamically updated TED is then exposed to an AI/ML Path Computation Server. This Path Computation Server utilizes TAPI models for path computation. A key aspect of this implementation is that the path computation server does not retrieve the full TAPI context from the SDN controller through synchronous polling. Instead, it uses the asynchronously updated TED to maintain an up-to-date view of the network, leading to more accurate and effective path computation decisions. Figure 5.3 shows that the MQQT Subscriber provides the same topology information as the SDN Controller.

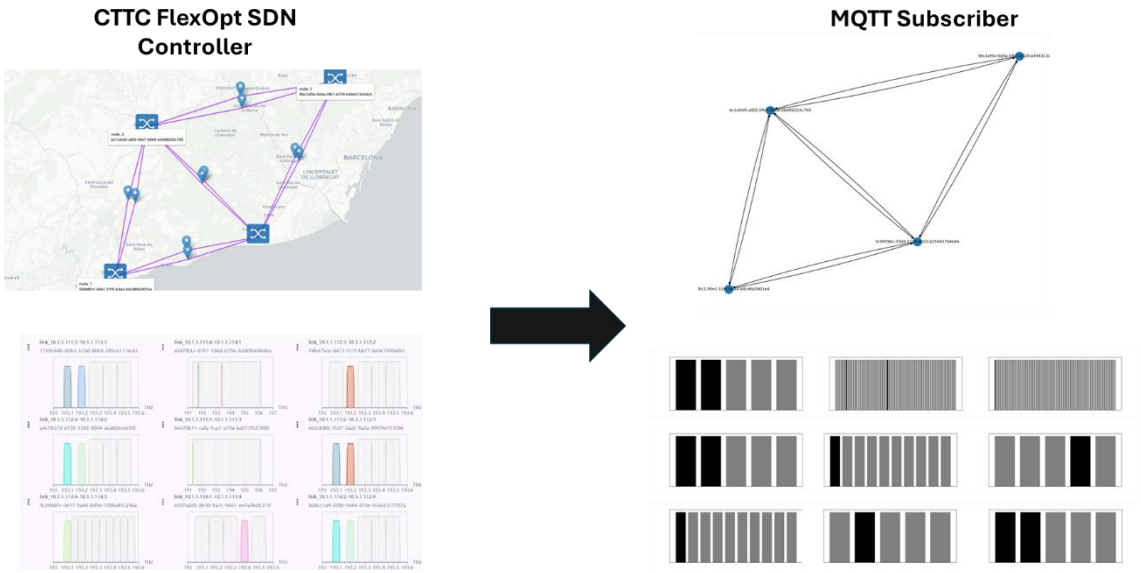


Figure 5.3: Real-time TED updating.

5.1.3 Concluding Remarks

The implemented MQTT-based telemetry system successfully provides a real-time, event-driven mechanism for network monitoring and topology synchronization. It effectively reduces data exchange by transmitting only changes, rather than full network snapshots. The system facilitates dynamic updates of the network topology, which is crucial for AI/ML-based path computation servers to make accurate and effective decisions without synchronous polling. This approach demonstrates a robust solution for maintaining an accurate and up-to-date view of the network topology and optical resources. This work has been published in [Her25].

5.2 Leveraging LLMs for Telemetry Data Analysis

Modern optical networks are fundamental for delivering high-bandwidth services and low-latency communications, demanding exceptional performance and reliability. Effective monitoring, particularly through telemetry, is essential for managing complex optical networks, which often comprise disaggregated and multi-vendor components. Telemetry provides real-time data from various observation points, offering critical insights into network behavior and performance. This real-time visibility enables operators to proactively identify potential issues, optimize resource allocation, and dynamically respond to changing conditions, including the detection of subtle performance degradations or "soft failures" that traditional methods might miss.

Large Language Models (LLMs) have significantly advanced fields such as data analysis and natural language processing due to their capability to process vast amounts of textual data, answer complex questions, and even generate code. However, LLMs can lack current or specific information, a limitation addressed by Retrieval Augmented Generation (RAG). RAG allows LLMs to access and utilize relevant information from external sources (e.g., real-time telemetry data) without requiring time-consuming and labor-intensive retraining. This capability is especially critical in network control management where real-time data and updates are continuously needed.

In parallel, process mining offers a potent technique for analyzing event data, such as that provided by telemetry, to understand and improve network processes. By applying process mining to telemetry data from optical networks, valuable insights into network behavior can be gained, enabling the identification of inefficiencies and opportunities for performance optimization.

5.2.1 Integrated LLM, RAG, and Process Mining Framework

The integration of LLMs with RAG and process mining tools (e.g., PM4PY) offers several compelling advantages for analyzing telemetry data in optical networks. This combination facilitates improved autonomous network control and management.

Key benefits include:

- *Improved LLM Accuracy:* RAG enhances the accuracy of LLM responses by providing access to external, current, and relevant information, ensuring LLMs have the necessary context for correct answers.
- *Privacy Preservation through Abstraction:* A significant benefit is the abstraction capability provided by process mining tools, which helps preserve privacy. Due to the inherent limitations of LLM context windows (which cannot handle the massive volume and complexity of raw event logs), abstraction techniques condense this information into more manageable formats. This allows LLMs to act as intermediaries, translating natural language queries from users into SQL queries that can be executed against telemetry data without directly exposing raw, sensitive network information.
- *User-Friendly Interfaces for Non-Technical Users:* This integration enables the development of intuitive interfaces where users can ask questions in natural language, eliminating the need for specialized knowledge of complex query languages or process mining concepts. This democratizes access to valuable network insights, allowing a broader range of personnel to contribute to network management and optimization.
- *Efficient Data Summarization and Pattern Identification:* Process mining tools excel at processing large quantities of telemetry messages generated by modern optical

networks. They effectively summarize data, identify patterns, and generate hypotheses for further investigation. For instance, a process mining tool combined with an LLM can analyze telemetry data for anomaly detection or to identify patterns indicating an emerging traffic bottleneck. The results can then be used to analyze specific processes and network paths contributing to issues, potentially triggering automated actions to mitigate problems.

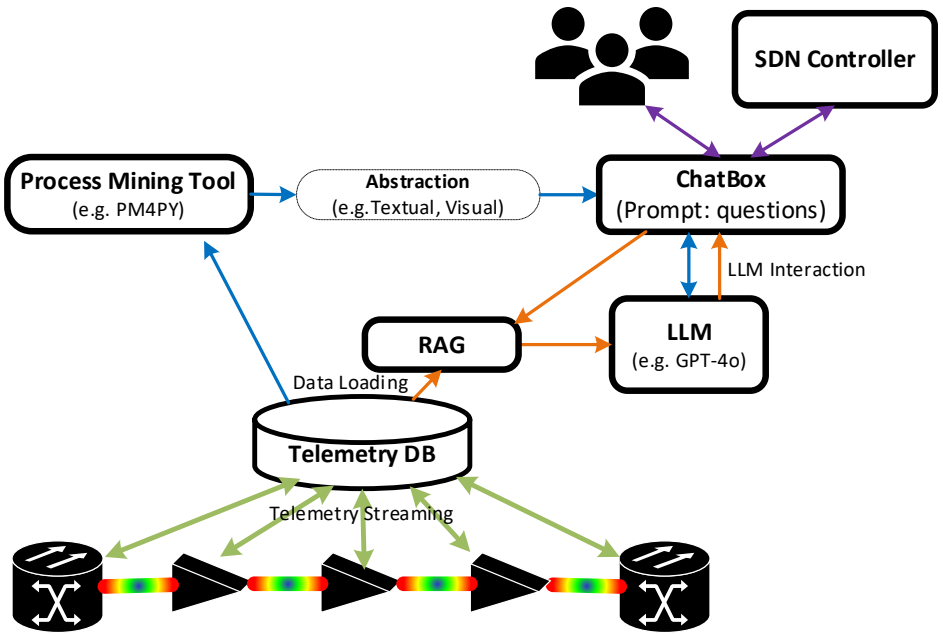


Figure 5.4: Integration of LLMs, RAG, process mining and telemetry data.

The proposed framework, depicted in Figure 5.4, shows how LLMs can analyze and comprehend updated and complex telemetry data, providing actionable insights for optical network management. A primary objective is to showcase the use of RAG to retrieve relevant information from external sources for generating LLM responses. It also shows the value of process mining for structuring and analyzing raw telemetry data. This highlights the importance of abstraction techniques to bridge the gap between vast telemetry data and LLMs' limited context windows. The system empowers network operators to identify performance bottlenecks, potential failures, and optimization opportunities.

For instance, this system can use a telemetry dataset containing specific information on OSNR (Optical Signal-to-Noise Ratio) and BER (Bit Error Rate) measurements at optical transceivers, along with input and output power at intermediate amplifiers. It also includes a "Failure" field indicating entries recorded during simulated failure scenarios. Through an interactive ChatBox, users can query for information such as the BER, OSNR levels, or input/output power thresholds that lead an optical connectivity service to be considered failed, as shown in Figure 5.5. This work has been published in [Her25].

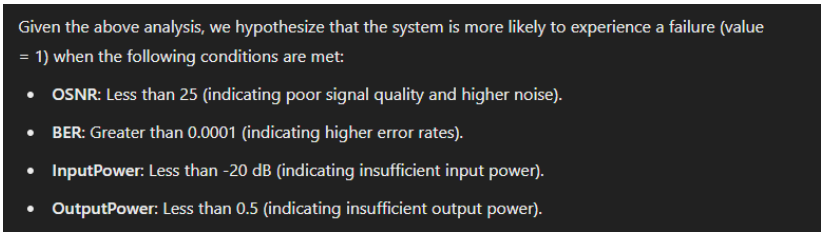


Figure 5.5: Chatbox analyzing telemetry data.

5.3 RIC TELEMETRY SYSTEM

Section 4.3 of D4.2, the previous version of this deliverable, gave an overview of the RIC telemetry system and its role in the O-RAN architecture. In this version of the Deliverable, the focus is on the actual telemetry that is available to support the SEASON use cases, with a particular focus on energy saving and NetDevOps.

The primary sources of RAN-specific telemetry in SEASON are the Distributed Unit (O-DU), and Central Unit (O-CU) and a summary of the telemetry they expose is presented here in Table 4 (O-DU Telemetry) and Table 5 (O-CU Telemetry).

5.3.1 O-DU Telemetry

Table 5-1: DU Telemetry in SEASON.

Counter Family	KPI Name	KPI	Description
Radio resource utilization	DL Total PRB Usage	RRU.PrbTotDl	This measurement provides the total usage (in percentage) of physical resource blocks (PRBs) on the downlink for any purpose.
	UL Total PRB Usage	RRU.PrbTotUl	This measurement provides the total usage (in percentage) of physical resource blocks (PRBs) on the uplink for any purpose.
	DL total available PRB	RRU.PrbAvailDl	This measurement provides the total number of physical resource blocks (PRBs) in average available downlink.
	UL total available PRB	RRU.PrbAvailUl	This measurement provides the total number of physical resource blocks (PRBs) available uplink.
UE throughput	Average DL UE throughput in gNB	DRB.UEThpDl	This measurement provides the average UE throughput in downlink. This measurement is intended for data bursts that are large enough to require transmissions to be split across multiple slots.
	Average UL UE throughput in gNB	DRB.UEThpUl	This measurement provides the average UE throughput in uplink. This measurement is intended for data bursts that are large

			enough to require transmissions to be split across multiple slots.
CQI related measurements	Wideband CQI distribution	CARR.WBCQIDist.BinX.BinY.BinZ	This measurement provides the distribution of Wideband CQI (Channel Quality Indicator) reported by UEs in the cell.
Packet Drop Rate	DL RLC SDU Packet Drop Rate in gNB-DU	DRB.RlcPacketDropRateDl	This measurement provides the fraction of RLC SDU packets which are dropped on the downlink, due to high traffic load, traffic management etc in the gNB-DU.
Packet delay	Average delay DL in gNB-DU	DRB.RlcSduDelayDl	This measurement provides the average (arithmetic mean) RLC SDU delay on the downlink within the gNB-DU, for initial transmission of all RLC packets.
Data Volume	DL Transmitted Data Volume	DRB.RlcSduTransmittedVolumeDL_Filter	This measurement provides the transmitted data volume in the downlink in a measurement time.
	UL Transmitted Data Volume	DRB.RlcSduTransmittedVolumeUL_Filter	This measurement provides the transmitted data volume in the uplink in a certain period.
Packet Success	UL Packet Success Rate	DRB.PacketSuccessRateUlgNBUu	UL Packet success rate for the DRB for a UE during measurement interval
Signal Quality	RSRP	RSRP	Average RSRP of the UE during measurement interval
	RSRQ	RSRQ	Average RSRQ of the UE during measurement interval
RLC Tx Rx Common Metrics	DU Id	du_id	DU id of the UE to which these measurement stats belong
	UE Id	ue_id	UE id of the UE to which these measurement stats belong
	DRB Id	drb_id	DRB id of the UE to which these measurement stats belong
RLC Tx Metrics	Number of SDUs	num_sdus	Number of SDUs transmitted during a measurement interval
	Number of SDU bytes	num_sdu_bytes	Number of SDU bytes transmitted during a measurement interval
	Number of dropped SDUs (due to full queue)	num_dropped_sdus	Number of dropped SDUs due to Queue full error during a measurement interval
	Number of discarded SDUs (instructed from higher layer)	num_discarded_sdus	Number of SDU discards during a measurement interval
	Total SDU latency	sum_sdu_latency_us	Total SDU latency in microseconds
	Number of transmitted PDUs	num_pdus_no_segmentation	Number of transmitted PDUs without segmentation during the measurement interval

	without segmentation		
	Number of transmitted PDU bytes without segmentation	num_pdu_bytes_no_segmentation	Number of transmitted PDU bytes without segmentation during the measurement interval
	RLC Mode	mode	DL RLC mode for the DRB
	Number of allocations that are too small to TX PDU	num_small_allocs	Number of allocations that are too small for a TX PDU
	Number of transmitted PDUs with segmentation	num_pdus_with_segmentation	Number of downlink PDU segments transmitted to a UE during measurement interval
	Number of transmitted PDU bytes with segmentation	num_pdu_bytes_with_segmentation	Number of downlink RLC PDU bytes transmitted to a UE during a measurement interval
RLC TM Mode Tx Metrics	Number of RETX'ed PDUs	num_retx_pdus	Number of downlink RLC PDUs retransmitted to a UE during a measurement interval
RLC UM Mode Tx Metrics	Number of RETX'ed PDUs bytes	num_retx_pdu_bytes	Number of downlink RLC PDU bytes retransmitted to a UE during a measurement interval
	Number of transmitted PDUs with segmentation	num_pdus_with_segmentation	Number of downlink RLC PDU segments sent for a UE during measurement interval
RLC AM Mode Tx Metrics	Number of transmitted PDU bytes with segmentation	num_pdu_bytes_with_segmentation	Number of downlink RLC PDU bytes sent for a UE during measurement interval
	Number of control PDUs	num_ctrl_pdus	Number of downlink control PDUs sent for a UE during measurement interval
	Number of control PDUs bytes	num_ctrl_pdu_bytes	Number of DL RLC control PDU bytes sent for a UE during measurement interval
	Number of SDUs	num_sdus	Number of UL RLC SDUs received for a UE during measurement interval
	Number of SDU bytes	num_sdu_bytes	Number of UL RLC PDU bytes received for a UE during measurement interval
	Number of PDUs	num_pdus	Number of UL RLC PDUs received for a UE during measurement interval
RLC Rx Metrics	Number of PDU bytes	num_pdu_bytes	Number of UL RLC PDU bytes received for a UE during measurement interval
	Number of dropped PDUs (reassembly timeout expiry or out of rx window)	num_lost_pdus	Number of UL RLC PDUs lost during measurement interval

	Number of malformed PDUs	num_malformed_pdus	Number of UL RLC malformed PDUs received during measurement interval
	RLC Mode	mode	UL RLC mode for a UE
	Number of SDU segments RX'ed	num_sdu_segments	Number of UL RLC SDUs received for a UE during measurement interval
	Number of SDU segments Bytes	num_sdu_segment_bytes	Number of UL RLC bytes received for a UE during measurement interval
	Number of SDU segments RX'ed	num_sdu_segments	Number of SDU segments received at RLC
RLC UM Mode Rx Metrics	Number of SDU segments bytes	num_sdu_segment_bytes	Num of SDU bytes received at RLC
	Number of control PDUs	num_ctrl_pdus	Number of control PDUs received at RLC
RLC AM Mode Rx Metrics	Number of control PDUs bytes	num_ctrl_pdu_bytes	Number of control PDU bytes received by RLC
	PCI	pci	PCI of the cell to which the UE is connected
	Number of PRBs	nof_prbs	Number of PRBs scheduled for the UE
	RNTI	rnti	RNTI of the UE
Scheduler Metrics	Channel Quality Indicator	cqi	Channel quality indication(CQI) of the UE
	Rank Indicator	ri	Rank indicator of the UE
	Downlink MCS	dl_mcs	Downlink MCS of the UE
	Downlink PRB Used	dl_prbs_used	Number of DL PRBs used by the UE
	Downlink Data Rate	dl_brate_kbps	Downlink data rate of the UE
	Downlink Data HARQ Acks	dl_nof_ok	Downlink HARQ ACKs received for the UE
	Downlink Data HARQ Nacks	dl_nof_nok	Downlink HARQ NACKs received for the UE
	Average PUSCH SNR	pusch_snr_db	Average PUSCH SINR detected for the UE
	Average PUSCH RSRP	pusch_rsrp_db	Average PUSCH RSRP for UE during measurement interval
	Average PUCCH SNR	pucch_snr_db	Average PUCCH SINR detected for the UE
	Uplink MCS	ul_mcs	Uplink MCS detected for the UE
	Uplink PRB Used	ul_prbs_used	Uplink PRBs used by the UE
	Uplink Data Rate	ul_brate_kbps	Uplink data rate of the UE
	Uplink Data HARQ Acks	ul_nof_ok	Uplink HARQ ACKs sent for the UE
	Uplink Data HARQ Nacks	ul_nof_nok	Uplink HARQ NACKs sent for the UE
	Uplink Buffer Status Report	bsr	Uplink buffer status report for the UE
	Downlink buffer Status	dl_bs	Downlink Buffer status for the UE

	Last Timing Advance	last_ta	Timing advance computed for the UE
	Last PHR	last_phr	Power Headroom reported by the UE

5.3.2 O-CU Telemetry

Table 5-2: CU Telemetry in SEASON.

Counter Family	KPI Name	KPI	Description
RRC connection number	Mean number of RRC Connections	RRC.ConnMean	This measurement provides the mean number of users in RRC connected mode for each NR cell during each granularity period.
	Max number of RRC Connections	RRC.ConnMax	This measurement provides the maximum number of users in RRC connected mode for each NR cell during each granularity period.
PDU Session Management	Number of PDU Sessions requested to setup	SM.PDUSessionSetupReq	This measurement provides the number of PDU Sessions by the gNB.
	Number of PDU Sessions failed to setup	SM.PDUSessionSetupFail	This measurement provides the number of PDU Sessions failed to setup by the gNB. This measurement is split into subcounters per failure cause.
Mobility Management - Inter gNB HO	Number of requested legacy handover preparations	MM.HoPrepInterReq	This measurement provides the number of legacy handover preparations requested by the source gNB.
	Number of successful legacy handover preparations	MM.HoPrepInterSucc	This measurement provides the number of successful legacy handover preparations received by the source NR cell CU.
Intra gNB HO	Number of requested legacy handover executions	MM.HoExeIntraReq	This measurement provides the number of outgoing intra gNB legacy handover executions requested by the source NRCellCU.
	Number of successful legacy handover executions	MM.HoExeIntraSucc	This measurement provides the number of successful intra gNB legacy handover executions received by the source NRCellCU.

DRB related measurements	Number of DRBs attempted to setup	DRB.EstabAtt	This measurement provides the number of DRBs attempted to setup to support all requested QoS flows in the PDU sessions to be setup by the INITIAL CONTEXT SETUP REQUESTs, PDU SESSION RESOURCE SETUP REQUESTs and PDU SESSION RESOURCE MODIFY REQUEST message received by the gNB from AMF.
	Number of DRBs successfully setup	DRB.EstabSucc	This measurement provides the number of DRBs successfully setup to support all requested QoS flows in the PDU sessions to be setup by the INITIAL CONTEXT SETUP REQUESTs, PDU SESSION RESOURCE SETUP REQUESTs and PDU SESSION RESOURCE MODIFY REQUEST message received by the gNB from AMF.
	Number of released active DRBs	DRB.RelActNbr	This measurement provides the number of abnormally released DRBs that were active at the time of release.
RRC connection establishment related measurements	Attempted RRC connection establishments	RRC.ConnEstabAtt	This measurement provides the number of RRC connection establishment attempts for each establishment cause.
	Successful RRC connection establishments	RRC.ConnEstabSucc	This measurement provides the number of successful RRC establishments for each establishment cause.
Packet Loss Rate	UL PDCP SDU Loss Rate	DRB.PacketLossRateUl	This measurement provides the fraction of PDCP SDU packets which are not successfully received at gNB-CU-UP.
	UL F1-U Packet Loss Rate	DRB.F1UpacketLossRateUl	This measurement provides the fraction of PDCP SDU packets which are not successfully received at gNB-CU-UP.
Packet Drop Rate	DL PDCP SDU Drop rate in gNB-CU-UP	DRB.PdcpPacketDropRateDl	This measurement provides the fraction of PDCP SDU packets which are dropped on the downlink, due to high traffic load, traffic management etc in the gNB-CU-UP.

Packet delay	Average delay DL in CU-UP	DRB.PdcpSduDelayDl_Filter	This measurement provides the average (arithmetic mean) PDCP SDU delay on the downlink within the gNB-CU-UP, for all PDCP packets.
	Distribution of delay DL in CU-UP	DRB.PdcpSduDelayDlDist.Bin_Filter	This measurement provides the distribution of PDCP SDU delay on the downlink within the gNB-CU-UP, for all PDCP packets.
GTP Throughput	UL GTP Throughput	UL_GTP_THP	Measure of UL GTP data received during the reporting period in the observed cell. Data for all UEs (all PDU sessions) in a cell is aggregated
	DL GTP Throughput	DL_GTP_THP	Measure of DL GTP data received during the reporting period in the observed cell. Data for all UEs (all PDU sessions) in a cell is aggregated

6 KPI TRACKING

There are several project-wide Key Performance Indicators (KPI) that affect work being done with WP4. Macroscopically, the methodology to evaluate the target KPIs includes monitoring evaluation in lab in WP3, integration with WP4 and assessment in WP5. Within WP4, this involves the 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.

Currently, we have identified key KPIs and the baseline values, along with initial first assessments. The SEASON project maintains a table of KPIs, including KPI description, owner, assessment methodology, baseline values as well as status. Most of the KPIs will be evaluated in the demonstrators within WP5 although for some KPIs their assessment can be done by means of simulations or theoretical analysis.

The Table 6-1 provides a simplified list of KPIs that involve WP4. The final assessment will be concluded at the end of the project within WP5 activities.

Table 6-1: Key WP4 related KPIs and their status.

KPI ID	KPI DESCRIPTION	STATUS
2.3	Network connectivity service with creation time < 3 min combining control and data planes.	The KPI will be evaluated and reported after integration on components in WP5.
4.1	<1ms mobile user latency via coordinated resource allocation at optical access and mobile network for SDM-PON mid-hauled RU/DU as a result of SEASON's target integration against >5ms delay in non-integrated approach.	The investigation on the KPI is in progress and the intermediate results are available with 20ms end-to-end latency with a PON contribution of 0.6ms.
4.2	>50% contribution in energy saving via dynamic spatial channels aggregation and deactivation of unused transceivers at the OLT side basing on traffic conditions over total 70% energy saving	The preliminary results available in T2.3 (by simulations), and experimental results are expected in WP5.
6.2	>40% reduction of O/E/O conversions in edge-edge and edge-cloud communications by developing smart edges with high-speed coherent intelligent pluggables and by moving 5G functions closer to the cell sites	The work towards the KPI is progressing well. The RAN software stack fully developed by ACC in edge computing nodes at deployed in CNIT Lab. Integration with monitoring system and closed loop automation by UPC in progress. Integration with HW PTMP optical system in progress by

		CNIT and INF-G. OFC2024 submission expected.
6.3	Supporting traffic adaptation/monitoring at µs granularity on innovative HW-accelerated networking smartNICs/DPUs (including computing, networking and optical resources) for selected low-latency services.	The work towards achieving this KPI is progressing well. The RAN software stack fully developed by ACC in edge computing nodes at CNIT Lab. Integration with monitoring system and closed loop automation by UPC in progress. Integration with HW PTMP optical system in progress by CNIT and INF-G. OFC2024 submission expected.
7.1	Intelligent data aggregation to provide data compression ratio >90% without significant information loss.	The KPI is achieved and assessed/reported in: L. Velasco, P. González, and M. Ruiz, "Distributed Intelligence for Pervasive Optical Network Telemetry," IEEE/OPTICA Journal of Optical Communications and Networking (JOCN), vol. 15, pp. 676-686, 2023.
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).	These KPIs of obj 7 are achieved. The study was conducted and tested in WP4, and will be finally demonstrated in WP5.
7.3	Configuration of the MBoSDM node prototype agents for simple operations from the SDN control plane of O(100ms). *Can be achieved from a control plane perspective, validated in R.Casellas et al ICTON2025. When using the actual prototype this gets extended to O(5000ms) due to the configuration of the WSS hardware. This means that the KPI from the control plane perspective is achieved and latencies are mainly introduced by hardware configuration delays.*	Initial studies with results are reported in: Dynamic Spatial Aggregation for Energy-Efficient Passive Optical Networks, ONDM 2024 and Power and Spectral Savings in Metro-Aggregation Networks Exploiting Coherent Point-to-Multipoint Transceivers, ECOC 2024, (KPI 7.4) NetDevOps pipeline is established for infrastructure configurations and service configuration. The results are evaluated and reported in ONDM 2025.
7.4	Demonstration of CI/CD approach with < 10 minutes pipeline execution time after committing or detection of network change (new node/pluggable, LOS, …), including device configuration modification, testing, validating, and deploying of infrastructure configuration compared to ~ 30 minutes of manual configuration via CLI or SDN intent creation, execution and testing using REST-APIs.	

8.1a	Autonomous operation based on multi-agent systems to reduce >25% OpEx w.r.t. manual/static operation (WP2, T2.2)	The KPI achievement is partially addressed in H. Shakespear-Miles, Q. Lin, S. Barzegar, M. Ruiz, X. Chen, L. Velasco, "Centralized and Distributed Approaches to Control Optical Point-to-Multipoint Systems Near-Real-Time," IEEE/OPTICA Journal of Optical Communications and Networking (JOCN), vol. 16, pp. 565-576, 2024.
8.1b		Functional description under definition in T2.3, experimental validation in WP5 (not started).
8.2	Near-real time local (inside a node) control loops, including data collection, analysis, and decision making in <10 ms, as compared to seconds for control loops involving the SDN controller. (WP5, WEST Demo)	The KPI is achieved in WP% final demo, and addressed in H. Shakespear-Miles, Q. Lin, S. Barzegar, M. Ruiz, X. Chen, L. Velasco, "Centralized and Distributed Approaches to Control Optical Point-to-Multipoint Systems Near-Real-Time," IEEE/OPTICA Journal of Optical Communications and Networking (JOCN), vol. 16, pp. 565-576, 2024.
8.3	Optical layer digital twin for gradual soft-failure detection and localization with at least 1min before major impact on the service. >90% accuracy in soft-failure identification. (WP5)	The KPI is achieved and is reported in M. Devigili, M. Ruiz, N. Costa, C. Castro, A. Napoli, J. Pedro, and L. Velasco, "Applications of the OCATA Time Domain Digital Twin: from QoT Estimation to Failure Management," IEEE/OPTICA Journal of Optical Communications and Networking (JOCN), vol. 16, pp. 221-232, 2024.
8.4	Service creation < 90 min (such as ETSI Network Service or Micro-services application deployed with Kubernetes)	The KPI will be evaluated and reported after the integration of components in WP5 final demo.

7 CONCLUSIONS

Deliverable D4.3 presents the final version of the SEASON control plane architecture. It builds upon the work reported in earlier deliverables D4.1 and D4.2, which covered the initial design, implementation, and intermediate results of the control plane infrastructure. This report includes updates and evaluation results for each network segment in the SEASON architecture, specifically for the RAN, PON, IPoWDM, MBoSDM, and optical transport domains. Each component's final version is described along with its evaluation details.

The deliverable also documents the software components developed in WP4. These include telemetry and monitoring components in different network segments for monitoring data in RAN, IPoWDM, MBoSDM, and optical transport. These tools improve data plane visibility using standard APIs and YANG models, making AI/ML-based control more accessible. Intelligent data aggregation and fine-grained monitoring are addressed, contributing to KPIs 6.3 and 7.1, as reported in Section 6.

SDN agents have been developed to provide standard interfaces between the data plane and control plane, enabling multi-domain control and orchestration. Each domain is managed by dedicated controllers that support service and device configuration. These controllers are designed to fully understand domain-specific functionalities, helping identify and address gaps in network control. using NetDevOps CI/CD pipelines. The control of MBoSDM node prototype agents allows simple operations to be executed in around 100ms, meeting KPI 7.3. Algorithms supporting network operation, planning, and control are also included. Some KPIs targeted in WP4 will be evaluated in WP5 (e.g., KPI 8.4), as they depend on the real testbed and integration of network segments.

The deliverable also reported on the development of APIs to integrate all the network segments RAN, PON, IPoWDM, MBoSDM, and optical transport, and control and orchestrate using the components developed in WP4. Details on the integration of domain controllers and the network orchestrator are provided in their respective sections. The demonstration of how the network orchestrator controls different domain controllers will be presented in WP5. KPIs related to the assessment of the end-to-end service provisioning in SEASON control plane architecture will be evaluated and reported in WP5. KPI tracking has been continuously performed to monitor progress and achievements. The latest updates are maintained and summarized in the Table 6-1from Section 6.

8 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*
EON	*Elastic Optical Networks*
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	*IP over Wavelength Division Multiplexing*
ITU-T	*International Telecommunications Union – Telecommunications sector*
KPI	*Key Performance Indicator*
LSTM	*Long Short-Term Memory*
LTE	*Long Term Evolution*
MAS	*Multi-Agent System*
MB	*MultiBand*
MBoSDM	*MultiBand 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*

9 REFERENCES

[Cug24] F. Cugini, R. Abu Bakar, A Sgambelluri, N. Sambo, L. De Marinis, A. Giorgetti, P. Castoldi, J. J. Vegas Olmos, F. Paolucci, “Data Processing Unit (DPU) and P4 Programmability in Support of the Edge Continuum”, ECOC 2024

[D41] SEASON D4.1, “First design and implementation of control plane infrastructure", Mar. 2024

[D42] SEASON D4.2, “Second year design of control plane infrastructure”, Dec. 2024

[Gum22] A. Gumaste, M. Sosa, H. Bock, and P. Kandappan, “Optimized ip-over-wdm core networks using zr+ and flexible muxponders for 400 gb/s and beyond,” J. Opt. Commun. Netw. 14 (2022).

[Her23] J. A. Hernandez, F. Arpanaei, G. Martínez, O. G. de Dios, J. P. Fernández-Palacios, and A. Napoli, “Clustering-based dynamic bandwidth allocation for point-to-multipoint coherent optics,” in Optical Fiber Communication Conference (OFC) 2023, p. Tu2D.5, Optica Publishing Group, 2023.

[Her25] C. Hernández-Chulde, R. Martínez and R. Casellas, " Evaluation of Integrating LLMs with Telemetry Data from Optical Networks," *2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, Barcelona, Spain, 2025

[Her25-1] C. Hernández-Chulde, R. Casellas, R. Martínez, R. Vilalta and R. Muñoz, " Demonstration of Autonomous DRL-Based QoT-Aware Provisioning in Multi-Band EONs," *2025 25th International Conference on Transparent Optical Networks (ICTON)*, Barcelona, Spain, 2025

[Sga25] Andrea Sgambelluri, Piero Castoldi, Pablo Armigol Robles, Oscar Gonzalez de Dios, Ghulam Bahoo, Filippo Cugini, "Provisioning of L2VPN services in IPoWDM networks", ICTON Conf. 2025

[Zha22] Q. Zhang, A. Morea, and M. Tornatore, “A pragmatic power-consumption analysis for ipowdm networks with zr/zr+ modules,” in European Conference on Optical Communication, (2022), p. We5.64

[Cas25] R. Casellas, et al., “Dynamic Service Provisioning in Multi-Band over Spatial Division Multiplexing Optical Networks” ICTON 2025 Barcelona, July 6-12, 2025.