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 D4.2

Second year design of control plane infrastructure

Editor Ramon Casellas (CTTC)

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

Version 1.0

Date December 16, 2024

Distribution PU

DISCLAIMER

REVISION HISTORY

Revision	Date	Responsible	Comment
0.1	*October 2, 2024*	*CTTC*	*Initial ToC version.*
0.2	*November 6, 2024*	*CTTC*	*First set of contributions.*
0.3	*November 27, 2024*	*CTTC*	*Final Contributions.*
0.5	*November 28, 2024*	*CTTC*	*Start Quality Review*
0.6	*December 12, 2024*	*INF-P, C. Pinho*	*Quality check*
1.0	*December 16, 2024*	*CTTC, R. Casellas*	*Final version for submission.*

LIST OF AUTHORS

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

EXECUTIVE SUMMARY

This deliverable reports an update in the design and implementation of the SEASON control plane infrastructure. In particular, it contains the first results of all SEASON WP4 tasks, reporting on monitoring/telemetry, single and multi-domain control solutions, and artificial intelligence / machine learning (AI/ML)-based self-management. The main objective of this report is dual: to identify and characterize the key functional components of the SEASON control plane solution, their roles and responsibilities; and the interfaces that enable interworking across the different systems. This report has been organized as follows.

Section 2 presents a brief summary of the SEASON Control plane architecture. With regards to the initial designs that were reported in D4.1, several interworking and interfaces have been clarified and consolidated, focusing on the role of each component in the WP5 demonstrations. Consequently, the list of elements has been provided to WP5 for integration.

Section 3 is the first part of the deliverable that deals with the control and configuration of the optical data plane. It consists of four blocks, which map roughly to the different tasks that are part of WP4. The first reports of the Software Defined Networking (SDN) agents, which are the software entities that are executed collocated with the data plane devices and are responsible for mapping high level configurations coming from the centralized SDN controller to the underlying hardware and to configure it accordingly. Such agents include the agent for Data Processing Units (DPUs), agents for Internet Protocol over Wavelength Division Multiplexing (IPoWDM) scenarios using pluggables in which the ROADM layer has been suppressed, the agent for the Multiband over Space Division Multiplexing (MBoSDM) node prototype that has been developed in WP3 and the FlexTelemetry agent. The second block covers the SDN controllers and the different use cases and scenarios that have been identified. This corresponds to a “per domain controller” setting and we list the components such as the Optical Line System (OLS) Controller, responsible for provisioning optical channels across a Reconfigurable Optical Add/Drop Multiplexer (ROADM) network; the MBoSDM controller that is responsible for controlling a network and provisioning services across a dual-level network; the controller for OpenXR Point-to-Multipoint (P2MP) pluggables that has been integrated; the control of the Space Division Multiplexing (SDM) Passive Optical Network (PON) and, in view of the integration with the transport system, the Radio Access Network (RAN) Intelligent Controller (RIC). The third block addresses the multi-domain and service aspects, which characterize the “end-to-end” aspect of the SEASON solution and finally, the fourth block contains the operational, planning and resource allocation tools.

As part of the closed-loop control that enables autonomous networks, Section 4 covers the different telemetry systems that have been explored and their support to the use cases such as the usage of Digital Twins, optimizing energy efficiency or the novel adoption of DevOps to network control and management.

TABLE OF CONTENTS

1 INTRODUCTION

The previous SEASON D4.1 deliverable defined the main architectural blocks of the SEASON control plane solutions. WP4 proposed a Software Defined Networking (SDN) control plane architecture covering the RAN, access/metro, and core segments, in an over-arching hierarchical control with a hierarchical arrangement of controllers. The control plane components and applications drive towards: (a) automatic network configuration; (b) self-healing during failure; (c) secure access and control of the devices; and (d) optimal use of network resources to make the network truly self-managed. Key aspects of this control plane are the full support of innovative Multiband over SDM transmission and switching hardware (HW) solutions, the integration of RAN Intelligent Controller (RIC) and access/metro SDN Control, the applicability of new control paradigms based on Network Development Operations (NetDevOps) approaches jointly with Artificial Intelligence / Machine Learning (AI/ML) in support of network operation and network orchestration.

This report provides a consolidation of each identified software component, which is independently developed for integration via open interfaces. Each component is described in terms of functional roles, interfaces to implement (if applicable), specific per component KPIs, and preliminary validations and integrations.

The document is structured as follows. Section 2 presents a brief summary of the SEASON Control plane architecture. Section 3 focuses on the control and configuration of the optical data plane, including controllers, agents, orchestrators, and software tools. Section 4 covers the different telemetry systems that have been explored and their support to the use cases such as the usage of Digital Twins, optimizing energy efficiency or the novel adoption of DevOps to network control and management.

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

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

2.1 OVERALL SEASON SOLUTION – CONTROL PLANE

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

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

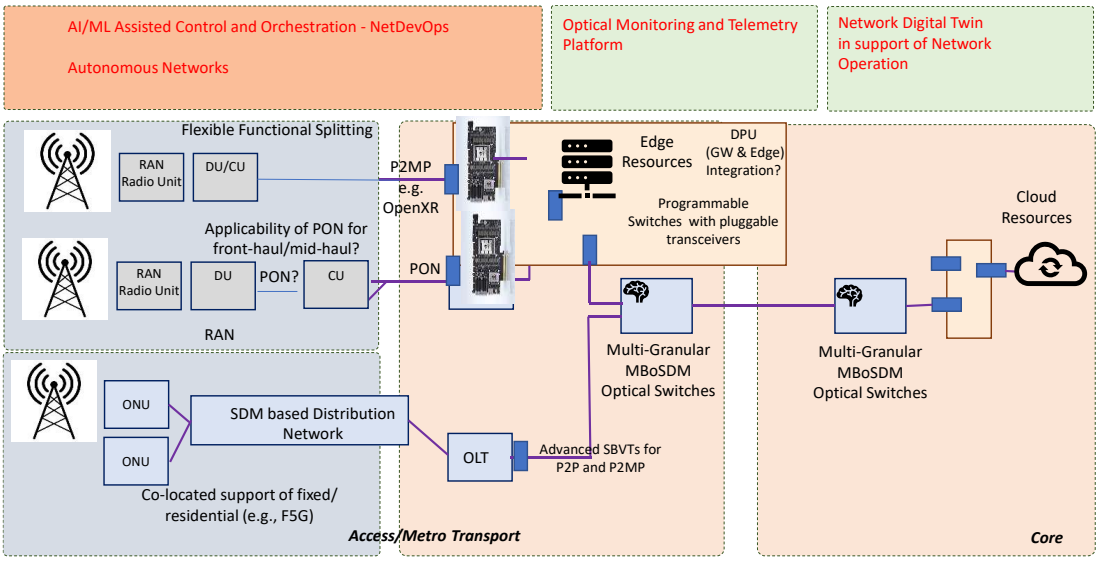


Figure 2.1-1: SEASON Architecture Self-managed and autonomous networking.

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

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

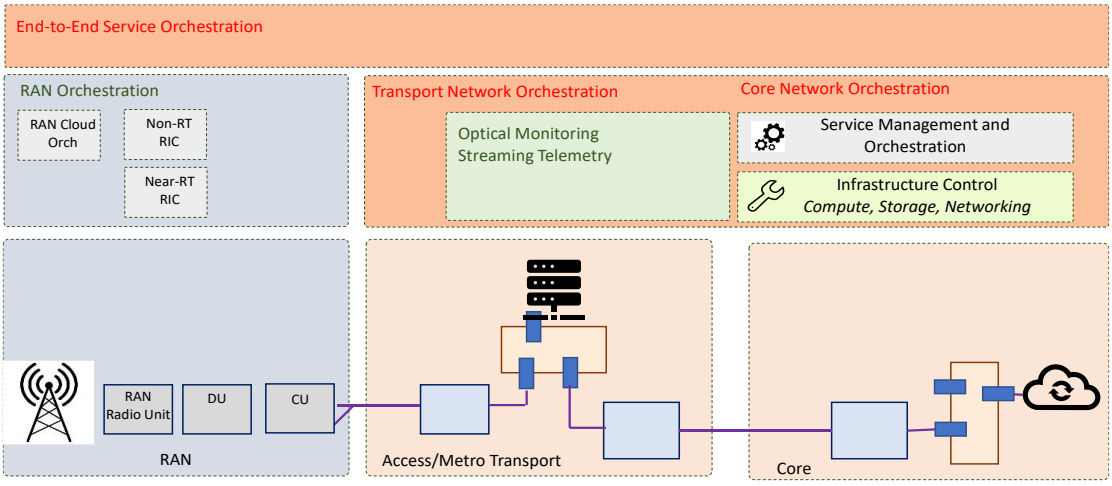


Figure 2.1-2: SEASON Architecture overarching control for RAN/PON/Backhaul network segments.

layer is linear, and the bias of the first hidden layer and output layer is zero. This set is used to generate an alternative disaggregated model η'. Although η is more accurate than η', the latter

exhibits good properties for security that allow easily hiding the concatenation of consecutive components by merging layers, thus obtaining a layered intra-domain model whose components cannot be isolated.

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

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

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 advances in transmission technologies and network programmability are driving the effective integration of optical, packet, and computing resources in support of edge-to-edge and edge-to-cloud continuum. At the transmission level, coherent pluggable transceivers in small form factors are boosting the development of IPoWDM switches effectively integrating optical and packet resources in a single network element, with remarkable benefits in terms of capital expenditures, latency, and power consumption. In addition, coherent pluggables have the potential to be inserted directly into latest generation high-speed Smart Network Interface Card (SmartNIC), also called Data Processing Unit (DPU), further reducing opto-electro-optical conversions. Both switches and DPU can also leverage network programmability and hardware acceleration of networking functions (e.g., using P4 or DOCA technologies, respectively).

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 this document, we focus on a comprehensive framework for pervasive telemetry and accelerated networking, suitable for effective integration of optical, packet, and computing resources, leveraging DPU programmability in support of edge-to-edge continuum.

Figure 3.1- shows the proposed scenario including edge computing resources equipped with DPU (E1-E3) and IPoWDM switches equipped with programmable ASIC (N1-N3). Both DPU and IPoWDM switch exploit coherent pluggable modules. This scenario assumes the overcoming of a few technological limitations, such as the hardware compatibility of DPU to host coherent transceivers. The integrated scenario enables the deployment of a scalable pervasive telemetry system, exploiting the offloading of telemetry packet processing to the programmable IPoWDM switches and DPUs. Furthermore, it provides the capability to rapidly react against critical events (e.g., failure or SLA degradations) by dynamically enforcing network reconfigurations, without requiring the intervention of SDN Controllers.

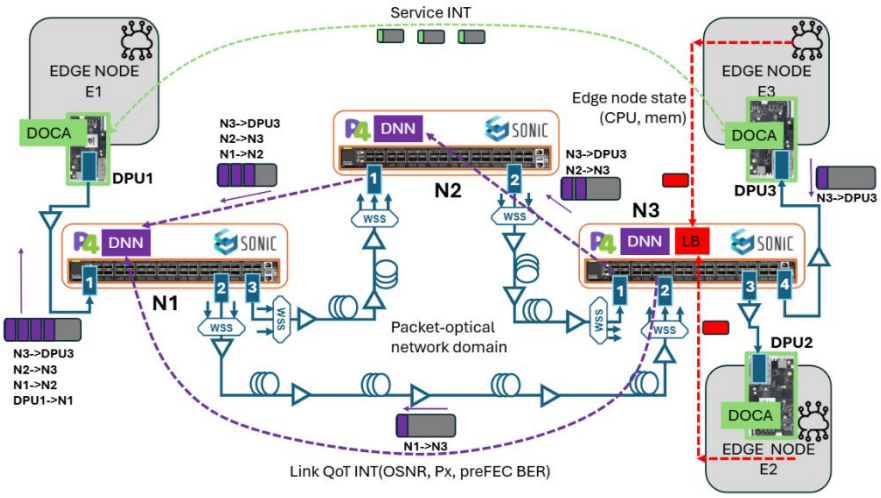


Figure 3.1-1: Programmable Packet optical network connecting Edge computing nodes.

Figure 3.1- shows a network of three IPoWDM nodes interconnected through optical line systems. The connectivity between N1 and N3 is assumed to be served by a pair of unidirectional lightpaths (for simplicity, only one shown in the figure). An alternative route is also available, passing through N2. Each node is aware of the Quality of Transmission (QoT) of the lightpaths terminated at its coherent receivers (e.g., received OSNR, pre-FEC BER, RX power). In addition, each node has access to the packet level Quality of Service (QoS) parameters (e.g., in/out packet/bit rates, queue occupation). For example, N1 monitors QoT and QoS of the optical connections originated at N2 and N3. That is, N1 can immediately detect a (soft) failure affecting N3-N1 link. However, with traditional SDN-based mechanisms, N1 has no visibility on possible QoT degradation affecting the lightpath in the reverse direction (N1-N3) or affecting non-directly connected links (e.g., N2-N3). Indeed, notification methods through a centralized SDN system might not be scalable and sufficiently fast to avoid data losses. In [Cug23] we introduced a pervasive decentralized telemetry system which relies on HW offloading. In particular, it exploits direct, in-network generation and processing of Multi-Layer Network Telemetry (MLNT) information exchanged among packet-optical nodes. MLNT packet reports provide detailed real-time performance (i.e., QoT and QoS) of those links that are of interest for a specific node. For example, N1 can receive telemetry data from N2 and N3, including the N2-N3 connection. Extracting information from the data plane to generate telemetry packet still requires software-level operation. On the other hand, the processing telemetry packets is performed in-network by the P4 ASIC, operating at wire speed within around 1 microsecond. In this framework, the SDN Controller only pre-computes the alternative route(s) to be enforced upon a QoT threshold violation for each active lightpath in the network. Each MLNT packet occupies between 60 and 100 bytes. Thus, the overall bandwidth consumed by MLNT packets depends on the generation rate. For example, 0.08 Mb/s is obtained in the case of a MLNT packet every 10ms. Even higher rates (e.g., 1 pck/ms) would still occupy less than 0.001% of a 100 Gb/s connection. In these measurements, we limited the forwarding of MLNT data to 1-hop distances.

In SEASON, we enhanced the telemetry and control system performed by intermediate transit IPoWDM by including the SmartNICs/DPUs at the source and destination nodes (e.g., E1-E3 in Figure 3.1-). Three DPUs, potentially equipped with coherent pluggable modules, are integrated into the system to handle the data processing tasks, ensuring high throughput and low latency required for real-time processing. We leverage the DOCA Flow API, which enhances the system's ability to manage network flows dynamically while maintaining the integrity and performance

of data processing. This way, all network elements can inject telemetry headers (e.g., including OSNR and pre-FEC BER within the UDP payload), enabling detailed monitoring at the packet level. The DOCA Flow API uses Deep Packet Inspection (DPI) accelerations available at the DPU hardware to inspect the packet payload of UDP packets containing telemetry headers from DPU3, DPU2, N3, N2, and N1. DPU1 extracts telemetry information from the payload of identified packets payload using DPI as well. The DPU1 then processes the extracted telemetry data and removes the telemetry headers. Cleaned packets are sent to the end host using the DOCA Flow API for efficient packet forwarding in hardware offload mode. DOCA Telemetry Service (DTS) supports telemetry related to end-to-end applications or DPU-related metadata (i.e., service INT as depicted in the figure). DTS monitoring counters, analyze the network traffic patterns, and ensures that the network interfaces are handling the expected load. DTS also provides error detection counters, packet drop counters, and CRC counters for monitoring. In addition, the state of DPU (e.g., the load of its internal ARM-based CPU cores) may be monitored and exchanged in a decentralized environment (e.g., in federated learning scenarios), allowing improved task distribution on the actual accelerated and computing resources.

The deep packet inspection performance of a 100G DPU have been tested considering specific telemetry payload matches. Results showed that hardware-accelerated DPI enables regular expression match at rates even higher than 30Gb/s, largely exceeding the requirements for pervasive telemetry systems over converged computing, packet, and optical infrastructures.

This work has been presented at the ECOC Conf. [Cug24].

3.1.2 SDN Agent for IPoWDM nodes in ROADM-free networks

In recent years, hyperscale cloud providers have driven extraordinary advancements in IP over WDM (IPoWDM) technologies, specifically coherent transceivers and packet-switching ASICs. This progress has arrived thanks to the massive increase in traffic growth exchange between data centers. Transceiver line rate is nearly doubling every two years, at a pace much higher than Telco traffic growth. Such evolution is driving Telco Operators to reconsider the way Metro networks are traditionally designed and implemented, assessing the so-called opaque or *ROADM-free* networks, which include only IPoWDM nodes without encompassing multi-degree ROADMs (see also Figure 3.1-). That is, no optical bypass is performed in intermediate nodes and optical connections are operated only on point-to-point links between IPoWDM nodes.

In specific selected network scenarios, ROADM-free solutions bring potential benefits including the drastic simplification of control procedures. Indeed, there is no need to execute complex routing and spectrum assignment algorithms for transparent mesh topology, complex impairment validation assessment, and complex equalization procedures handling transparent interconnections across multiple ROADMs. Furthermore, failure localization and recovery become easier and faster, since no multi-layer procedures are involved. In addition, the overall management procedures are performed in a single-layer by the consolidated IP/MPLS and Segment Routing technology, which can be handled without requiring a team of highly skilled optical experts. Potential drawbacks include additional power consumption due to processing of transit traffic, partially compensated by the reduced amount of equipment to be powered on. The additional latency due to electronic conversion and processing in transit nodes appears not relevant, since IPoWDM nodes equipped with 800G transceiver introduce an overall latency lower than 1ms, with negligible jitter.

In this work, we design and validate a novel control plane solution specifically designed for ROADM-free point-to-point IPoWDM networks. The simplified yet essential optical network

configurations are performed in a decentralized way by lightweight software modules, named Optical Line Controller (OLC), without relying on a centralized Optical SDN Controller.

ROADM-free network architecture

Traditionally, IPoWDM nodes have been interconnected using Reconfigurable Optical Add/Drop Multiplexers (ROADMs). A ROADM is built with several elements, mainly ROADMs on Blade (RoB) and Multi-Cast Switches (MCS). Each node degree consists of a RoB, which integrates a pair of Wavelength Selective Switches (WSS) and optical amplifiers, for handling both TX and RX side.

For example, Figure 3.1-(a) shows an IPoWDM node equipped with coherent transceivers that rely on a two-degree ROADM to connect to a different IPoWDM node. The ROADM is composed of two RoB (one for east and one for west sides). A third element, typically a MCS, is also included to serve as add/drop module. This overall multi-layer node architecture enables effective optical bypass in intermediate nodes. Indeed, electronic processing in such ROADM-based optical networks only occurs at the source and destination IPoWDM nodes. On the contrary, a ROADM-free network does not exploit optical bypass, always terminating the optical connection at the adjacent IPoWDM node. Network infrastructure can be designed according to three different models, depending on the amount of data to be exchanged and the distance between the nodes. In the first ROADM-free network model, the node architecture of Figure 3.1-(b) is adopted where only one coherent transceiver per direction is used for the interconnection to the adjacent IPoWDM node. That is, the transceiver is directly attached to the optical fiber. No optical amplification is adopted. This model is suitable for short metro distances (e.g., few tens of km, given the limited sensitivity of the transceivers, in the order of -10dBm) and relatively limited capacity (the one provided by a single transceiver, e.g. 400 or 800Gb/s). In case additional capacity is needed, management problems arise since the line needs to go offline to enable the insertion of coupler/splitters or RoB to serve additional transceivers. In the second model, suitable for larger metro distances (up to few hundreds of km), optical amplification is introduced. This model, still considering a single transceiver per direction (i.e., relatively limited capacity), may also lead to the same critical upgrade procedures. In the third model, suitable for larger capacity, multiple transceivers per direction are considered. To aggregate multiple signals in a single fiber, different cases are available: (i) *Fixed AWG*. Pros: low cost; low insertion loss; no upgrade issues, scales well to accommodate a large number of transceivers. Cons: rigidity in spectrum allocation due to fixed filtering; it may not be future proof with the increase of baud rate. (ii) *Splitters/Coupler.* Pros: Low cost; no filtering constraints. Cons: possible slight transmission issues due to non-filtered signals; insertion loss increases with the splitting ratio, making it suitable to practically accommodate only very few transceivers. Upgrade issues. (iii) *RoB* (shown in Figure 3.1-(c)). Pros: relatively low insertion loss; no upgrade issues, no rigidity in spectrum allocation; suitable to accommodate large number of transceivers; future proof with the increase of baud rate. Cons: higher cost. Each of the above models requires to be properly controlled and managed. Traditional solutions based on centralized Optical SDN controllers appear over-complicated for a set of independent optical links, also considering the complex interactions with the IP Controller (see open discussions in the MANTRA working group).

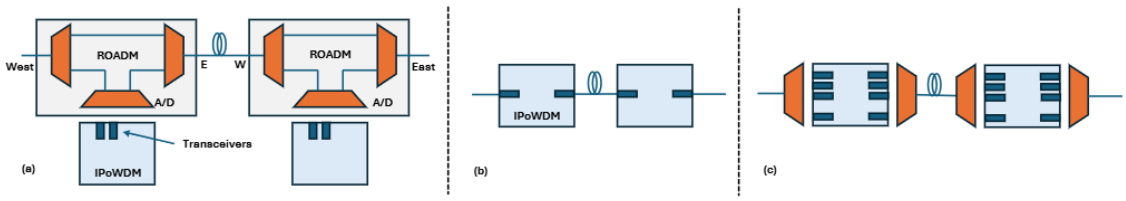
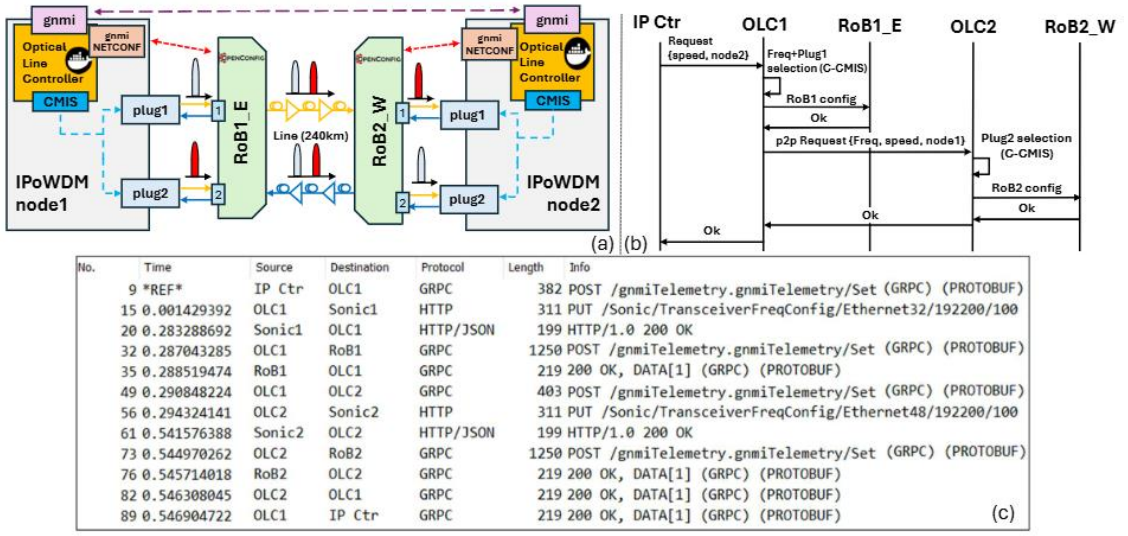


Figure 3.1-2: (a) Traditional multi-layer architecture with IPoWDM connected via ROADMs; (b) IPoWDM directly connected; (c) IPoWDM connected using multiple transceivers aggregated through RoB.

Proposed decentralized control Agent for ROADM-free networks

The *IP* SDN Controller is considered as the single entity controlling the network, directly configuring the IPoWDM nodes. The configuration and monitoring of optical links is instead delegated to lightweight Optical Line Controllers (OLCs). Each OLC has configuration rights to the transceiver parameters and to the optical elements of the controlled optical link(s). This may include, according to the considered ROADM-free model, WSS and amplifiers in a pair of RoB and in-line amplifiers. In this work, we assume one OLC instance having responsibility of one optical link, in the outgoing direction (i.e., TX side of the link). The OLC has knowledge of local details (including transceiver capabilities and pluggables/ROB port mapping. It is provided, by the IP Controller or management system, with the IP of the remote node to be connected. Each OLC discovers and configures both local transceiver parameters and pertaining RoB. Furthermore, it is capable of receiving monitoring data from all involved elements, including the RX parameters on the connected IPoWDM node. In our implementation, the OLC has been implemented as docker container installed in the IPoWDM node. It is provided with direct access via C-CMIS to the transceiver parameters, and it relies on OpenConfig YANG models via gNMI (or NETCONF) to configure and monitor the intermediate RoBs and amplifiers. Furthermore, the OLC runs a local computation engine performing impairment assessment, spectrum assignment, TX power computation and operational mode assignment for each optical signal activated by the IP Controller. Even if dedicated tools like GNPy or AI models could be adopted as computation engine, in our implementation, given the limited number of supported operational modes and the relatively short distances to cover between IPoWDM nodes (up to few hundreds km), the path computation engine mainly relies on look-up tables over distances, with dynamic feedback based on experienced Pre-FEC and RX power performance.



No.	Time	Source	Destination	Protocol	Length	Info
9	*REF*	IP Ctr	OLC1	GRPC	382	POST /gnmiTelemetry.gnmiTelemetry/Set (GRPC) (PROTOBUF)
15	0.001429392	OLC1	Sonic1	HTTP	311	PUT /Sonic/TransceiverFreqConfig/Ethernet32/192200/100
20	0.283288692	Sonic1	OLC1	HTTP/JSON	199	HTTP/1.0 200 OK
32	0.287043285	OLC1	RoB1	GRPC	1250	POST /gnmiTelemetry.gnmiTelemetry/Set (GRPC) (PROTOBUF)
35	0.288519474	RoB1	OLC1	GRPC	219	200 OK, DATA[1] (GRPC) (PROTOBUF)
49	0.290848224	OLC1	OLC2	GRPC	403	POST /gnmiTelemetry.gnmiTelemetry/Set (GRPC) (PROTOBUF)
56	0.294324141	OLC2	Sonic2	HTTP	311	PUT /Sonic/TransceiverFreqConfig/Ethernet48/192200/100
61	0.541576388	Sonic2	OLC2	HTTP/JSON	199	HTTP/1.0 200 OK
73	0.544970262	OLC2	RoB2	GRPC	1250	POST /gnmiTelemetry.gnmiTelemetry/Set (GRPC) (PROTOBUF)
76	0.545714018	RoB2	OLC2	GRPC	219	200 OK, DATA[1] (GRPC) (PROTOBUF)
82	0.546308045	OLC2	OLC1	GRPC	219	200 OK, DATA[1] (GRPC) (PROTOBUF)
89	0.546904722	OLC1	IP Ctr	GRPC	219	200 OK, DATA[1] (GRPC) (PROTOBUF)

(c)

Figure 3.1-3: (a) Testbed with multiple pluggables and RoBs; (b) Workflow; (c) Wireshark capture.

The proposed OLC solution has been preliminary implemented and validated on a network testbed including two IPoWDM Edgecore nodes, with SONiC Operating System, connected through a three-span optical link, as shown in Figure 3.1-. In the considered scenario, reproducing the most complex implementation of Figure 3.1-(c), IPoWDM nodes are interconnected through Lumentum RoBs (encompassing WSS and pre/booster per direction) across three spans (240km in total). Two inline amplifiers are used, operating in constant gain mode, to guarantee for each channel a launch power of 0dBm. Each IPoWDM node uses more than one transceiver (two of type 400ZR+ in the experiment). In each IPoWDM node a containerized OLC module has been on-boarded. The OLC has been equipped with three main plugins: (i) the *gNMI plugin*, enabling both North Bound (towards an IP controller or the Network Administrator) and peer-to-peer (for direct configuration/monitoring functionalities) interfaces; (ii) a REST-based *C-CMIS plugin* for the host port/pluggable configuration and (iii) a gNMI/NETCONF plugin, being able to perform the configuration of the pertaining RoB (the OpenConfig wavelength-router model has been adopted). In the experiment, gNMI communication to the RoBs is exploited.

Figure 3.1- (b)-(c) show respectively the proposed workflow and the related Wireshark capture. A connectivity request, including bitrate and remote node, is sent by the IP Controller to OLC of IPoWDM node 1 (OLC1), using the gNMI interface (message 9). Each OLC is aware of the local details including the pluggable capabilities and the port mapping among pluggable modules/RoB. OLC1 selects the frequency to be used for connectivity (First-Fit policy has been implemented, more advanced solution could be adopted). OLC1 performs the selection and configuration of both pluggables, via C-CMIS plugin (messages 15-20) and the local RoB (RoB1_E), using the gNMI channel (messages 32-35). Then, it triggers, via peer-to-peer gNMI, the OLC at IPoWDM node2 (OLC2), to perform the remote configuration (message 49). OLC2 is responsible to select and configure the local pluggable, via C-CMIS plugin (messages 56-61), and to perform the configuration of the related RoB (i.e., RoB2_W), via gNMI/NETCONF (messages 73-76), acknowledging OLC1 at the end (message 82). The setup of an optical connectivity among the two IPoWDM nodes has been repeated 20 times, collecting the control plane end-to-end setup time. Obtained results show an end-to-end control plane setup time in the order of 0.5s (min 506ms, max 570ms, avg 541ms), while data plane operation is established in around 60s.

To summarize, a novel Optical Line Controller (OLC) control plane module is proposed and preliminary validated on a ROADM-free network, where no optical bypass is performed and multi-degree ROADMs are not deployed. The OLC Agent successfully configures and monitors point-to-point optical links between IPoWDM nodes, without requiring complex centralized Optical SDN Controllers and related MANTRA Architectures.

3.1.3 SDN Agent for the MBoSDM node prototype

Description: The SDN Agent for the MBoSDM node prototype is the software agent/component that maps high level configuration operations from the SDN controller into the device configuration operations (typically at lower level) and provides a given device model abstraction. The SDN agent exports an Application Programming Interface (API) to higher layers enabling remote configuration and device programmability. The SDN agent allows the configuration of the data plane in terms of “cross-connections” that can happen at the level of optical spectrum or fiber/port switching.

APIs and Interfaces for integration: the APIs is based on a NETCONF/Yang interface with a device data model that abstracts the operations to be performed in the device.

Status and features under development: this component is in the early design stages and depends on the MBoSDM prototype that has been developed in WP3. Once the prototype has been consolidated,

Validation: no validation activities have taken place at the time of writing. Validation will be done in the context of WP5 activities.

3.1.4 FlexTelemetry Agent

Description: The FlexTelemetry agent is a Python-based telemetry application designed for real-time and historical data management, focusing on network devices such as Optical Transponders, Carrier Ethernet switches, and Optical Amplifiers. The application leverages NETCONF and SNMP protocols for device data collection, offering simultaneous streaming from multiple sources. The application supports plugin-based flexibility, allowing users to direct telemetry data to various time-series databases and message brokers, including Influx DB, Kafka, MQTT, and Redis (Figure 3.1-).

The application includes automated integration with Telegraf when using Kafka or MQTT as plugins. This integration enables the application to simultaneously provide real-time access to telemetry data—ideal for machine learning and analytics applications—while storing historical data in Influx DB for future reference and model training. By ensuring both instant data access and reliable data retention, the application provides a comprehensive solution for telecom and network operations, enabling faster insights, predictive analysis, and long-term data storage. Moreover, the Python application is optimized for scalability, supporting high-volume, simultaneous telemetry collection from multiple devices through multi process parallel programming.

A key feature of this application is its use of Python's multiprocessing capabilities, enabling it to handle multiple device connections simultaneously. Each device's data is processed in a separate process, which optimizes data parsing and enhances the application's ability to handle high-frequency telemetry streams. The multiprocessing design not only improves data handling speed but also ensures that the collected metrics are parsed efficiently and routed quickly to the selected northbound plugins, such as Kafka, MQTT, Redis, and Influx DB. This flexible plugin system allows the application to cater to both real-time and historical data needs, supporting immediate analytics as well as long-term data storage. Real-time data is monitored through Grafana, which subscribes to Influx DB, enabling dynamic visualization and analysis of telemetry metrics.

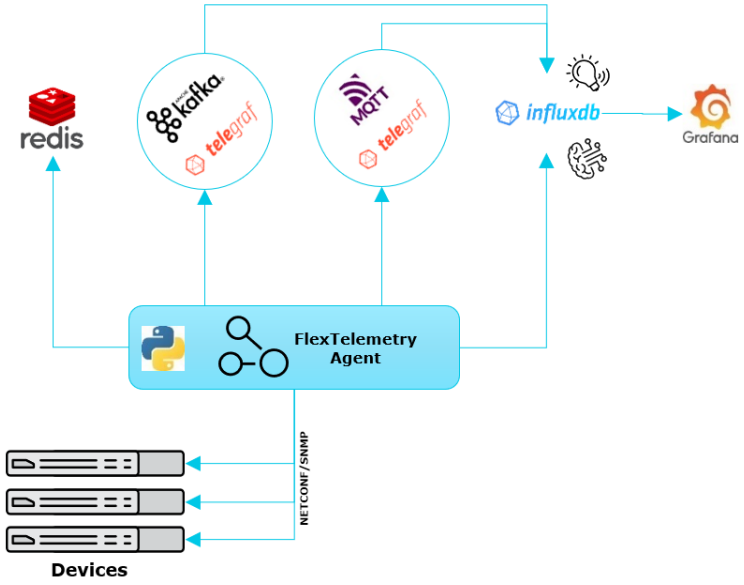


Figure 3.1-4: Flex Telemetry Agent application architecture.

APIs and Interfaces for Integration:

1) **Southbound device integration APIs:**
   a. **NETCONF API:** The FlexTelemetry agent uses NETCONF to establish connections with the optical transponders and Carrier ethernet switches. This API handles device authentication, session management, and data retrieval.
   b. **SNMP API:** For Optical Amplifiers, the application provides an SNMP API that facilitates data collection through standardized SNMP commands. This API supports SNMP polling and traps, allowing efficient data capture from devices that do not support NETCONF.
2) **Northbound Plugin Interfaces:**
   a. **Plugin API:** The application offers a flexible Plugin API, allowing it to connect with multiple data storage and messaging solutions, such as Kafka, MQTT, Redis, and Influx DB.
   b. **Telegraf Integration API:** When Kafka or MQTT is selected as a plugin, the application automatically deploys a Telegraf agent to capture data from the message broker and forward it to Influx DB.

Status:

- Future development plans include expanding support for additional Carrier Ethernet switches and Optical Transport Network (OTN) devices. Additionally, enhancing the plugin capabilities, including support for various types of message brokers and databases.
- The planned integration of gNMI-based streaming will further extend the application's compatibility, blending seamlessly into the current architecture to support a broader range of network data sources.

3.2 SDN CONTROLLERS

3.2.1 MBoSDM Optical Controller

3.2.1.1 Description

The steady traffic increase means that solutions based on exploiting additional bands will not suffice in the long term. Further capacity scaling means going parallel and exploiting the spatial dimension, including the usage of recently developed Sliceable Multidimensional (spectral and spatial) transceivers. Macroscopically, this can be divided into i) single level or flat networks and ii) multi-level networks.

In single level or flat optical networks, parallel links are deployed either as bundles of fibers (BoF) or, in a longer term as multi-core fibers with fan-in / fan-out systems. However, such SDM systems are mostly point-to-point and switching only happens at the photonic media / media channel (DWDM) level. In other words, the SDM or spatial layer is systematically terminated at each node and transported OTSi signals are switched (by switching their media channels). ROADM architectures based on WSS require, in such cases, many ports. A network with max node degree of 9 with bundles of 8 fibers may typically require WSS with ~70 ports (this is without considering add/drop ports). Considering node architectures such as non-spatial lane switching or limiting internal connectivity may reduce the number of required ports at the expense of reduced performance. The introduction of additional bands and the limitations of WSS technology render this problem even more complex. From the control plane perspective, control functions need to consider internal ROADM connectivity with e.g., WSS modelling e.g. by means of internal links between circuit pack ports, as in OpenROADM device data models.

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); Optical Band Switching (e.g., C-band switching), Waveband switching or Fiber / Core switching. At specific points, signals may be switched at a given level/layer. It is known that switching at the highest level (e.g. DWDM switching) yields better performance but introducing switching at the lowest (aggregated) layers may simplify node architectures and be more cost effective. From the control plane implications, multi-level networks are more complex to manage than single level networks, for the simple fact that the SDN control plane needs to address the additional associated complexity of controlling multiple levels. In this setting node modelling and abstraction are critical aspects to consider and the control plane needs to account for the existence of transitional links (from one client level or layer). SDN control can leverage the previous know-how in terms of control of multilayer networks. In Multi-Level Networks with hybrid (multigranular) nodes, network elements are able to terminate data links with different switching capabilities and are characterized by the ability to terminate and/or switch connections at different levels. In the optical domain, such

nodes are also referred to as muti-granular nodes. Cases of interest are nodes with dual DWDM (i.e., multiband) over fiber or core (SDM) switching. Such nodes present a classical arrangement with different switching (sub) matrices. Scalability / Efficiency or Feasibility constraints drive multi-stage configurations or hierarchical/coarse/multigranular nodes. From the control plane perspective, the question to address is what are the extensions (architectural, protocol or algorithmic) required to, potentially jointly, address wavelength (media channel), waveband and SDM switching. With a common deployment model of a single SDN controller per domain, further modeling and standardization work is required to efficiently reuse the model-driven development and existing frameworks for SDN control of MB over SDM networks. Extension to multi-level networks requires:

- Addressing virtual network topologies (VNT) at different levels, the management of logical (virtual) links supported by low layer connections and the overall formal description of additional protocol layers and layer protocol qualifiers (e.g., formally introducing the capability of “core switching” / fiber switching in a standard way).
- Characterize optical links at the SDM layer (e.g., identifying fiber indexes within a bundle of fibers or identifying cores/modes in a multi-core setting.
- Define propagation models and physical impairments extensions to account for aspects such as inter-core cross-talk
- Extend node capability and interconnection models and related abstractions, including transactional links.
- Devise new multi-layer algorithms that enable optical by-pass and grooming with the addition of constraints such as the core-continuity constraint, in which the path computation function and resource allocation must ensure that a given spatial path uses a single core index, analog to the spectrum continuity constraint of WDM networks.

In the following, we assume nodes with dual switching layers, as shown in the figure below.

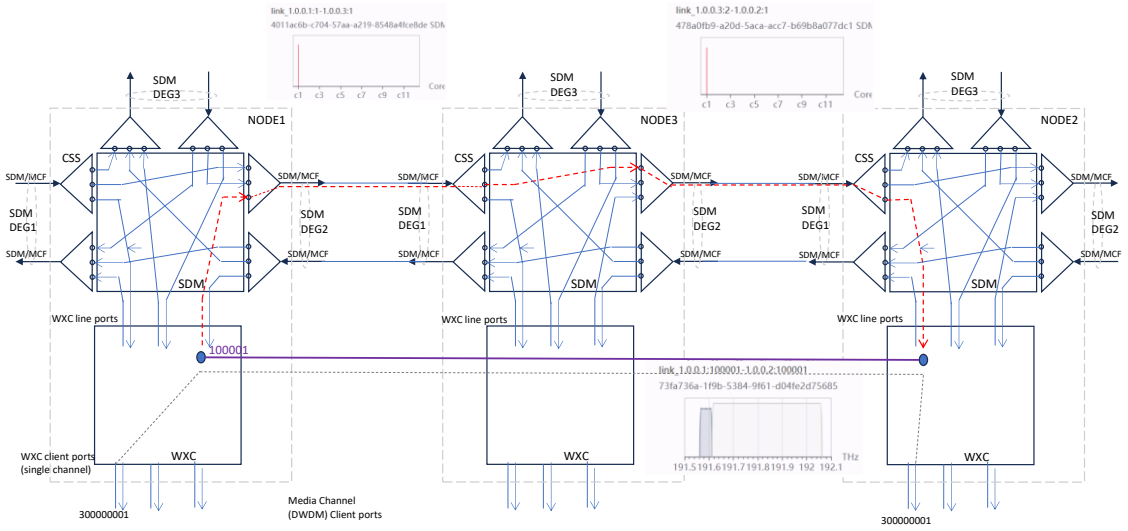


Figure 3.2-1: Multi-Stage optical Nodes (MBoSDM) showing DWDM and SDM switching.

Dissemination Level	PUB (Public)

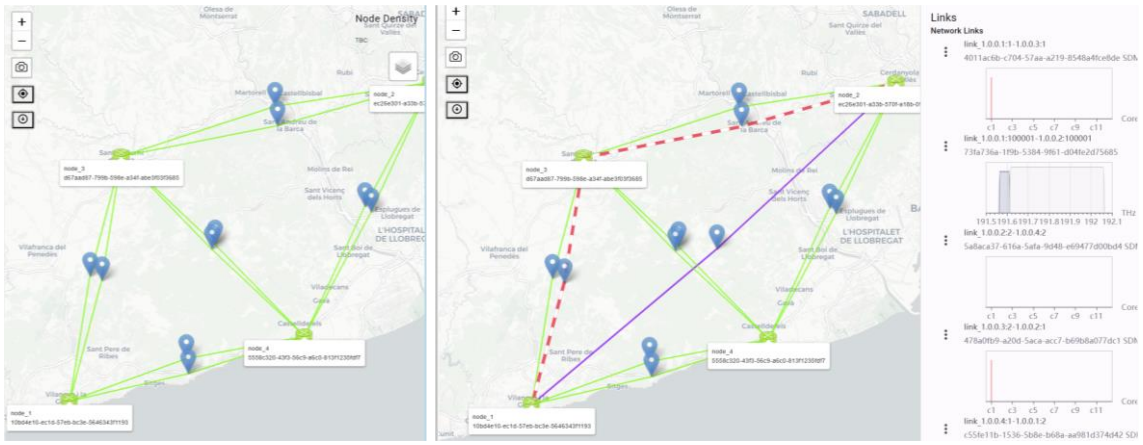


Figure 3.2-2: Concept of virtual link and aggregation. An SDM connection (red dash) uses a fiber core and the connection supports a new DWDM multiband link (purple link) between the SDM connection endpoints for services.

3.2.1.2 APIs and Interfaces for Integration

S.No	Interface	Description
1	RESTCONF TAPI Models	The FlexOpt SDN controller exports the NBI following TAPI data models. Several API entries are supported, including creating Connectivity Services with some requirements (source and destination, bandwidth provision, latency constraints, QoT conditions, etc.) or topology/context retrieval. The TAPI models have been extended to support SDM core switching. This affects: - Service Interface Points (SIPs) - Node Edge Points (NEPs) - Connection End-Points (CEPs) These entities are extended to include information about the status of the core and virtual DWDM links.
2	SBI Netconf	The innovative MBoSDM nodes developed in SEASON are being modelled and a set of application programming interfaces (APIs) defined for their configuration and control. A NETCONF agent will be provided by the project at the end of the activity. The main KPIs will relate to the validation on a real testbed, including configuration latency.

3.2.1.3 Status and features under development

At the time of writing, the status is as follows:

- That the control plane has been implemented and demonstrated with ideal nodes which are emulated.
- Service provisioning can be performed in both media channels as well as SDM layer.

The key innovations are in the scope of network control and management:

i) The conception of a path computation and resource allocation algorithm that works in a two-step process, that considers multiband spectrum assignment with contiguity and continuity as well as core continuity constraint and showing key concepts such as aggregation and grooming as well as virtual link management.

ii) The design and implementation of a MBoSDM control plane via a centralized SDN controller, including the provisioning of services in both the DWDM and the SDM layer. This includes the extension of Transport API (TAPI) north bound interfaces models to support SDM switching and core allocation.

iii) The definition of an abstracted MBoSDM node device model, with configuration API and yang data model along with the NETCONF agent that enables remote configuration and control. Demo with 60+ node Telefonica Spanish topology, showing the applicability to core networks with a significant number of nodes.

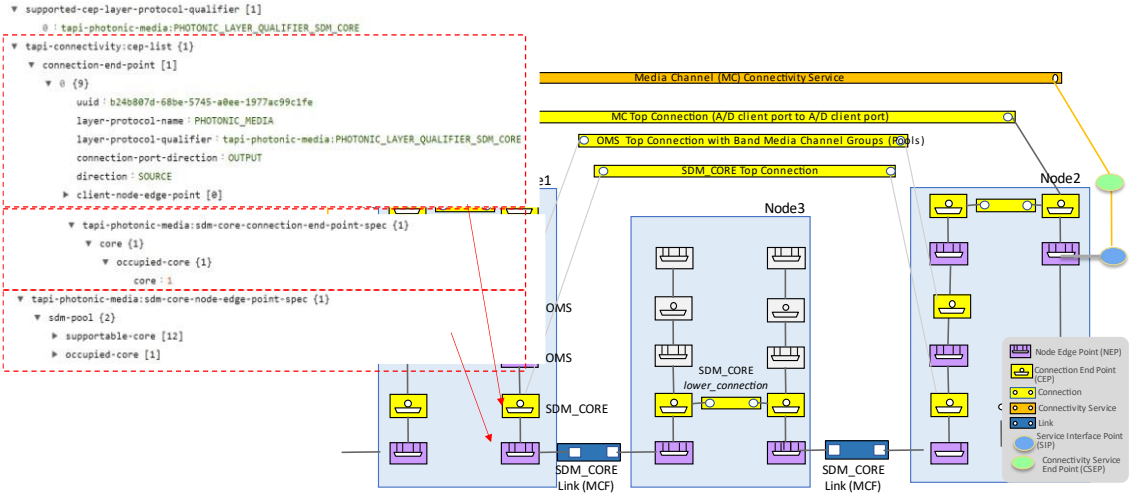


Figure 3.2-3: TAPI context augmented with SDM core information.

Features under development:

- NETCONF SDN agent with the node abstraction agreed in cooperation with WP3. It is expected that this feature will be developed around Q1-Q2 2025.

3.2.1.4 Validations and Intermediate results

Initial validations have been carried out with simple networks and with Telefonica 60+ node network. A video has been recorded and shown during reporting period 1. A demonstration has been submitted to OFC'25 demo zone. All these demonstrations take place with emulated hardware.

The topology used can be seen in the figure below:

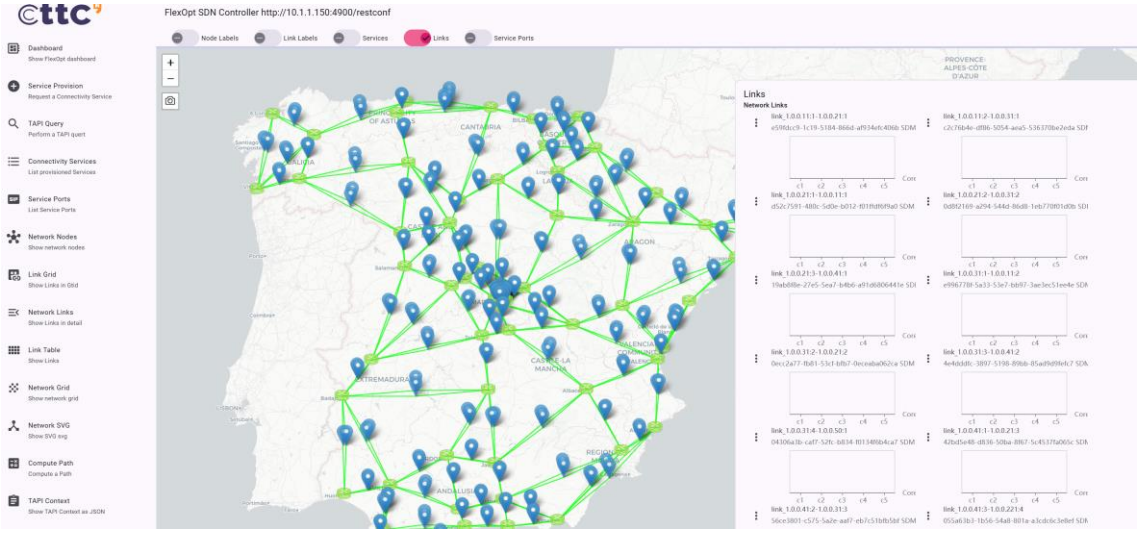


Figure 3.2-4: Telefonica Spanish topology to illustrate and evaluate the MBoSDM algorithm.

The demonstration shows the capability of setting up WDM connections that rely on the dynamic provisioning of SDM connections that support virtual WDM links

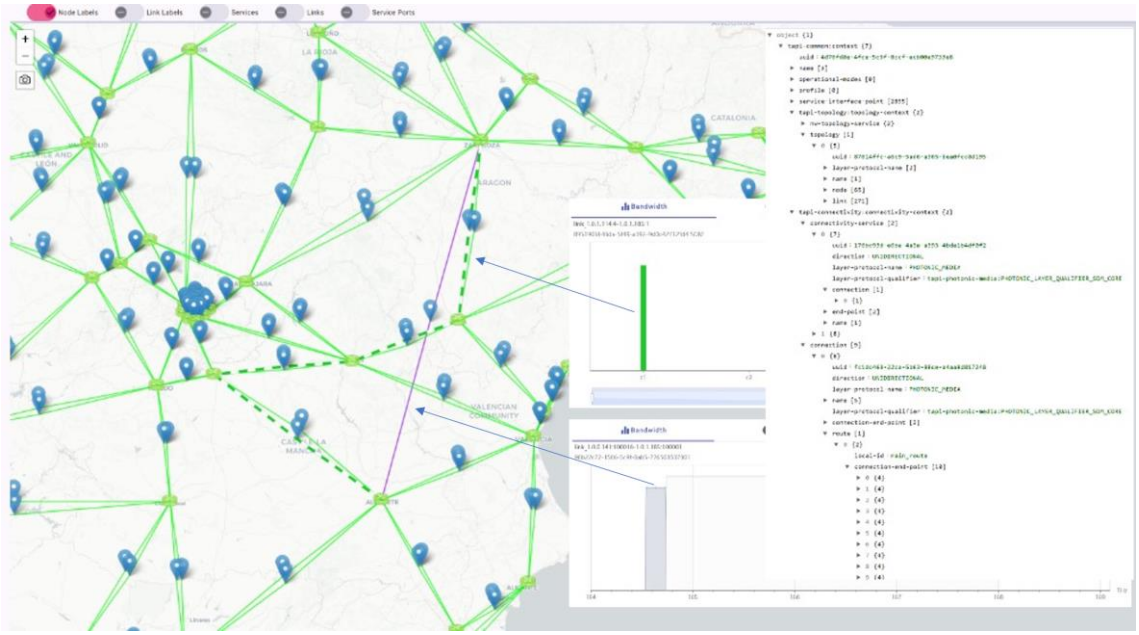


Figure 3.2-5: Detail of a service provisioning involving a core allocation as well as a optical channel (flexi-grid) showing the role of the two optical levels.

3.2.2 Ensemble OLS controller

Description:

The optical controller proposed in SEASON for OLS control, based on the Adtran Ensemble Network Controller software solution and is offering a northbound ONF Transport-API v2.1

Dissemination Level	PUB (Public)

(TAPI). The OLS components can be controlled using the southbound interfaces with native YANG models and uses NETCONF/SNMP to perform network operations. It takes care of the provisioning of optical services which includes configuration of ports, channel frequencies and much more on the OLS. On the other hand, the ENC supports TAPI v2.1 as interface in the northbound. The TAPI topology model provides the explicit multilayer topology that the Layer 2 to Layer 0 represents. This topology includes the OTS, OMS, OCH, ODU, and DSR layers.

Based on ONF TAPI 2.1 models, the Adtran TAPI Implementation supports a TAPI topology flat abstraction model that collapses all layers into a single multilayer topology. A single topology represents all network layers such as DSR, ODU, OCH, and Photonic Media, which include media channels, OMS, OTS and so on. Service Interface Points SIPs associate to all DSR, ODU and PHOTONIC_MEDIA NEPs in the network support service configuration. The Adtran ENC TAPI implementation supports both fully aggregated topological scenario, where Adtran provides both optical line systems (OSLs) and optical transponders, as well as a partially disaggregated scenario, where Adtran provides only the OLS.

The following two novel works are conducted as part of the WP4,

1. The ENC TAPI v2.1 is adopted such that flexible channel bandwidth can be configured in terms of (50GHz, 75GHz, 100GHz) instead of 50GHz standard bandwidth.
2. The TAPI driver is mounted to ENC is complied to v2.1 as mentioned previously. Also, gNMI/gRPC based telemetry streaming is adopted to streaming the telemetry data from OLS components . More details are described in Section 4.2.

APIs and Interfaces for Integration:

The ENC TAPI works over the REST framework offering the CRUD operations to perform over the network. This helps to create, read, update and delete services on the network. Further details on the operations are provided in Table 3.2-1.

Table 3.2-1: REST API of TAPI-Driver integrated to ADTRAN ENC Controller.

S.No	Interface	Description
1	REST-TAPI TAPI-Context	METHOD: GET URL: https://IP:PORT/restconf/data/tapi-common:context HEADERS: 'Authorization: Basic encoded_64_pwd'
2	REST-TAPI Service Creation	METHOD: POST URL: https://IP:PORT/restconf/data/tapi-common:context/tapi-connectivity:connectivity-context HEADERS: 'Authorization: Basic encoded_64_pwd ' 'Content-Type: application/yang-data+json' BODY: '{"tapi-connectivity:connectivity-service": [{"tapi-adva:adva-connectivity-service-spec": {"customer": "CUSTOMER_NAME"}, "end-point": [{...}, {...}], "service-layer": "PHOTONIC_MEDIA", "service-type": "POINT_TO_POINT_CONNECTIVITY", "name": [{"value": "SERVICE_NAME", "value-name": "USER"}]}]}'
3	REST-TAPI	METHOD: DELETE

	Service Deletion	URL: https://IP:PORT/restconf/data/tapi-common:context/tapi-connectivity:connectivity-context/tapi-connectivity:connectivity-service=UUID/ HEADERS: 'Authorization: Basic xxx'

Status:

- The integration of ADTRAN OLS controller with Optical Controller (CTTC) in section 3.2.1 is ongoing. The TAPI client from Optical Controller communicates with ADTRAN TAPI driver to retrieve the SIP and context details of the OLS network. This helps to perform the CRUD operations as mentioned above to provision/configuration/re-configuration of the service in the OLS from optical controller.
- The TAPI based gNMI/gRPC streaming is implemented as a proxy-agent. The integration of this agent to the TAPI driver v2.1 is planned for integration.

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

Point-to-multipoint (P2MP) coherent optics based on digital subcarrier multiplexing (DSCM) 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 subcarriers (SCs) at a lower symbol rate. Each individual SC can be treated independently of all others, including modulation, management, 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. The goal is to experimentally demonstrate the feasibility of the dynamic assignment using P2MP transceivers in a metro scenario with a full SDN control plane.

3.2.3.1 Description

The FlexOpt optical SDN Controller can be deployed as an optical controller under the control of a transport Network Orchestrator. Its role is dual: on a first step, abstract the details of the Intelligent Pluggable Manager (IPM) controller and offer a standard TAPI interface towards the Orchestrator and, on a second step, coordinate the configuration of the P2MP pluggables and the provisioning of spectrum services (media channel services) using a delegate OLS controller.

The P2MP programmable pluggables implement a dual management interface. The FlexOpt Optical Network Orchestrator is able to process user requests for P2P and P2MP services. In a first step we focus on P2P digital signal rate (DSR) services between router ports. 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 the FlexOpt controller 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.

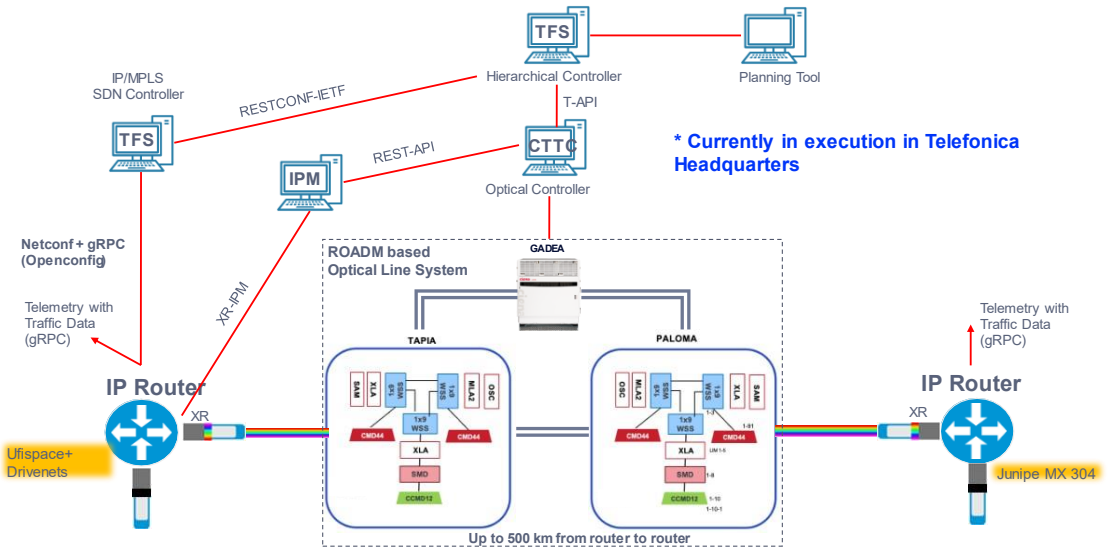


Figure 3.2-6: Network demonstrator and scenario for the P2P OpenXR control integration

3.2.3.2 APIs and Interfaces for Integration

S.No	Interface	Description
1	RESTCONF TAPI Models	The FlexOpt SDN controller exports the NBI following TAPI data models. Several API entries are supported, including • Creating Connectivity Services with some requirements (source and destination, bandwidth provision, latency constraints, QoT conditions, etc.). • Performing and/or replying to path computation requests (based on a specific instance of path computation interface defined in TAPI) • Additional interfaces have been defined to support the augmentation of topological elements with physical layer information data. • The interface towards the Telemetry System can use TTAPI streaming records TAPI Reference Implementation Agreement (RIA) for streaming TR-548 and encoded using MQTT or Kafka.
2	REST IPM	In order to interact with the IPM controller, FlexOpt implements the REST interface as provided by the IPM documentation This covers: • Network topology discovery • Create XR networks (constellations) • Add /Attach leaves • Map ports to XR networks

3.2.3.3 Status and features under development

Phase I has been implemented as reported in the next section (scenario defined in Figure 3.2-6: Network demonstrator and scenario for the P2P OpenXR control integration).

Phase II is still under development. A critical aspect is to extend the TAPI data models to support point to multipoint services and how this is mapped into the optical connections. This will be developed during Q1-Q2 2025.

3.2.3.4 Validations and Intermediate results

Initial validation

The FlexOpt receives the request from it NBI and, as part of the process, it computes the nominal central frequency for the DSCM and requests the creation of an XR network or constellation to the IPM (1), configuring the optical line system as needed (2) as well as the creation of the service (3) 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 attach 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).

The experimental setup has been implemented in Telefonica's Future Network Laboratory (see Figure below), utilizing two IPoWDM routers: a Cisco NCS 57B1 running IOS 24.3.1 and a Ufispace router with Drivenets software both 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. For system management, 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 Adtran 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 connects to Telefonica's laboratory via a VPN, ensuring seamless integration and control across the distributed experimental setup.

Currently, P2P setup is up & running XR pluggables properly working in the commercial routers, the connection to IPM established and the first work with SDN integration has been completed. Energy measurements are in progress

Topology and Resource discovery

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 nodes (see Fig. 1c) and module Ethernet client ports are mapped to TAPI DSR Service-Interface-Points (SIPs). 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.

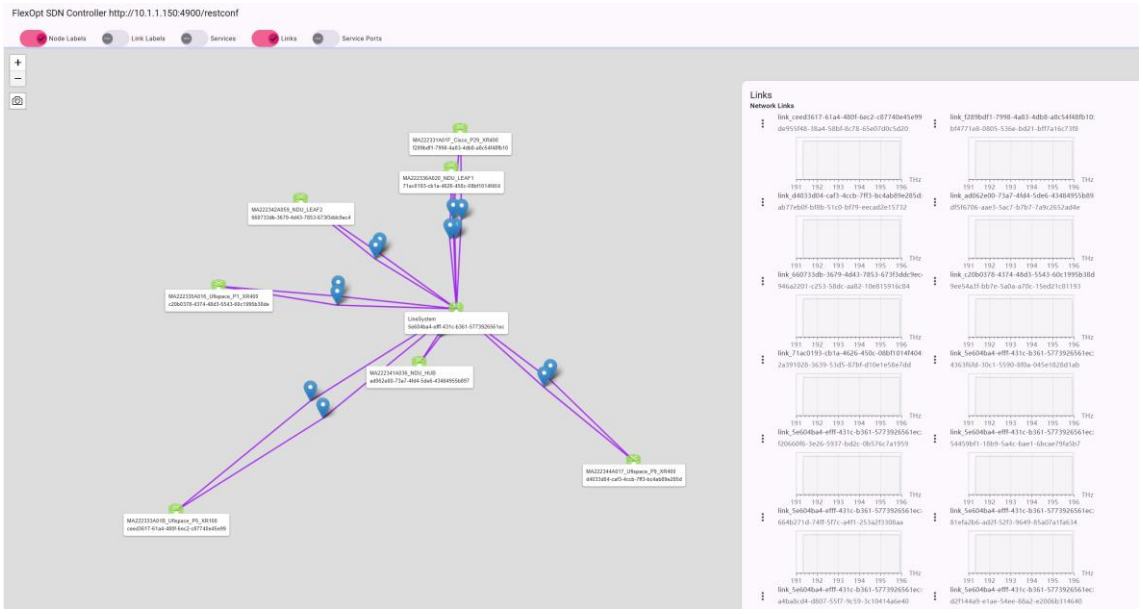


Figure 3.2-7: IPM topology with discovery of pluggables.

Service provisioning

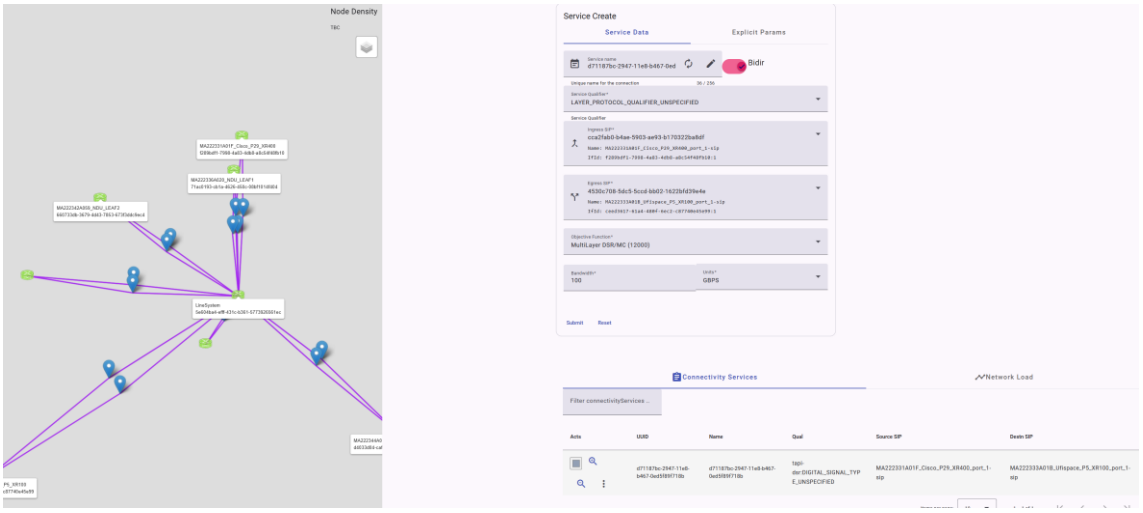


Figure 3.2-8: Establishment of a 100 Gb/s service is triggered using FlexOpt SDN controller

The establishment of a 100 Gb/s service is triggered using FlexOpt SDN controller GUI Interface (Figure 3.2-8) 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 involved 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. The demonstration shown in Figure 3.2-9 has been submitted to OFC'25 demo zone.

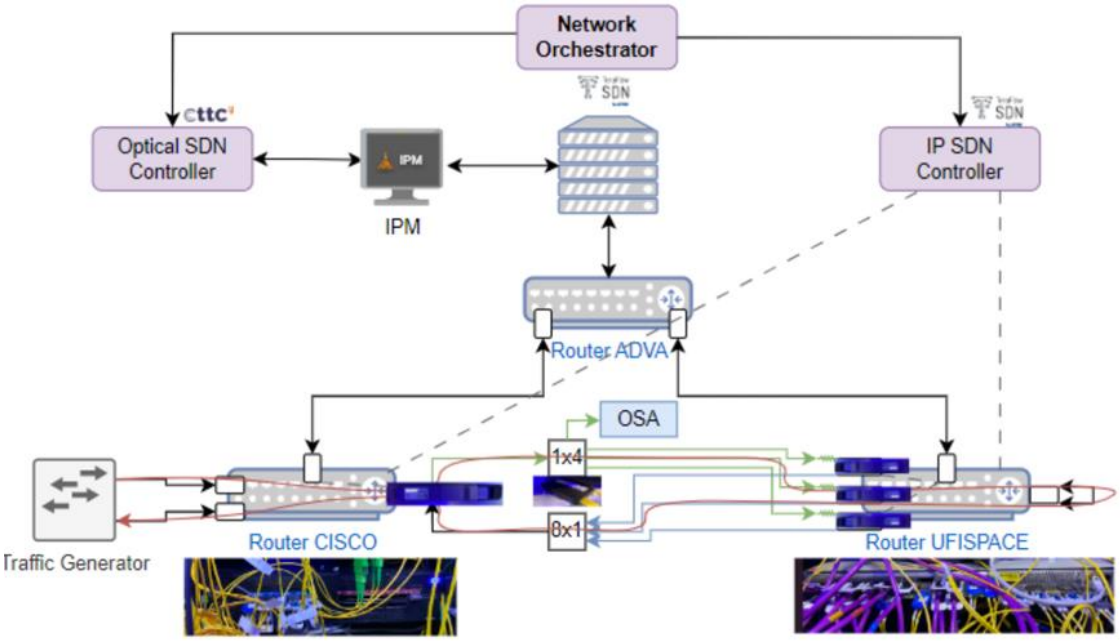


Figure 3.2-9: Experimental Setup for the validation of the Provisioning of OpenXR services

Dissemination Level	PUB (Public)

```
[13:33:51.807715] 203486 [D] [IPM]: Creating XR Network constellation for LSP 1b8c188e-d4f5-4df4-a6fd-fb83c361e3d8
[13:33:51.807800] 203486 [T] xr_network: [{"config":{"name":"1b8c188e-d4f5-4df4-a6fd-fb83c361e3d8","constellationFr
| /home/rcasellas/src/spce/src/server/plugins/ipm/client.cpp:285
[13:33:51.807843] 203486 [D] [IPM]: posting ...
[
    {
        "config" : {
            "name" : "1b8c188e-d4f5-4df4-a6fd-fb83c361e3d8",
            "constellationFrequency" : 191375000,
            "modulation" : "16QAM",
            "tcMode" : true,
            "cTEOptimization" : "dualMarker",
            "topology" : "auto"
        },
        "hubModule" : {
            "selector" : {
                "moduleSelectorByModuleName" : {
                    "moduleName" : "MA222331A01F_Cisco_P29_XR400"
                }
            },
            "module" : {
                "trafficMode" : "L1Mode",
                "txCLPtarget" : -300,
                "fecIterations" : "standard"
            }
        },
        "leafModules" : [

        ]
    }
```

Figure 3.2-10: Example of JSON message sent to configure an OpenXR hub between the SDN optical orchestrator and the Intelligent Pluggable Manager

3.2.4 SDM Passive Optical Network (PON) Controller

The Passive Optical Network (PON) Controller is a component delegated to manage and configure PON infrastructures within the SEASON framework. This SDN-enabled controller operates as an intermediary, providing a programmable interface to abstract the complexity of underlying hardware while enabling flexible service configurations. The PON controller bridges the gap between high-level orchestration systems and the underlying PON hardware, providing a seamless interface for configuration, monitoring, and service provisioning.

The PON Controller exposes a RESTful API as its northbound interface (NBI), enabling seamless integration with orchestration platforms and network management tools. This API allows for retrieving device configurations, accessing operational data, and provisioning advanced point-to-point (P2P) links. Developed with OpenAPI (Swagger 2.0) specifications, the API supports both JSON and XML data handling, ensuring compatibility with a wide range of management solutions. The JSON support facilitates integration with modern data processing tools, while XML ensures consistency with NETCONF, the protocol typically used for communication with PON devices.

The controller's southbound interface (SBI) relies on NETCONF over SSH to communicate securely and reliably with the underlying PON hardware. NETCONF provides a structured framework for configuration management, telemetry streaming, and event notifications, leveraging YANG models to ensure precise and interoperable configuration mappings. The adoption of SSH ensures that all interactions between the controller and PON devices are encrypted and authenticated, further enhancing the security and robustness of the system.

Currently, the RESTful API supports key functionalities such as retrieving full device configurations, accessing profile-specific settings, and conducting element-specific searches. Operational data retrieval is similarly supported, enabling administrators to monitor real-time device states and operational metrics. These features facilitate efficient management of Calix E7-2 devices and pave the way for advanced PON use cases.

In the Table 3.2-2 are listed the currently functioning and tested APIs. The IP and port provided refer to our test environment, specifically the PON Controller at IP address 10.10.164.88 and port 5001.

Table 3.2-2: PON related APIs to be integrated in over-arching control and network orchestration

S.No	Interface	Description
1	PON Controller API	Retrieve Ethernet Class-Map Configuration (JSON) METHOD: GET URL: http://10.10.164.88:5001/api/v1/json/device/config/class-map/ethernet HEADERS: 'accept: application/json'
2	PON Controller API	Retrieve Ethernet Class-Map Configuration (XML) METHOD: GET URL: http://10.10.164.88:5001/api/v1/device/config/profile/class-map/ethernet HEADERS: 'accept: application/json'
3	PON Controller API	Retrieve Full Device Configuration METHOD: GET URL: http://10.10.164.88:5001/api/v1/device/config HEADERS: 'accept: application/json'
4	PON Controller API	Search Configuration Elements METHOD: GET URL: http://10.10.164.88:5001/api/v1/device/config/search/{element} HEADERS: 'accept: application/json'
5	PON Controller API	Retrieve Operational Data (JSON) METHOD: GET URL: http://10.10.164.88:5001/api/v1/json/operational/operational HEADERS: 'accept: application/json'

3.2.5 IPoWDM Controller

The TeraFlow SDN Controller (TFS) is an open-source, microservice-based SDN controller that in SEASON is also adopted as IP Controller for IPoWDM devices (e.g., IPoWDM switch/routers and SmartNICs, equipped with coherent pluggable modules).

In SEASON D4.1, the TFS SDN Controller was extended with an OpenConfig SBI driver module. The new driver successfully enabled the topology discovery and the onboard of the devices. The TFS driver discovered automatically the device endpoints and its components.

In this deliverable, we report on three newly added functionalities to support OpenConfig white boxes as well as the novel concept of DPU/SmartNICs equipped with coherent pluggable modules.

To achieve this objective, the following enhancements and extensions have been performed:

A. REST API for configuring the OpenConfig transceiver(s)

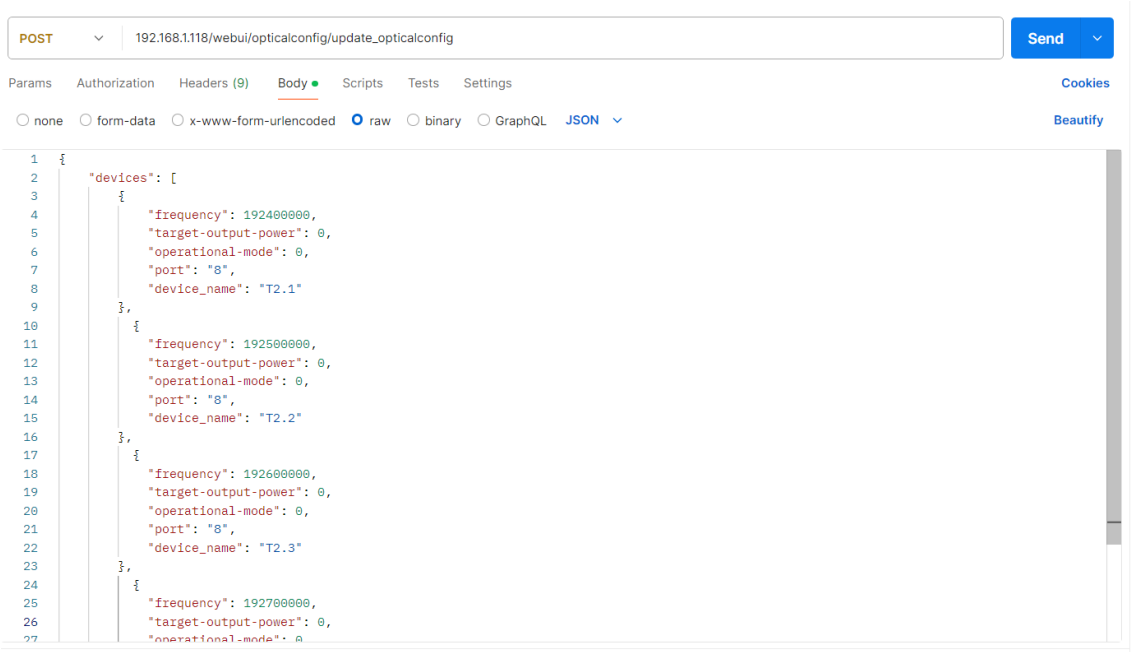


Figure 3.2-11: Post request.

This API enables the possibility to directly configure, through TFS, a number (one of more) of transceivers at once. This API permits to bypass all the TFS logic (e.g., internally performed path computation, in case it is delegated to an external path computation element), directly invoking the OpenConfig driver to configure the optical device(s).

As we can see from the request in Figure 3.2-11, we can send a post request with a JSON array of devices as payload. For each device, it is specified the device_name, the optical-channel to be configured with the specific values for the following parameters (Frequency, target_output_power, operational_mode, status).

In case of error in choosing the correct port, or if the device is not available, the REST API will handle the error, sending back an informative message about the error, beside the success message in case everything went well.

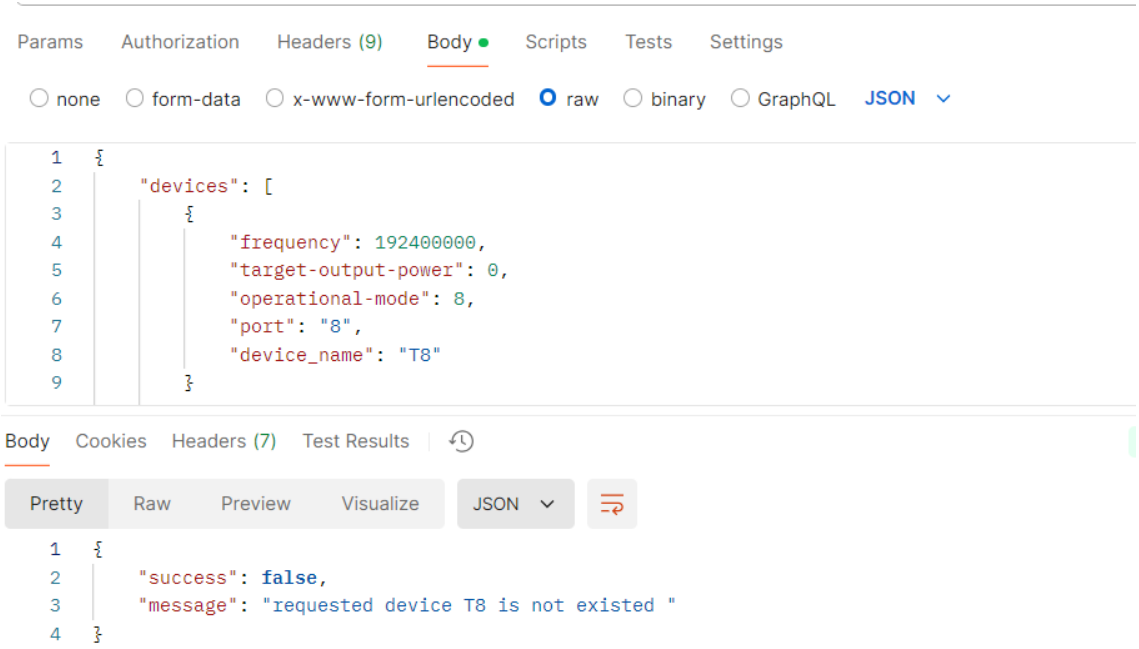


Figure 3.2-12: Example of error message.

In Figure 3.2-12 an example of error message is shown, while requesting unavailable device. If configuring the device was successful, the updates will be reflected into the context DB, showing the changes also in the GUI in real time.

This contribution on the REST API to configure the OpenConfig transceiver(s) has been successfully merged within the develop branch:

https://labs.etsi.org/rep/tfs/controller/-/issues/204

B. Automatic discovery of OpenConfig transceivers

This functionality has been significantly extended compared to the version described in D4.1. TFS is now able to discover automatically all the endpoints (e.g., optical-channels) related to transceivers Figure 3.2-13 shows the result of the discovery process, after successful onboarding of a device. More specifically, a simple JSON file is used to onboard a device. The file includes control plane entry-point (IP address and port), type of device and driver to use. No interfaces or ports are provided. Indeed, this information is automatically discovered by TFS, by performing a *get* on the device and parsing the result. All the transceivers will be listed with the number of endpoints. In the example, considering the transceiver T2.2 as in Figure 3.2-14, it is possible to see the endpoints' details. Please note, in TFS, the term optical-transponder is currently used to refer to any type of optical endpoint, including both standalone transponders and pluggable transceivers.

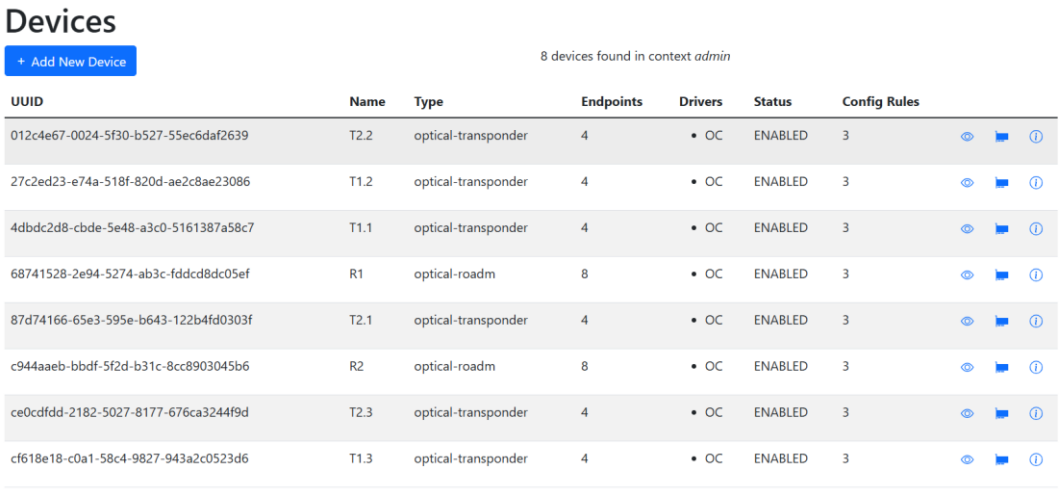


Figure 3.2-13: View of devices.

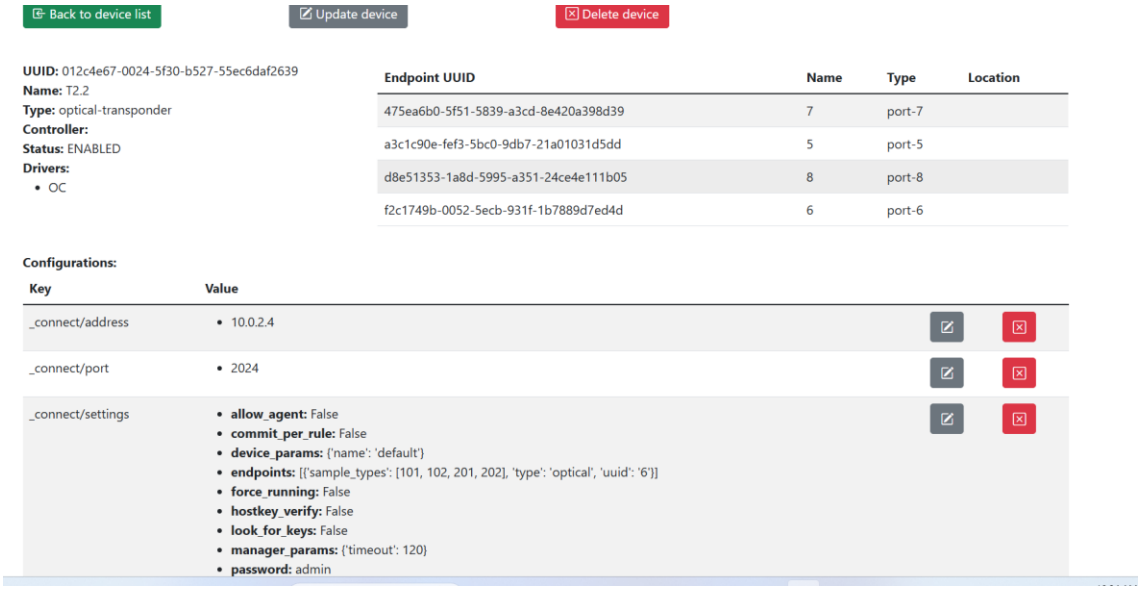


Figure 3.2-14: Configured end point details.

This contribution on automatic discovery of OpenConfig transceivers has been successfully merged within the develop branch:

https://labs.etsi.org/rep/tfs/controller/-/issues/203

C. GUI view of the OpenConfig transceiver configuration

The webUI of TFS has been extended to support the configuration via GUI of the optical device configurations.

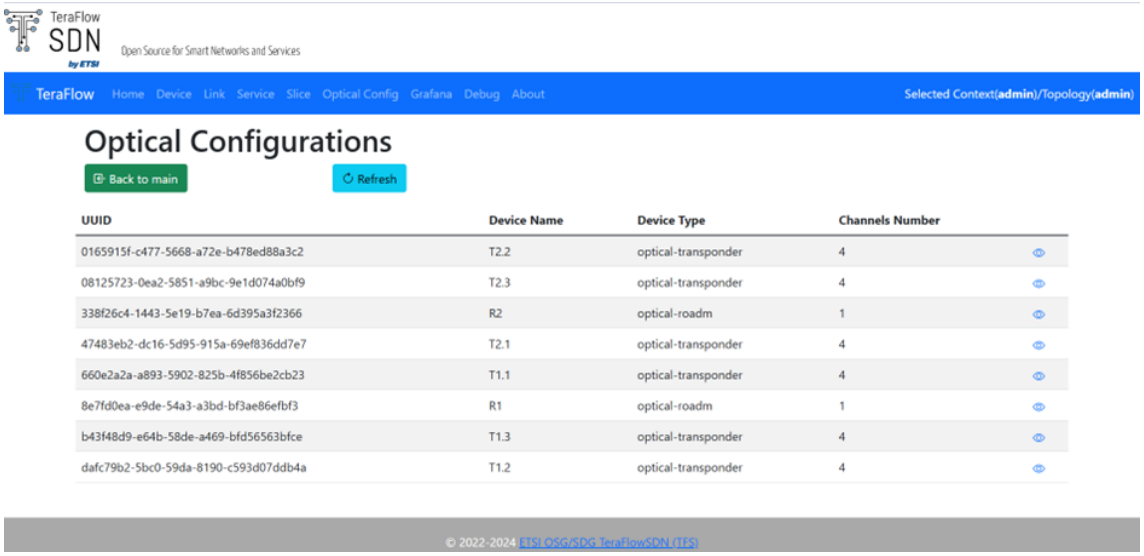


Figure 3.2-15: View of implemented section for optical device configuration.

Figure 3.2- shows the implemented Optical Device Configurations tab, where all the discovered optical devices are listed, highlighting their unique identifier, name, type, and number of relevant interfaces. From this list, it is possible to further explode each device, accessing the device details.

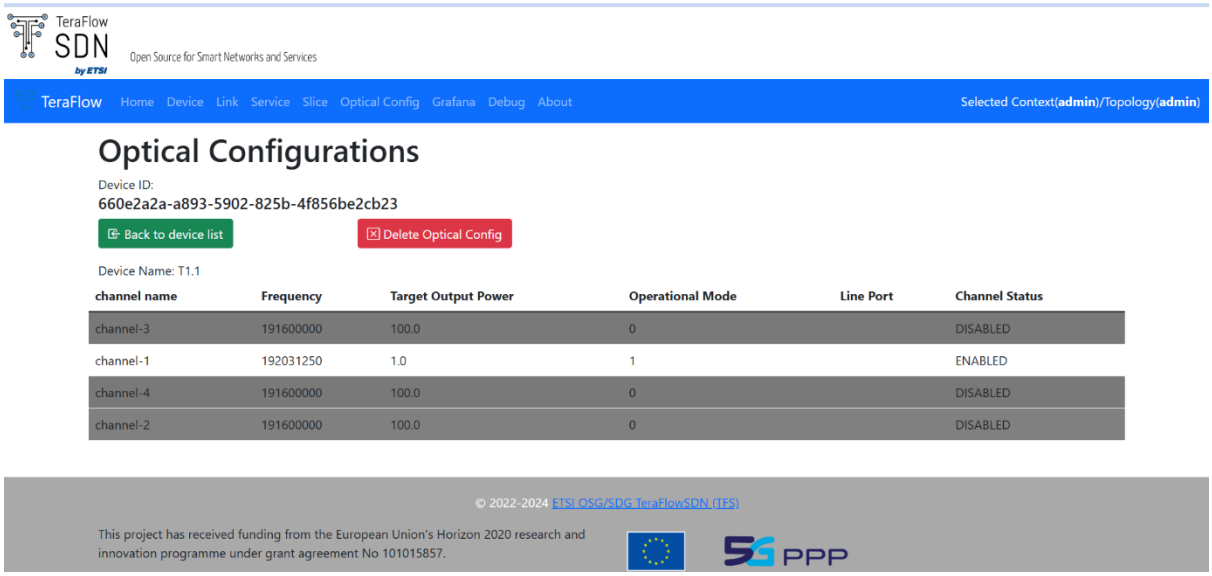


Figure 3.2-16: Detail of transceiver configuration.

Figure 3.2- shows the details of a transceiver device configuration. In this table, all the interfaces (e.g., optical channels) are listed, the key of the table is the channel_id and the key configuration parameters of the channel are included:

- Frequency
- Target Output Power
- Operational Mode

- Status which will present its status if enabled (white background) or disabled (grey background)

This information is retrieved via *get rpc* from the optical device and is stored in the context DB. The page provided also a button to perform the delete of the configuration from the context DB.

This contribution on the GUI for OpenConfig transceiver configuration has been successfully merged within the develop branch:

https://labs.etsi.org/rep/tfs/controller/-/issues/108

3.2.6 RAN Intelligent controller

The RAN Intelligent Controller (RIC) is a central component in the Open Radio Access Network (O-RAN) architecture. Details of the O-RAN architecture, as shown in Figure 3.2-, can be found [ORAN24].

The purpose of the RIC is to enhance the performance, flexibility, and management of the Radio Access Network (RAN) by enabling more dynamic and automated control over RAN network functions. The RIC plays a central role in optimizing network operations, reducing latency, improving service quality, and fostering innovation by providing a platform for third-party applications (called rApps and xApps) to control and manage RAN elements in real time.

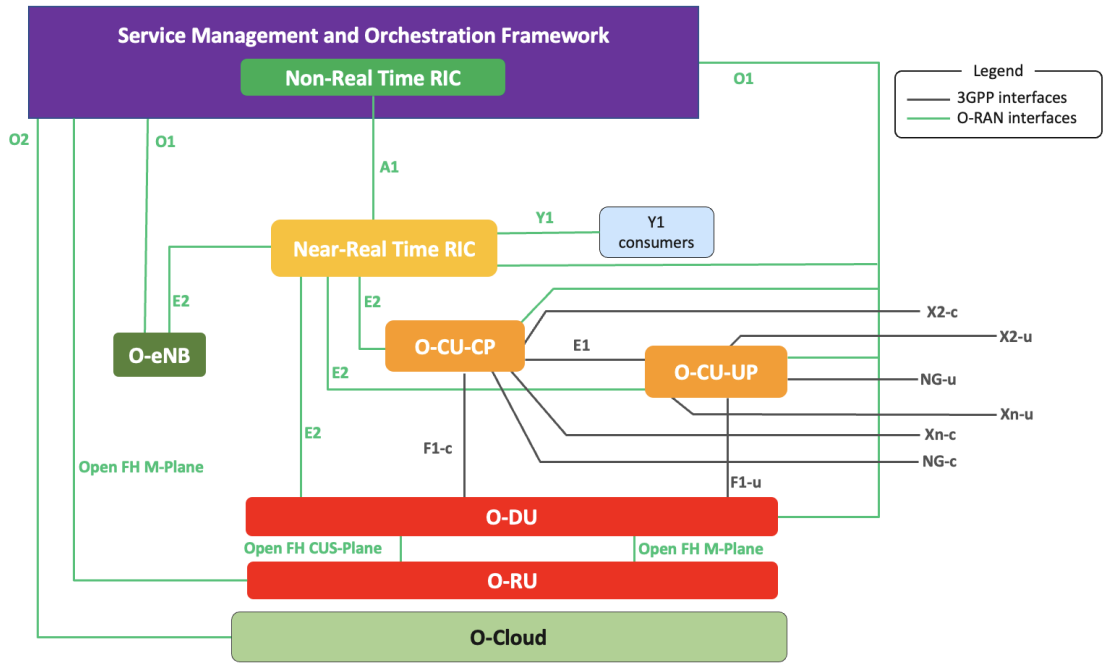


Figure 3.2-17: O-RAN architecture overview from O-RAN Alliance

The RIC exists in two forms:

1. **Near-Real-Time RIC** (Near-RT RIC):
    - Operates with a time frame of less than one second (typically in the range of 10 milliseconds to 1 second).

- It is responsible for tasks that require real-time or near-real-time control of RAN functions, such as handover management, interference reduction, and scheduling of radio resources.
- It interacts with the base stations and RAN elements closely, making rapid decisions to optimize network performance.

2. **Non-Real-Time RIC** (Non-RT RIC):
   - Operates on a longer time scale (typically in minutes, hours, or even days).
   - This controller is focused on higher-level tasks such as policy management, network planning, and long-term optimizations. It also provides machine learning and data analytics to continuously improve the efficiency of the RAN over time.
   - Non-RT RIC interacts with the Near-RT RIC, influencing its operations with long-term strategies and optimizations.

Some of the key benefits and functions of the RIC include:

- **AI and Machine Learning Integration**: The RIC can leverage AI/ML models for predictive and adaptive control, helping to optimize traffic, manage spectrum, and improve energy efficiency.
- **Flexibility**: It allows operators to customize and modify network behavior by introducing new applications or services without needing to change the underlying hardware.
- **Vendor Interoperability**: Open RAN and the RIC facilitate multi-vendor ecosystems, reducing reliance on a single vendor's hardware and software, promoting innovation, and reducing costs.
- **Improved Performance**: By enabling near-real-time adjustments and optimizations, the RIC helps to improve network performance in areas such as throughput, latency, and resource allocation.

In summary, the RIC is a critical enabler of the Open RAN vision, allowing operators to improve efficiency and innovation in their network management by utilizing open interfaces, AI-driven applications, and dynamic control mechanisms.

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. The TNO maintains a layered view of the transport network, from PON to metro-core, based on the information received by the domain controllers. It is able to 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 scope. In the context of SEASON, the main connectivity service that is going to be provided is the "slice" service.

Network Slice service: The Network Slice Service offered by the SEASON Transport Network Orchestrator follows the RFC 9543 definition and enables connectivity between a set of end points with specific Service Level Objectives (SLOs) and Service Level Expectations (SLEs) over a common underlay network. The SLOs are measurable network attributes and characteristics. In the context of season, the main SLOs are Guaranteed Minimum Bandwidth between two end points of the service and a guaranteed maximum latency, which is the upper bound of the network latency when transmitting between two end points. The end to end service

orchestrator can request the transport network orchestrator using the Yang model defined in draft-ietf-teas-ietf-network-slice-nbi-yang

Exposure of Service Attachment points (Customer view towards service orchestration): The TNO exposes a SAP (Service Attachment points) topology, which is an abstract view of the Service Provider network topology that contains the points from which connectivity services can be attached, for example basic connectivity, VPN or slices in the case of SEASON. Also, the SAP topology can be used to retrieve the points where the services are actually being delivered to customers. In the context of SEASON, the End-to-end service orchestrator will make use of the SAP topology to learn the end-points of the connectivity services. The Yang model to describe the SAP topology is defined in RFC 9408. A sample json is shown in figure


```
{
    "ietf-network:networks": {
      "network": [
        {
          "network-types": {
            "ietf-sap-ntw:sap-network": {
              "service-type": [
                "ietf-vpn-common:ietf-network-slice"
              ]
            }
          },
          "network-id": "season-topology",
          "node": [
            {
              "node-id": "ONU-1",
              "ietf-sap-ntw:service": [
                {
                  "service-type": "ietf-vpn-common:ietf-network-slice",
                  "sap": [
                    {
                      "sap-id": "sap#11",
                      "description": "ONU port",
                      "role": "ietf-sap-ntw:uni"
                    }
                    ],
                    "sap-status": {
                      "status": "ietf-vpn-common:op-up"
                    }
                  }
                ]
              }
            ]
          }
        ]
      }
    }
```


Figure 3.3-1: JSON sample of SAP topology.

3.3.2 Service Orchestration with Kubernetes

In the SEASON project, Kubernetes serves as the primary service orchestrator. Kubernetes is an open-source platform designed to automate the deployment, scaling, and management of containerized applications. Its robust features and flexibility make it an ideal choice for orchestrating services across diverse network domains and environments, including cloud and edge computing resources. Furthermore, it provides a unified platform for managing the

lifecycle of microservices and applications. In the context of SEASON, it orchestrates network services by automating deployments across multiple nodes, ensuring consistency and reliability. Kubernetes' ability to automatically scale applications based on demand optimizes resource utilization, while its self-healing capabilities monitor the health of services and automatically restart or replace failed components. Additionally, Kubernetes facilitates service discovery and load balancing, automatically exposing services using DNS names or IP addresses and balancing network traffic to maintain optimal performance.

To enhance decision-making within the service orchestrator, Machine Learning Operations (MLOps) practices can be integrated into the Kubernetes environment. MLOps combines machine learning, DevOps, and data engineering to streamline the deployment and management of AI/ML models in production. This integration ensures that AI/ML models can be continuously developed, trained, deployed, and monitored in a systematic and scalable manner. MLOps within Kubernetes encompasses several key aspects. Continuous Integration and Continuous Deployment (CI/CD) pipelines automate the process of building, testing, and deploying AI/ML models, enabling rapid updates and reducing time to production. Kubernetes' native scalability and resource management capabilities ensure that AI/ML workloads are efficiently handled, adjusting computational resources based on demand. Robust monitoring and logging, facilitated by tools like Prometheus and Grafana, provide real-time insights into model performance and system metrics, allowing for timely detection of drifts or anomalies and triggering retraining processes when necessary. Moreover, version control and reproducibility are maintained by managing versions of datasets, models, and configurations, ensuring that results are consistent and facilitating rollback if needed.

The integration of AI/ML models within the Kubernetes orchestrator significantly enhances the SEASON project's ability to make intelligent, data-driven decisions. For instance, AI/ML applications can predict network traffic patterns to proactively allocate resources and prevent congestion, optimizing bandwidth utilization. Adaptive scaling mechanisms can automatically adjust the number of instances of network functions based on current and projected demand, ensuring optimal performance without over-provisioning. Fault prediction and recovery systems can identify potential failures before they occur, initiating preventive measures or automated recovery processes to maintain network reliability. Additionally, energy efficiency can be improved by optimizing resource utilization to reduce energy consumption, aligning with the project's sustainability goals without compromising service quality. These AI/ML capabilities are deployed and managed using MLOps practices within Kubernetes, ensuring that models remain accurate, reliable, and up-to-date. By leveraging Kubernetes' scalability, reliability, and support for containerized workloads, the service orchestrator can respond dynamically to changing network conditions and service requirements, enhancing overall efficiency and user experience.

3.3.3 Integration with Transport Network Orchestration

The TeraFlowSDN (TFS) controller is integral to transport network orchestration within the SEASON project. TFS is designed to handle the lifecycle management of network connection services across multi-domain, multi-technology contexts, including optical and packet networks. Its capabilities are essential for provisioning end-to-end connectivity services that meet the stringent performance requirements of applications like enhanced Mobile Broadband (eMBB). For comprehensive service orchestration, it is imperative to establish a seamless interface between Kubernetes and TFS. This integration ensures that service requests and operational directives originating from Kubernetes can be effectively translated into network configurations and actions within the transport domain managed by TFS.

To facilitate effective communication between Kubernetes and TFS, the IETF Network Slice model is adopted. This standardized framework allows for the representation of end-to-end network services as "slices," abstracting the complexities of underlying network technologies. Using the IETF Network Slice model enables intent-based service requests, where Kubernetes can specify desired service characteristics such as bandwidth and latency without needing detailed knowledge of the transport network. This abstraction provides flexibility and agility, allowing TFS to map high-level intents to specific configurations efficiently. Adhering to IETF standards ensures interoperability and simplifies integration with other systems.

For the communication mechanism, gRPC (Google Remote Procedure Call) is employed. gRPC is a high-performance, open-source RPC framework that supports bi-directional streaming and is language-agnostic, making it suitable for microservices architectures. The use of gRPC ensures low-latency, high-throughput communication essential for real-time orchestration tasks. Additionally, gRPC's support for Protocol Buffers allows for efficient serialization of messages, while its built-in features for load balancing and authentication enhance the scalability and security of the communication channels.

Using gRPC provides several advantages:

- **Performance**: gRPC uses HTTP/2, enabling multiplexed streams and efficient binary serialization, which is suitable for high-throughput, low-latency communication.
- **Language Agnostic**: gRPC supports multiple programming languages, facilitating integration between modules written in different languages.
- **Bi-directional Streaming**: gRPC supports streaming in both directions, which is useful for real-time data exchange and notifications.

Defining the specific data exchanged between Kubernetes and TFS is critical for successful integration. The data exchange includes:

- **Service Requests**: Kubernetes sends service requests using the IETF Network Slice model, specifying desired performance metrics and service characteristics.
- **Operational Metrics**: TFS exposes metrics such as latency, bandwidth utilization, and energy consumption associated with network slices.
- **Control Commands**: Instructions for creating, modifying, or terminating network slices, enabling dynamic orchestration based on AI/ML insights.

This clear definition of data elements ensures that Kubernetes and TFS collaborate effectively. Kubernetes leverages AI/ML models to optimize service orchestration based on real-time network conditions, while TFS provides the necessary network configurations and feedback.

3.4 OPERATIONAL, PLANNING AND RESOURCE ALLOCATION TOOLS

3.4.1 OCATA Digital Twin for Multiband

Description

The classical OCATA architecture for C-band (see Figure 3.4-a) considers Deep Neural Networks (DNN) models that are independent of the specific channel; a reference channel (RCh) in the middle of the C-band is considered. Incoming optical signals generated using modulation format m-QAM are processed in the time domain. Therefore, incoming in-phase (I) and quadrature (Q) optical constellation with m distinct constellation points (CP) are processed. Specifically, an input IQ optical constellation (X) consist of a set of symbols $x \in X$, where each $x=[x^I, x^Q]$ belongs to a CP.

A features extraction (FeX) module finds features for each CP of the received signal by applying Gaussian mixture model (GMM) fitting, which characterizes the CP as a set of bivariate Gaussian distributions. The output of the FeX module is a set of features Y, which contains 5 features for each CP, i.e., $Y_i=(\mu^I, \mu^Q, \sigma^I, \sigma^Q, \sigma^{IQ})_i$ is the vector of features for CP i, where μ^I and μ^Q represent the mean position for i over I and Q axes, respectively, σ^I and σ^Q are the variance of i over the axes and σ^{IQ} is the covariance between the axes. It is worth noting that the dispersion of symbols that belong to a certain CP provides valuable insight of the level of noise affecting the signal, both linear (LI) and NLI noise. As the total noise increases, the dispersion of the symbols also increases. The FeX module is followed by a set of DNNs modeling the propagation of the signal on the optical components in the route of the lightpath. DNNs are pre-trained for the RCh, so the concatenated DNN representing the signal propagation can be very quicky composed by selecting individual DNN models from a repository. Such propagation increases the LI and NLI noise resulting in higher values for σ^I and σ^Q at the output of the DNNs, which can be related to QoT-related indicators like pre-FEC BER and SNR. For scalability reasons, the DNN models consider the propagation of the features for a subset of selected CPs only. Precisely, two exterior and two interior CPs are selected, and from them, a Constellation Reconstruction block reconstructs the features for the non-propagated CPs with high accuracy.

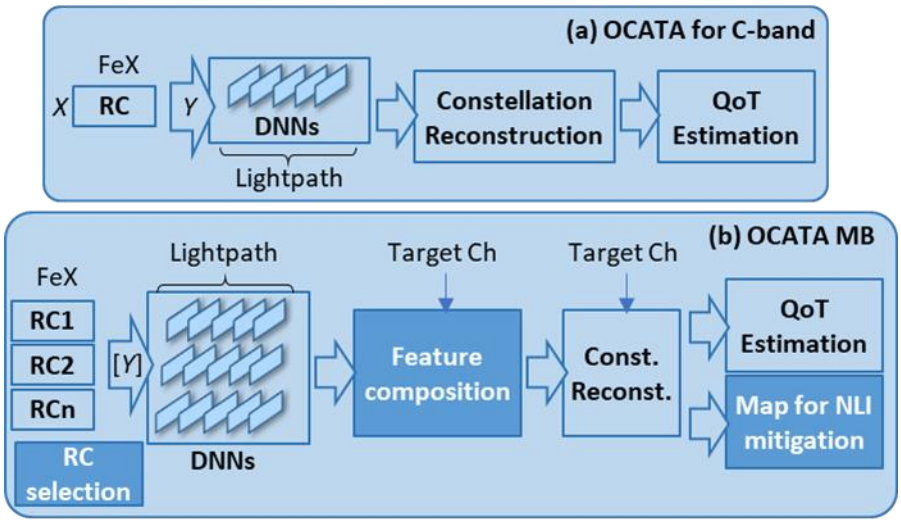


Figure 3.4-1: Main building blocks of the OCATA-MB time-domain digital twin.

The OCATA architecture for the C-band, however, is difficult to use directly in MB scenarios since the Inter-channel Stimulated Raman Scattering (ISRS) effect impacts the channels differently and therefore the number of pretrained DNN models in the repository will be too high. Because of scalability reasons, the architecture of novel OCATA MB (see Figure 3.4-b) considers a few RChs representing the specifics of different areas in the spectrum. Because OCATA models CP features propagation, the RChs are selected based on the different behavior of such features for selected CPs. To decide which channels become RChs, a RCh Selection procedure fairly selects them making a balance between the complexity of the models and the accuracy of the final results. However, the target channel is not necessarily one of the selected RChs. Therefore, OCATA MB includes the Feature Composition block to estimate the CP features for any channel in the C+L+S bands based on the output of the DNN models propagating the features of the RChs. The output of the Feature Composition block can then be used for QoT estimation as in the classical OCATA architecture. Note that the Constellation Reconstruction block is fed with the target channel to improve its accuracy in MB optical transmission.

Armed with the ability of estimating the QoT for different channel assignments in an efficient way, as well as with the possibility to improve the QoT of the signal by mitigating NLI noise, an algorithm for channel selection can be devised. After computing the set of available channels in the given route, the algorithm comprises the following steps: *i*) it determines whether there are available channels that can provide the required QoT given the characteristics of the route, specifically the total number of spans and their length; *ii*) if some channel can meet the required QoT, then it finds the optimal channel assignment; *iii*) otherwise, it selects the best channel assignment among the available channels, computes the optimal areas to mitigate the NLI noise as much as possible and determines whether that solution would provide the required QoT.

APIs and Interfaces for Integration

At the time of this deliverable OCATA is a callable module that receives inputs and computes the required outputs.

Status, features under development

A) QoT estimation

The set of basic features introduced above can be extended. One parameter that can be computed from the basic features is Φ_i^{out}, which computes the probability of receiving a symbol originally sent as part of CP *i*, out of the detection area assigned to that CP, represented as A_i. Φ_i^{out} is computed under the assumption that dispersion of symbols around CP *i* follows the bivariate Gaussian distribution characterized by Yi.

B) NLI mitigation

The NLI noise mitigation is carried out by optimizing the detection areas in the Rx side. The proposed method computes near-optimal detection areas for the CPs and estimates the resulting pre-FEC BER. The method is based on dividing the whole coordinate IQ plane (*A*) into k small square areas *a* to create a grid.

The proposed method consists in assigning each of the small areas *a* to the detection area Ai of the CP i with the highest probability. An algorithm computes a vector of detection areas [A_i], where each area defines a subset of small areas of the whole coordinate plane. Vector [A_i] can be then transformed into a map, so the Rx can easily decode the received symbols.

Validations and Intermediate Results

To evaluate accuracy of the optical performance estimation, Figure 3.4-a compares the pre-FEC BER calculated by numerical simulation with the one estimated by OCATA. The results obtained

for RCh 97 and two non-propagated channels, i.e., ch. 180 and ch. 310, are depicted for different transmission distances. We observe that the pre-FEC BER estimation for the three channels is very accurate, in line with the results of OCATA for the C band.

For illustrative purposes, Figure 3.4-b shows the optimized detection areas for channels 1, 150 and 337 after transmission along 6 spans. We observe different shapes of the detection areas for the different channels needed to mitigate the different impact of NLI noise mainly due to the ISRS effect.

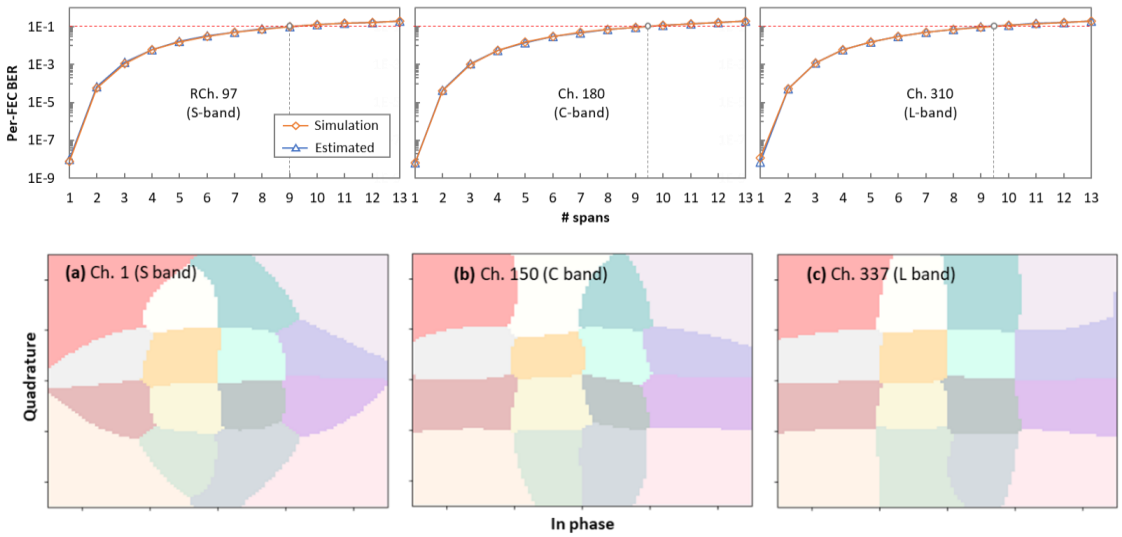


Figure 3.4-2: Evolution of Pre-FEC BER as a function of the number of spans for several channels (a) and Optimal detection areas for 5 spans (b).

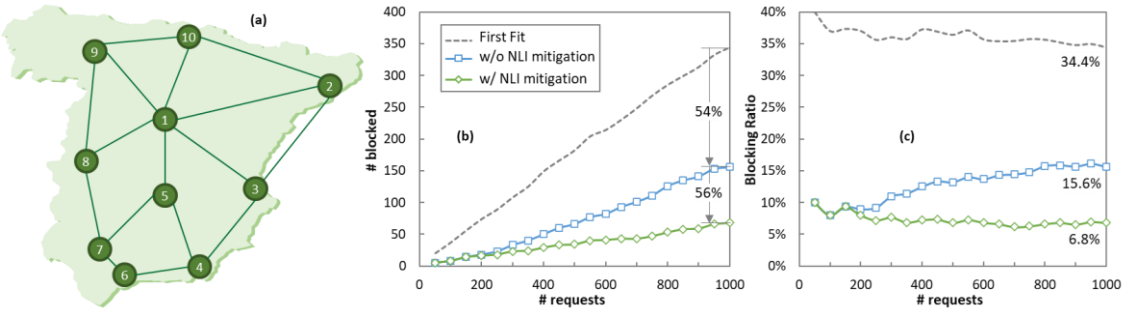


Figure 3.4-3: Spanish core optical network topology (a). Number of demands blocked (b) and blocking ratio evolution (c) vs demand number.

We have evaluated the performance of the OCATA-assisted for on-line MB-RSA. To this end, a simulation environment has been implemented in Python and the Spanish 10-node core network topology (Figure 3.4-a) with links with various configurations is assumed. In addition, to evaluate the performance of the NLI noise mitigation strategy, OCATA is used with and without NLI noise mitigation. For benchmarking purposes, the traditional first-fit channel assignment with QoT assurance is considered, where QoT is estimated for the selected channel and the request is rejected in case the required QoT is not met for that channel.

The simulation assumes a static traffic model, i.e., the simulation starts with no traffic on the network and connection requests are served sequentially between randomly chosen source and

Dissemination Level	PUB (Public)

destination nodes. Once a connection is successfully provisioned on the network, it stays without being changed or removed from the network. For the sake of simplicity, a set of 1,000 connection requests are generated beforehand, checking that the shortest route can provide the required QoT for at least one channel assignment. The simulator is then run for every considered approach with the same pre-generated list of connection requests. The number of shortest routes between each pair of nodes is limited to 3 in all the cases.

Figure 3.4-b shows the number of requests blocked whereas Figure 3.4-c depicts the blocking ratio as a function of the number of connection requests. We observe that the first-fit approach provides the highest number of connection requests blocked in all the cases because the channel selected cannot provide acceptable QoT, i.e., blocking is not a consequence of lack of available channels in the selected route. The final blocking ratio was as high as 34% after processing 1,000 connection requests. The number of requests blocked was highly reduced by 54% when the OCATA-assisted on-line MB-RSA algorithm was applied to find the best channel that can provide the needed QoT. In this case, the final blocking ratio reduced to 15.6%. Finally, when the NLI noise mitigation technique was also implemented, the number of requests blocked remarkable reduced by an additional 56% with a final blocking ratio of just 6.8%.

3.4.2 DevOps tools

Description:
As global internet demand intensifies amid a surge in 5G and advanced connectivity needs, network architectures are becoming increasingly complex. Managing a multitude of devices efficiently requires automated solutions. NetDevOps — the integration of development and operations in networking, addresses this need by automating configuration management and service provisioning. Following the SDN hierarchy, NetDevOps creates more efficient, flexible, and scalable networks, allowing organizations to leverage software-driven automation to meet growing demands. Currently, 95% of these configurations are performed manually, which often leads to policy violations due to human error, along with increased labour and operational costs. Implementing network configuration through programmability and automation can address these challenges by significantly reducing both operational costs and the time required to resolve issues. 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.

Existing approaches to network programmability, focus on integrating SDN and VNFs to automate service creation but often lack robust mechanisms for pre-deployment testing and efficient rollbacks, which are critical for reducing errors and minimizing downtime. Network management solutions using YANG data models and NETCONF, emphasize standardization and automated configuration. However, they primarily focus on uniform configurations, lacking flexibility for device-specific settings and efficient pre-deployment testing, often leading to complex, manual verification processes. The proposed solution git-based tracking, per-device configurations, and scalable parallel execution, address the aforementioned gaps by enhancing efficiency, reliability, and comprehensive pre-deployment testing through sandbox environments for optical networks.

Implementation:

The implementation leverages a NetDevOps approach to streamline complex network adjustments, improve deployment accuracy, and reduce manual errors through automation. A digital twin of the optical network is used that mirrors the actual infrastructure, enabling safe testing of configurations before deployment. By replicating devices and network conditions, engineers can validate changes, identify potential issues, and ensure everything works as expected. 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. 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 to the ONF-TAPI.

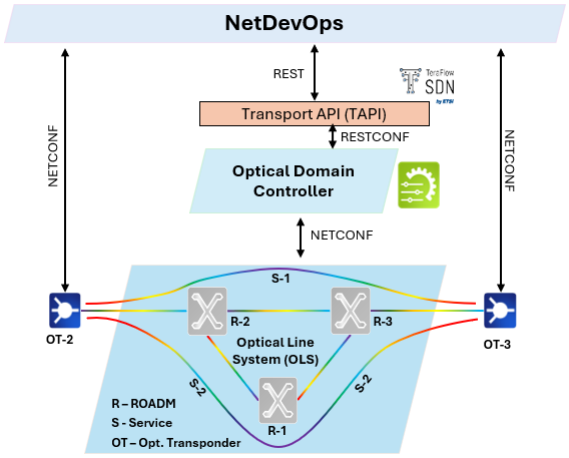


Figure 3.4-4 NetDevOps demonstration architecture showing service through Optical line system.

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.

(a). Configuration and Change Management: 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 to the 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 (Figure 3.4-).

(b). 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 (Figure 3.4-):

- Build Stage: Prepares the necessary environment, either through a virtual environment setup or by pulling a pre-configured Docker image.

- Change Detection: An Ansible playbook identifies modified files and matches them to specific network devices, allowing targeted configuration updates.
- Testing Stage: Checks device connectivity and verifies authentication using NETCONF or gNMI protocols. Adjustable parameters, like Ansible's fork setting, optimize simultaneous testing across multiple devices.
- Device Configuration: This stage deploys per-device configuration across network nodes using Ansible playbooks and Python NETCONF client (ncclient).
- 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.

(c). Security Compliance: 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 when errors occur, ensuring stability. Compliance with industry standards enables interoperability and seamless integration with network controllers.

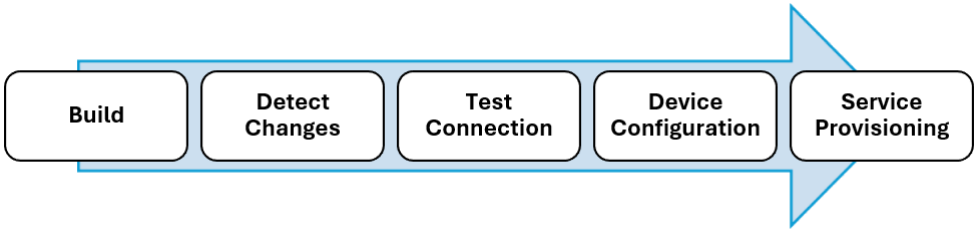
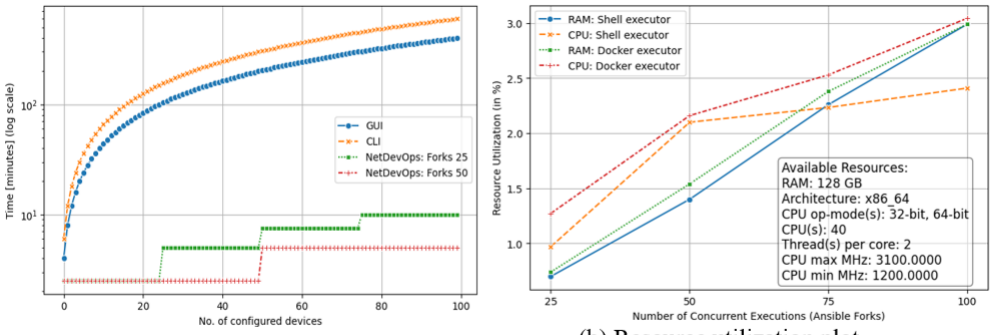


Figure 3.4-5: Implemented GitLab pipeline stages.

Evaluation:

To evaluate the pipeline's efficiency, we compared device configuration and service provisioning times using CLI, GUI, and the NetDevOps approach, focusing on scalability and resource utilization. Manual configurations via CLI and GUI are time-intensive and lack parallel processing, unlike the NetDevOps pipeline, which supports parallel execution. In tests with 100 devices, CLI averaged 6 minutes per device, and GUI averaged 4 minutes. Using the NetDevOps pipeline with Ansible's forks parameter, we tested 25 and 50 parallel processes, optimizing task execution based on available processing power (Figure 3.4-(a)). Resource utilization was analyzed for shell-based and Docker-based executors. The shell-based executor used less CPU and RAM but required more manual dependency management and relied on the host environment, while the Docker-based executor, though slightly slower, offered improved flexibility, security, and consistent performance across hosts due to containerization (Figure 3.4-(b)). This evaluation highlights a trade-off: the shell-based executor is faster but less flexible, whereas the Docker-based executor provides simplified management and greater reliability across varied environments.



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

Figure 3.4-6: NetDevOps performance evaluation plots.

APIs and Interfaces for Integration:

In Table 3.4-1 are presented the API/interfaces integrated within GitLab CLI and Python scripts invoked during the pipeline execution.

Table 3.4-1: API's Integrated to NetDevOps for Performing Configuration Operations.

API/Interface	Purpose	Integration Stage	Key Operations
NETCONF	Device configuration	Device configuration	<get-config>, <get>, <edit-config>, <edit>
RESTCONF	Service provisioning(TAPI)	Service provisioning	GET, POST, DELETE
Git	Version control	All stages(version control)	Commit, checkout, diff
MongoDB	Data persistence	Post- execution storage	InsertOne, Find
Ansible	Task automation and orchestration	All stages (pipeline orchestration)	ansible-playbook, dynamic inventory
TAPI	Optical service management	Service provisioning	Service create/delete, status check

Status:

- Future development plans include expanding device specific configuration support for Optical Transport Network (OTN) devices and Radio Units (RU's).
- Integration of Rollback mechanism to the current pipeline, for reverting to the last stable configuration when error occurs.

3.4.3 Distributed Intelligence and MAS Control

Description

Digital Subcarrier Multiplexing (DSCM) facilitates the deployment of P2MP optical connectivity, since subcarriers (SC) sourced from a single hub node can be assigned to different leaf nodes. In the reverse direction (denoted MP2P), SCs generated from different leaves can merge to connect the source nodes to the hub node. The number of leaf nodes in a P2MP connection can be increased with dynamic SC management, which can assign SCs dynamically to the leaf nodes, so those not requiring the full capacity of the transponder to support the local traffic can give

one or more of their SCs up to other leaf nodes with higher capacity requirements. This can result in power savings, since not every SC might be required to be active for the current traffic. For these gains to be fully realized, a control mechanism is necessary to ensure the proper P2MP connection operation, specifically from leaves to the hub (MP2P), to avoid oversubscriptions and to assure that the capacity needs of every leaf node are met.

We have developed distributed control of DSCM systems based on Multi-Agent Systems (MAS), i.e., a group of autonomous agents that interact among them and with their environment to fulfill some goal. Because intelligence and decision-making capabilities are brought down to each agent in the MAS, many considerations must be made, mainly regarding the design of the overall system (coordination, task allocation, agent interaction, and organization) and the specific design of the agents (learning, information exchanged, and interaction with the environment, including traffic prediction). In our case for the control of DSCM systems, intelligent agents anticipate traffic variations and guarantee that enough capacity was available. Note that the nature of reinforcement Learning (RL) makes it especially suited for dynamic traffic and capacity operation since the best policy is learned through experience. This also makes it compatible with MAS as models can be pretrained on similar traffic and retrained if traffic profiles evolve, as well as introducing flexibility since each agent can operate on a different model.

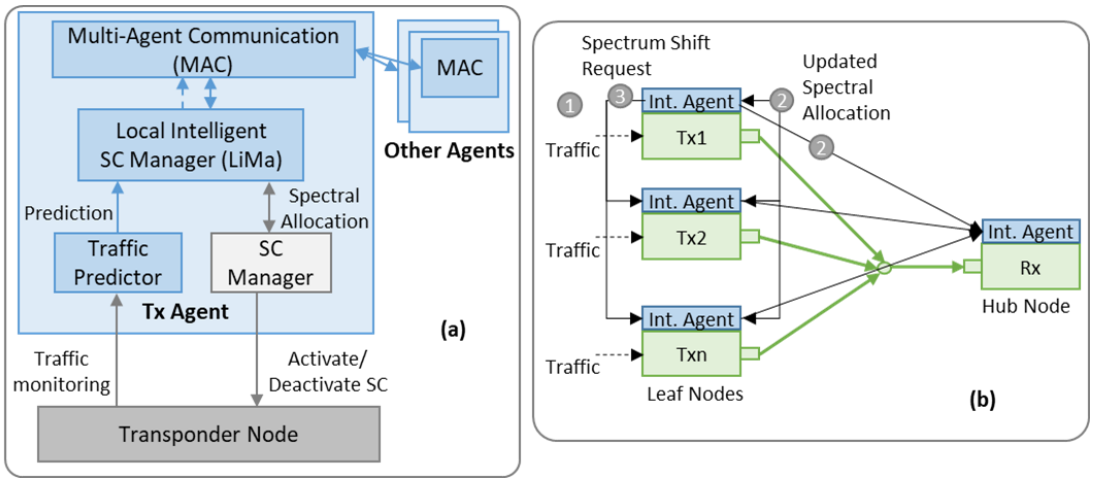


Figure 3.4-7: Agent design (a) and communication workflow (b) for distributed MAS control.

The architecture of the Tx agent in the distributed MAS approach is presented in Figure 3.4-a. Tx agents include the following modules:

- Traffic predictor module receives traffic monitoring from the transponder node and uses them for traffic prediction.
- Local intelligent SC Manager (LiMa) in charge of making local decisions for the spectral allocation.
- Multi-agent communications (MAC) module that coordinates operation with the rest of the agents in the MP2P system. The MAC module is in charge of coordinating spectrum operations with the neighboring Tx agents if applicable.

Similar to the Tx agents, the Rx agent also contains a MAC and an SC Manager that maintains a table <SC, Tx>.

The architecture in Figure 3.4-a needs to be complemented with the interactions among the agents in the MAS in Figure 3.4-b. Each agent collects traffic measurements (step 1) and uses them to predict the traffic and the capacity for the next time period. The agents then decide their own spectral allocation based on such prediction and from the spectral allocations previously shared by the agents and update their allocations to the rest of the agents when some change (SC activation/deactivation) is performed (2). However, it might happen that the spectral allocation cannot be extended to satisfy its capacity needs because the neighboring SCs are already part of the spectral allocation of other leaf nodes. To solve that, agents have coordination abilities to change spectral allocations among them. Thus, agents can ask other agents with neighboring spectral allocations to shift their spectral allocation in order to release one neighboring SC (3). Note that only one SC shift is allowed, which is implemented by activating one non-active neighboring SC, thus enlarging the current spectral allocation, and then releasing one SC on the opposite side of the allocation (make-before-break). We consider a collaborative scenario where spectrum shift requests are always accepted if possible. If the shift request cannot be fulfilled, the agent would simply wait for the next time interval and traffic loss will happen. Note that in this approach, messages are exchanged when some change in the spectrum happens, which limits the total number of messages sent for the sake of scalability.

APIs and Interfaces for Integration

The APIs and interfaces for integration are not defined at this stage. The component is currently used self-contained and in standalone mode.

Status, features under development

Implemented in a development environment since no access to real DSCM systems are available.

Validations and Intermediate Results

For performance comparison purposes, we have implemented a centralized approach.

Each approach has been evaluated at traffic loads in the range [70-95] % for 5, 6, and 7 leaf nodes in an MP2P connection. Figure 3.4- reports the resulting proportion of traffic loss with respect to the total traffic in the connection for a period of one day. The maximum load that every approach supports without any loss and with 0.1% loss is included in the inset tables.

Figure 3.4-(a) shows the percentage of traffic lost for 5 leaf nodes. We observe that the centralized approach produces traffic loss only when the total load reaches 83%, while MAS achieves 0 loss until the traffic load reaches 81%. Figure 3.4-(b) presents the results for 6 leaf nodes. In this case, the centralized method is able to maintain 0 losses up to load 80%, while the MAS supports 0 losses until traffic loads of 79%, showing low loss up to 83% traffic load. Finally, Figure 3.4-(c) shows the proportion of traffic lost for 7 leaf nodes, where similar results can be observed. Both approaches show higher losses at 90% and 95% loads for all number of leaf nodes.

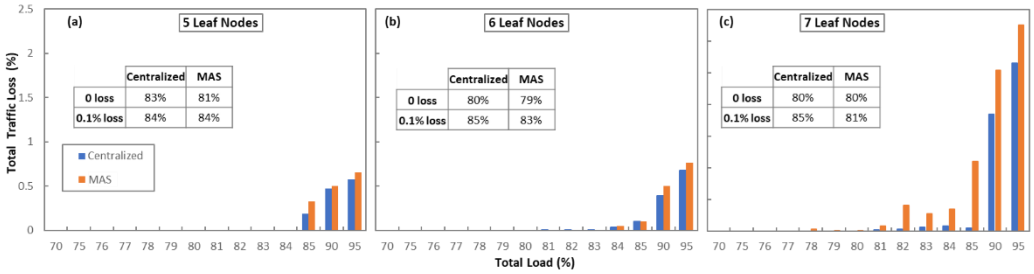


Figure 3.4-8: Proportion of traffic loss for different loads with 5 (a), 6 (b), and 7 (c) leaf nodes.

Table 3.4- shows the compounded number of spectra shifts for different traffic loads performed by each approach. In particular, shift operations occur when traffic is reaching the maximum load and resources are scarce.

Table 3.4-2: Number of spectrum shift operations performed by every approach.

Load	5 Leaf Nodes		6 Leaf Nodes		7 Leaf Nodes	
	Centralized	MAS	Centralized	MAS	Centralized	MAS
70	0	0	0	2	0	1
75	0	0	0	3	0	2
80	3	2	4	4	2	3
85	2	3	4	3	0	2
90	2	2	4	3	0	2
95	2	2	4	5	0	3

Table 3.4-3: Number of messages exchanged for 6 leaf nodes and 82% load.

Total # of messages	
Centralized	MAS
112.320	285

We also have analyzed the number of messages that are exchanged by each approach in a P2MP connection with 6 leaf nodes at a traffic load of 82%. The results are reported in Table 3.4- for the total number of messages along the day. The centralized approach where all the traffic measurements need to be conveyed to the centralized controller. In contrast, message exchange is needed when spectral operations are performed in the MAS approach. This results in a small total number of messages being exchanged and a more scalable approach.

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

3.4.4.1 Description

The integration of packet nodes (e.g., routers and switches) equipped with compact, pluggable coherent transceivers such as 400/800 ZR+ with wideband transport technologies like Elastic Optical Networks (EON) is crucial to cope with constant growth of broadband services [Dav24]. Such a transport infrastructure leads to simplify networks by eliminating traditional standalone DWDM transponders, improving the CapEx, and favoring the reduction of overall network power

consumption, i.e., enhancing the so-called energy-efficiency. Particularly, sustainability is becoming a major key goal for network operators that seek to improve the balance between data throughput (b/s) and power consumption (in W). To this end, *greener* technologies such as DWDM transceiver *pluggable* are considered being combined with advanced energy-aware routing and traffic engineering strategies [Cos21].

Optical Software-Defined Network (SDN) controllers increasingly incorporate autonomic control functions to manage optical connectivity services. Such operations and functions include Routing, Spectrum, and Modulation Assignment (RSMA) algorithms, monitoring/telemetry systems, and programmable APIs with robust protocols and standardized data models, e.g., NETCONF, OpenConfig, OpenROADM. In the context of fostering sustainable transport networks, SDN-based operations need to support and integrate energy-efficient provisioning and restoration mechanisms. Consequently, widely devised RSMA algorithms need to be improved to become energy-aware (EA-RSMA) strategies where besides optimizing paths, spectral resources, and transceiver configurations (Operational Modes – OMs- defined by symbol rates and modulation formats) also target the reduction of the network power consumption. EA-RSMA strategies must be designed to i) meet service demands (e.g., bandwidth) and satisfy EON constraints (spectrum continuity and contiguity) while ii) minimizing active elements like ROADMs, amplifiers, and transceivers. Traditionally addressed by heuristic RSMA algorithms, this complex problem now benefits from AI/ML methods like Deep Reinforcement Learning (DRL) [Mar24]. Herein, a DRL EA-RSMA model is explored, trained to dynamically manage connectivity requests and achieve superior energy efficiency (W/Gb/s) over traditional heuristics like EA-KSP-FF, validated through extensive performance tests under varying traffic loads and transceiver counts. The model processes connectivity requests with diverse bandwidth and latency requirements, with key performance metrics including: i) Blocked Bandwidth Ratio (BBR); ii) average energy efficiency; and iii) average number of active transceivers.

Adopted Network Power Consumption Model

Figure 3.4- illustrates the power consumption calculation for provisioning a connection across the path formed by R1-ROADM1-ROADM2-ROADM3-R2. The number of required DWDM coherent transceivers m is determined based on the requested service bandwidth (b/s). Various operational modes (OMs) are considered, shown in Table 3.4-, which provide different bit rates and maximum path distances based on aggregated link lengths. For each OM candidate supporting the necessary path distance, the number m of optical carriers/Frequency Slots required to meet the bandwidth demand are calculated, defining the transceivers at the endpoint nodes (R1 and R2). 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.

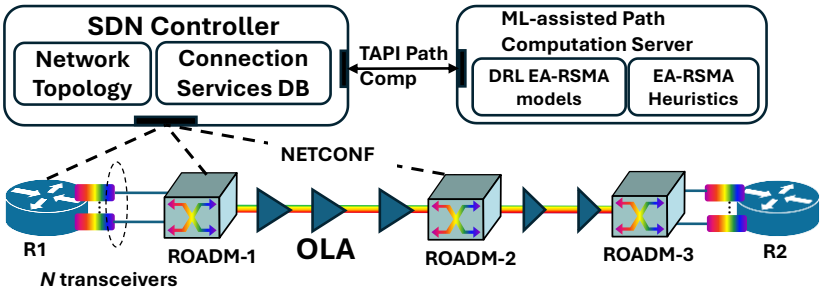
The router power consumption (P_{R_i}) using m transceivers for a specific optical connectivity service is given in Eq. 1 [Mos24], where P_{idle}^{router} denotes the *idle* power (router on, no traffic) and P_{OM}^{xcvr} represents the power consumption tied to a selected/configured OM in all the *m* transceivers. As depicted in Table 3.4-, higher OMs with complex modulation increase power dissipation in components like the ADC, DAC, and DSP, as higher data rates require more intensive signal processing. Power consumption from the Optical Line Amplifiers (OLAs) along fiber links is denoted by $P_{link_{i,j}}$, in Eq. 2.

Dissemination Level	PUB (Public)

Active ROADMs' power (Eq. 3) includes: i) idle power (P_{idle}^{ROADM}) associated to fans, control system, etc.; and ii) P_i^{OLP} being the power from active optical ports (e.g., WSSs, amplifiers, mux/demux) to interconnect to adjacent ROADM. Roughly, idle power of active nodes (routers and ROADMs) consumes 70-75% of the total network power. Notably, if a device (e.g., router, ROADM, port, or link) is already powered by an existing service, adding a new service does not increase power. Eq. 4 calculates the total power for devices activated along the connection path.

Considered Energy-Aware RSMA algorithms: Heuristics and DRL-based

An optical connectivity service request, R, is defined as the tuple $\langle s, d, b \rangle$, where s and d are the source and destination routers, and b is the demanded bitrate (b/s). Thus, when a new R is received, the applied RSMA algorithm searches for a feasible path, including the set of nodes and links, for a specific explored OM that allow meeting the R's bitrate. By doing so, the adopted RSMA algorithm determines the necessary m carriers/transceivers/frequency slots for the chosen OM, specifying for every optical flow its central frequency and slot width.
A common RSMA algorithm, KSP-FF, prioritizes the most advanced OMs and computes *K* solutions to minimize hop count while ensuring that all required carriers/optical flows meet end-to-end optical continuity and contiguity constraints. On the other hand, 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 [Mar24].



(Eq.1) $P_{R_i} = P_{idle}^{router} + m * P_{OM}^{xcvr}$

(Eq.2) $P_{link_{i,j}} = \sum P^{OLA}$

(Eq.3) $P_{ROADM_i} = P_{idle}^{ROADM} + \sum_i P_i^{OLP}$

(Eq.4) $P_{optServ} = P_{R_1} + P_{R_2} + \sum_i P_{ROADM_i} + \sum P_{link_{i,j}}$

Figure 3.4-9: Power consumption model for integrated packet nodes with 400G ZR+ pluggable and EON infrastructure.

Table 3.4-4: Considered power consumption parameters.

	Description				Value
P^{OLA}	Power Consumption of activated Optical Line Amplifier				12.0 W
P_{idle}^{ROADM}	ROADM environmental power consumption (e.g., control, fans, etc.) regardless of the connection services				150.0 W
P_i^{OLP}	ROADM optical line port power consumption				85.0 W
P_{idle}^{router}	Router environment power consumption				386.0 W
	OM	**Modulation Format**	**GBauds; Gb/s**	**Distance (km)**	**Power (W)**

P_{OM}^{xcvr}	1	DP-QPSK	30; 100	**3000**	**12**
	2	DP-QPSK	60; 200	**1500**	**15**
	3	DP-8QAM	60; 300	**1000**	**19**
	4	DP-16QAM	50; 400	**700**	**22**

To enhance the balance between network performance (e.g., BBR) and energy efficiency, we propose an offline-trained DRL model, DRL EA-KSP. The DRL agent, employing a Deep Neural Network (DNN), learns optimal decisions by interacting with the network environment, including packet and EON topology, optical spectrum, transceivers, and dynamic requests R. For each R, the *observation* (as shown in Figure 3.4-) includes two one-hot arrays for the source and destination nodes and a set of *OM*K* candidate paths. Each path is characterized by spectrum attributes, requested spectrum, and a vector indicating specific EON links across the k-path. The *action* space covers all feasible paths, up to *OM*K*. The reward function (Eq. 5) incentivizes actions that 1) increase network throughput via R's bitrate, and 2) minimize additional power consumption (*powerPath*) for a chosen path. If no feasible action exists (e.g., due to lack of available transceivers or EON resources), the reward is set to -10.

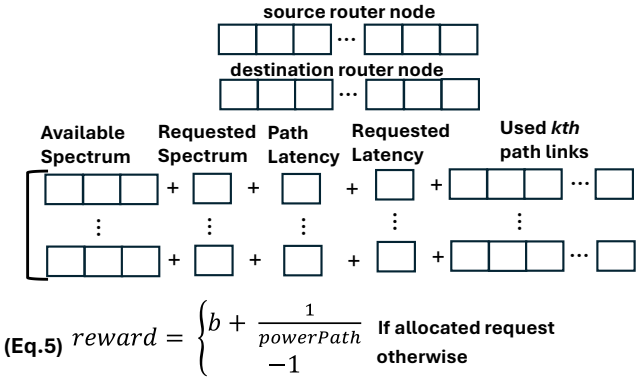


Figure 3.4-10: DRL EA-RSMA: observation space and reward.

3.4.4.2 APIs and Interfaces for Integration

S.No	Interface	Description
1	RESTCONF TAPI Models	The interaction between the Optical SDN controller and ML-assisted Path Computation Server (see Figure 3.4-is implemented via RESTCONF protocol using TAPI models. • TAPI DSR Path Computation (POST): Specifying SIPs and Bit Rate • TAPI Path Reply • TAPI Context Retrieval: TAPI Context is extended to provide power consumption information for *pluggables.* To do that, we leverage the TAPI Context Profile concept [Tapi]. Profiles provide static data information that can be referred to other TAPI object such as NEPs. It is augmented the transceiver profile for each OM listed in Table 3.4-.
2	SBI Netconf	Routers' and ROADMs' agents interact with the SDN controller using a NETCONF API (OpenConfig and OpenROADM data models) to program the DWDM transceivers and optical flows

Figure 3.4- (a) depicts the workflow conducted by the SDN controller to automatically process incoming connectivity service requests. The NBI, based on the ONF T-API and built upon RESTCONF protocol and YANG models, is exposed by the SDN controller to allow an external entity requesting the provisioning of a Digital Signal Rate (DSR) service. Then, the SDN controller initiates the path computation and resource selection in both packet and optical network elements. This process can be computationally intensive. Thus, the path computation function is externalized to a dedicated ML-assisted Path Computation Server to execute the routing algorithm. The output is formed by a spatial path (i.e., nodes and links) and resources meeting the service demands and constraints. The input of the RSMA algorithms comprises not only the connection requirements but also the Context. This context provides the network topology formed by routers, ROADMs, links attributes (e.g., distance), available optical spectrum, and transponder/pluggables capabilities (e.g., operational modes).

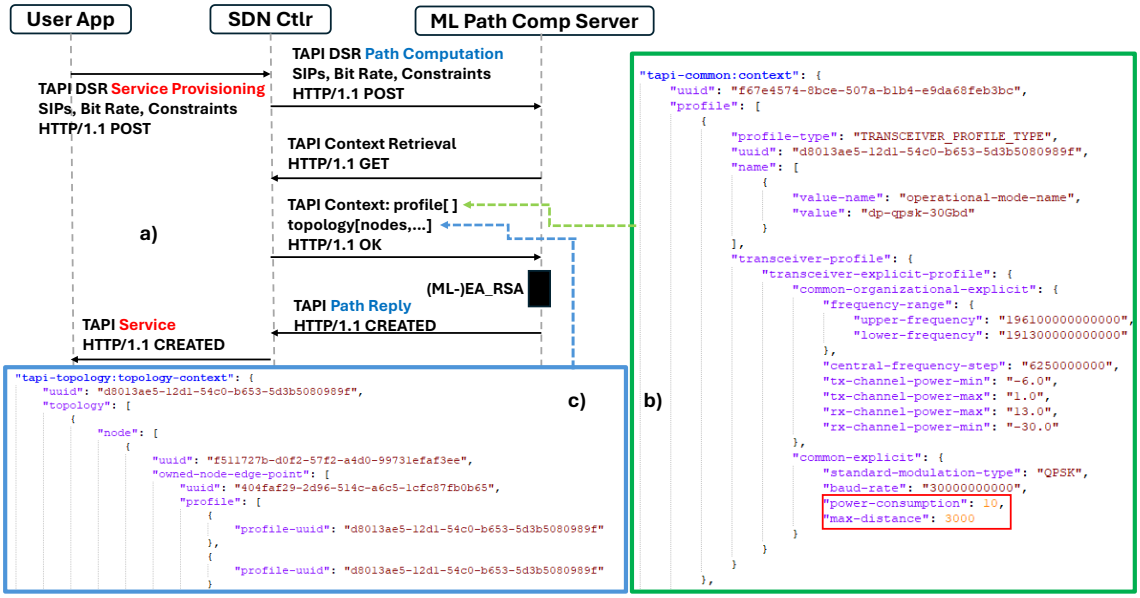


Figure 3.4-11: SDN Controller workflow; (b) TAPI Context Profile extensions to support power-consumption transceiver capabilities; (c) TAPI Context node w/ NEP's profiles.

The Path Computation Server polls the SDN controller to retrieve the *Context* sending a RESTCONF GET method. The TAPI *Context* contains the list of nodes specifying their Node Edge Points (NEPs) (see Figure 3.4- (c)). A pair of NEPs hosted in different nodes are inter-connected with a link. For each NEP, relevant attributes are detailed such as the available/occupied optical spectrum. For energy-efficient path computation, the TAPI Context is extended to provide power consumption information for DWDM transceivers. To do that, TAPI Context Profile concept is used. As shown in Figure 3.4- (b), for OM 1, the profile *dp-qpsk-30Gbd* is created which includes: the frequency range (i.e., upper, and lower frequencies) in Hz, the minimum and maximum powers for transmission and reception in Watts, the standard modulation format set to QPSK, the symbol rate in Gbaud, the power consumption in Watts, and the maximum supported distance in km.

3.4.4.3 Status, features under development

At the time of writing, the following describes the status of the implementation and conducted validations:

- Standalone ML-Assisted Path Computation Server is currently being validated by interacting with the optical SDN controller using TAPI
- The EA-aware RSMA algorithms, i.e. heuristics such as KSP-FF and EA-KSP-FF and DRL EA-KSP have been deployed and exhaustively evaluated considering different metrics such as the BBR, the energy-efficiency, and average transceiver utilization.

Upcoming enhancements will be addressed in the context of multiband with close interaction with WP3 activities. The idea is to extend not only the power model for other bands than C, but also, the algorithms (heuristics and DRL-based). This enhancement will require determining the devices and their power consumption involved in other optical bands such as transponders, amplifiers, etc.

3.4.4.4 Validations and Intermediate Results

This section focuses on describing the obtained performance when adopting the above-mentioned algorithms: KSP-FF, EA-KSP-FF and DRL EA-KSP. For the DRL EA-KSP model, it is trained using a maskable Proximal Policy Optimization (PPO) with 3 fully connected hidden layers (128 neurons each), a learning rate of 10^{-5}, and a discount factor of 0.95. Training spans 1000 episodes, each with 20,000 *R*s generated by a Poisson process with inter-arrival time set to 50 s. Routers are equipped with 15 DWDM transceivers. Holding times (HT), modeled exponentially, are 2000-3500s. Requested bitrate b is randomly selected from [100, 200, 300, 400] Gb/s. The network topology, shown in Fig. 2(d), labels optical link distances on each edge. Spectrum ranges from 191.3 to 196.1 THz, comprising 768 Nominal Central Frequencies (NCFs) spaced at 6.25 GHz. Carrier slot width is set at 50 GHz (8 NCFs) across all modulation formats, and OLAs are spaced every 80 km along fiber links. For all the RSMA algorithms *K* is set to 3.

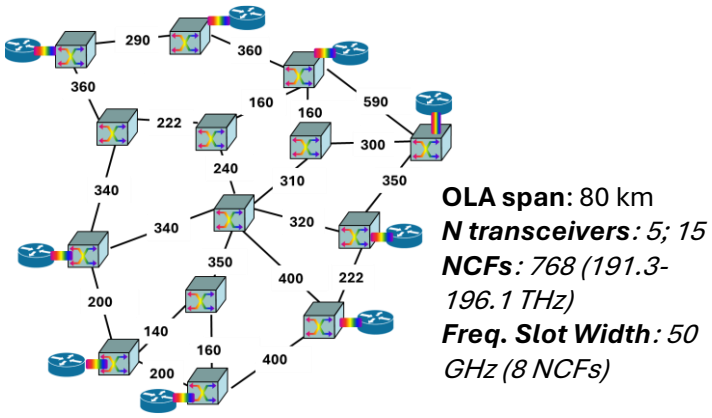


Figure 3.4-12: Router - EON transport topology & network characteristics: OLAs spacing, NCFs, Frequency Slot Width.

Figure 3.4-(a) compares the BBR of three RSMA algorithms. KSP-FF achieves the lowest BBR across all traffic loads, focusing on maximizing service accommodation (R) through efficient spectral resource use without prioritizing power consumption. However, BBR must be balanced with energy efficiency, calculated as average network power consumption (kW) per average served throughput (Gb/s). As shown in Figure 3.4-(b), EA-KSP-FF improves energy efficiency by 7.5% to 8.8% over KSP-FF by routing R through active ROADMs and links (e.g., OLAs) and favoring transceivers with lower-power OMs. This approach, however, often requires longer paths, increasing hop counts and spectral resource use, which complicates EON spectral constraints and slightly degrades BBR relative to KSP-FF.

The DRL EA-KSP model is designed to enhance EA-KSP-FF's energy efficiency. It achieves 2.3% to 4.8% higher energy efficiency than EA-KSP-FF, depending on traffic load, but at the cost of a higher BBR, with a degradation of 31.6% under low traffic load and 13.42% under high load. DRL EA-KSP prioritizes energy efficiency by favoring lower-power OMs, resulting in transceivers programmed with lower-bit-rate, less advanced modulation formats. Consequently, DRL EA-KSP requires more transceivers, as seen in Figure 3.4- (c), impacting BBR because: i) fewer transceivers are available, and ii) more optical flows must meet EON continuity and contiguity constraints. Some connections may fail to be established due to transceiver unavailability or the model's difficulty in fulfilling EON spectral constraints, revealing a trade-off between BBR and energy efficiency.

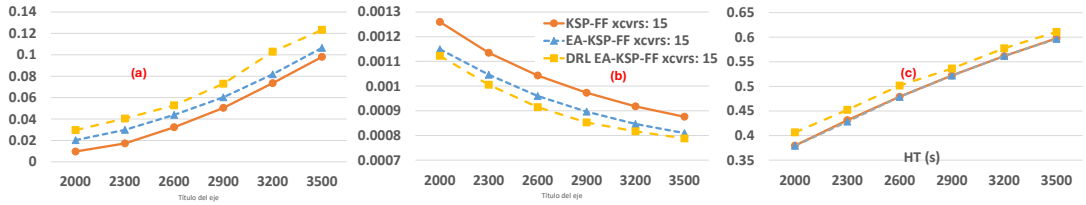


Figure 3.4-13: Figures of merit: (a) BBR; (b) Energy Efficiency (kW/Gb/s); (c) Average network transceiver utilization.

3.4.5 Latency-aware RSA based on DRL

Elastic optical networks (EONs) offer enhanced flexibility and efficiency in managing optical spectrum resources. Unlike traditional fixed-grid networks, EONs allocate variable-sized spectrum widths to accommodate diverse traffic demands. This adaptability enables EONs to support a wide range of applications (e.g., autonomous vehicles, telemedicine, and industrial automation) requiring high bandwidth and low latency. Furthermore, EONs benefit from software-defined networking (SDN) principles, allowing for centralized control and automation of network functions.

A fundamental challenge in EONs is the routing and spectrum assignment (RSA) problem. The RSA problem involves finding an optimal path through the network and assigning a contiguous optical spectrum along that path to satisfy a connection request. This task must consider several constraints, including spectrum continuity and contiguity, as well as meeting the requirements of the requested service (i.e., bandwidth and latency). Due to its complexity, RSA is classified as an NP-hard problem, meaning that finding optimal solutions can be computationally intensive. Traditional RSA algorithms, often based on heuristics, may not always find the most efficient solutions, particularly in dynamic network environments with fluctuating traffic patterns.

Machine learning (ML) techniques, and in particular, deep reinforcement learning (DRL), have emerged as promising approaches to address challenges in optical networks, including the RSA problem. DRL algorithms can learn optimal strategies by interacting with the network environment and receiving feedback in the form of rewards. In the context of RSA, DRL agents can adapt to changing network conditions and traffic demands, leading to more efficient spectrum allocation and improved overall network performance.

A novel DRL-based RSA solution that explicitly considers latency constraints in addition to spectrum allocation is proposed. This approach involves training a DRL agent to make path and spectrum allocation decisions that minimize bandwidth blocking while meeting the latency requirements of connection requests. The agent learns by interacting with a simulated network

environment, receiving rewards for successful path establishments that meet both bandwidth and latency constraints.

SDN Control Architecture

A DRL agent is integrated in the SDN control plane to efficiently provision end-to-end connections within an EON, as depicted in Figure 3.4-14(a). The architecture leverages a centralized SDN controller responsible for the complete service lifecycle within EON. This includes processing new connectivity service requests, finding and allocating available optical spectrum resources, configuring the necessary network elements like ROADMs, and finally, releasing those resources upon service termination.

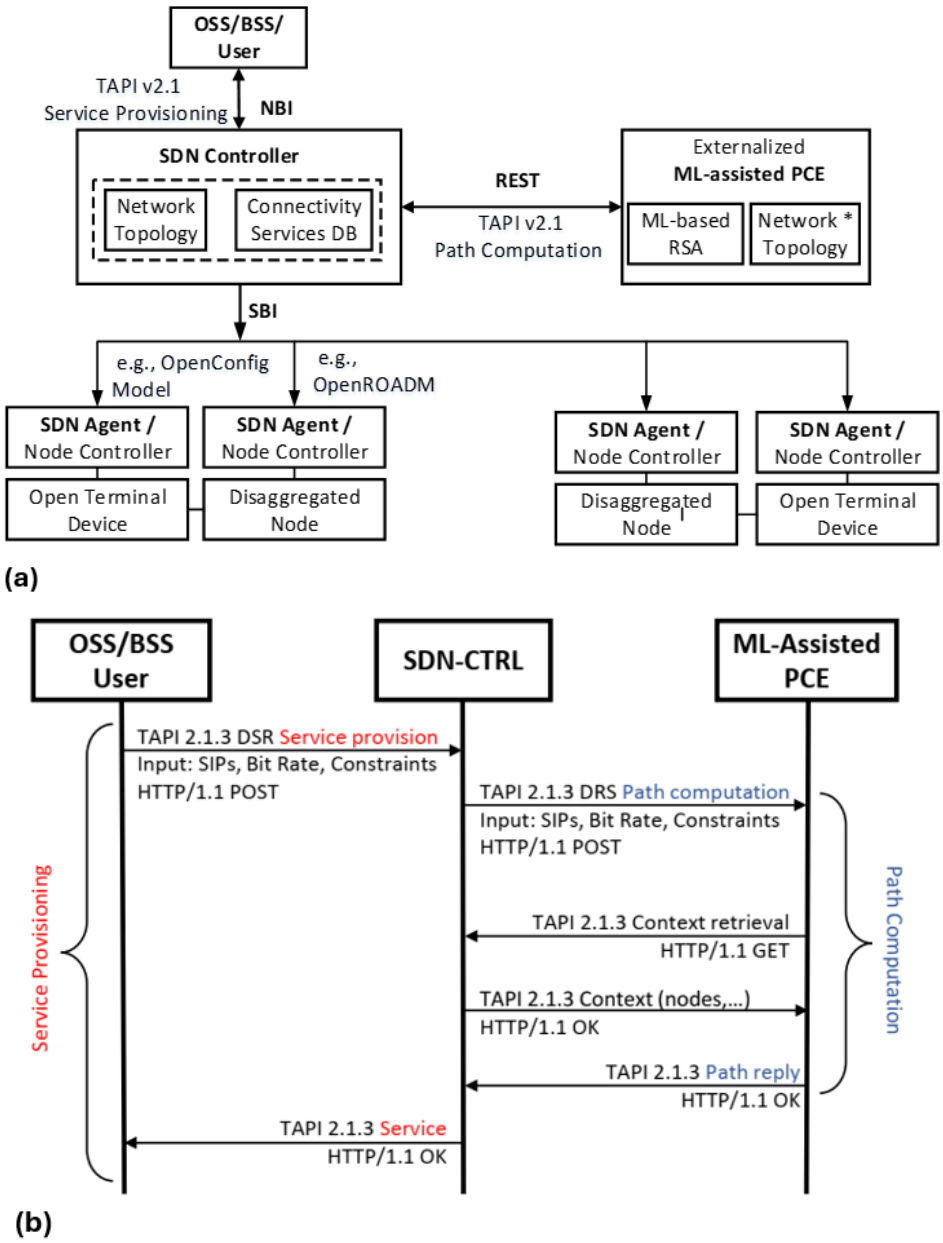


Figure 3.4-14: (a) SDN control plane architecture, and (b) Service provisioning workflow.

Dissemination Level	PUB (Public)

The SDN controller interacts with external entities and network elements through two key interfaces: the northbound interface (NBI) and the southbound interface (SBI).

- Northbound Interface (NBI): This interface facilitates interaction with operations support systems (OSS), business support systems (BSS), or end-users. The NBI allows these entities to request new connectivity services and receive updates on the network status. The specific implementation relies on the ONF Transport API (T-API) version 2.1. This API utilizes open and standardized data models defined in YANG, communicated via the RESTCONF protocol over HTTP. T-API establishes a client-server relationship where the SDN controller acts as the server and provides access to a shared context containing information about the network topology, available resources, and active services.
- Southbound Interface (SBI): This interface enables communication between the SDN controller and individual network elements such as optical transponders and ROADMs. The SDN agent within each network node uses the SBI to interact with the local node controller and program the allocation or deallocation of specific resources, such as frequency slots on specific ports.

To handle the complex task of RSA, the proposed SDN control architecture externalizes the path computation function to a dedicated path computation element (PCE). This PCE hosts a trained DRL agent responsible for making intelligent decisions regarding RSA, forming a ML-assisted PCE. This modular design provides numerous benefits, including increased flexibility, enhanced algorithm innovation, and dedicated resource allocation for computationally intensive processes.

The PCE acts as a server, exposing a REST API to facilitate communication with the SDN controller acting as a client. Upon receiving a path computation request from the SDN controller, the PCE retrieves the updated network state information from the controller's T-API context. This context includes details about the network topology, active connections, and available resources, allowing the PCE to have an accurate picture of the current network conditions. The PCE uses this information to compute a feasible path and allocate appropriate resources using its trained DRL agent. Once the DRL agent in the PCE computes the optimal path and resources for the requested connection, it communicates this information back to the SDN controller. The SDN controller then uses the SBI to configure the involved network elements and establish the desired lightpath. This workflow is illustrated in Figure 3.4-14(b).

This DRL-based RSA approach implemented in the PCE, offers a more intelligent and efficient solution for managing network resources and provisioning connectivity services in EONs. The modular design and standardized interfaces ensure flexibility and adaptability, allowing for future enhancements and integration with other solutions.

DRL Agent for latency-aware RSA

The DRL agent is trained offline to make optimal decisions considering both the constraints of the optical transport technology and the required service specifications, this means fulfilling both spectrum and latency constraints. This training utilizes the Asynchronous Advantage Actor-Critic (A3C) algorithm, which improves learning speed by running multiple agent threads in parallel. Each thread interacts with its copy of the network environment and updates a global policy. This training process aims to produce a well-trained model that can effectively make decisions in real-time during service provisioning.

Key aspects of the DRL agent's design and operation include:

Network State Representation: The DRL agent utilizes a specific representation of the network state as input for its decision-making process. This representation combines elements from previous DRL-based approaches and incorporates latency information for improved performance. The state representation comprises:

- Spectrum Utilization: This information is captured for each candidate path connecting the source and destination nodes and includes details like:
  - The number of available frequency slots (FSs)
  - The required number of FSs for the connection request
  - The average size of available FS blocks
  - The position of the first available FS block
- Latency: This includes:
  - The maximum tolerated latency specified in the connection request
  - The end-to-end delay for each candidate path

Action Space: The DRL agent has a set of discrete actions, each representing the selection of one of the k pre-computed candidate paths between the source and destination nodes. The agent chooses the path that it believes will best satisfy the connection request while optimizing network resource utilization.

Reward Function: The reward function guides the DRL agent to maximize the cumulative reward over time. The function awards actions that allocate higher bandwidth and prioritize lower-latency connections, promoting an optimal balance that enhances overall network performance. It is defined as follows:

- A reward proportional to the sum of a component proportional to bandwidth demand and an inverse component of latency is given if the connection request is successfully allocated, encouraging efficient and latency-aware resource allocation.
- A negative reward of -10 is assigned if the allocation fails due to insufficient spectrum resources or latency constraints, discouraging the agent from making these decisions.

$$R = \begin{cases} b + \frac{1}{l}, if\ sucessful\ allocation \\ -10, \qquad\qquad otherwise \end{cases}$$

Deep Neural Networks (DNNs): The DRL agent utilizes DNNs to process the network state representation and make decisions. The DNNs consist of multiple layers of interconnected neurons that learn complex patterns and relationships within the data. The specific architecture used includes five fully connected hidden layers. The number of neurons in the input layer is determined by the network topology and features of the candidate paths, while the output layer has k neurons representing the probability of choosing each candidate path.

The DRL agent undergoes a training phase where it learns to make optimal decisions by interacting with the network environment and receiving rewards. Once trained, the DRL model can be saved and used for online provisioning of connection requests. However, it is important to note that the agent needs periodic retraining to adapt to changes in network dynamics and traffic patterns.

APIs and Interfaces for Integration

S.No	Interface	Description
1	RESTConf POST TAPI	Path Computation (SDN Controller -> PCE) This includes path computation request and reply specifying the Service Interface Points (SIPs), Bir Rate, and topology constraints The supported paths include aspects related to TAPI context discovery (including dynamic discovery of SIPs), topology (node, link, edge points) discovery, service (either in digital signal rate or photonic media/media channel) provisioning and deletion and path computation. GET Context (PCE -> SDN Controller) This includes aspects related to TAPI context discovery (including dynamic discovery of service interface points), topology (node, link, edge points) discovery, service (either in digital signal rate or photonic media/media channel). This operation can be done synchronously by performing polling to the SDN controller. Path Delete Context (SDN Controller -> PCE) This includes notifying the PCE server that a service has been deleted which entails deleting the associated Path and resources in the TED.

Status, features under development

All necessary interfaces for communication between the PCE server and the SDN controller have been fully implemented, enabling seamless interaction for path computation. This includes the use of a RESTful API based on the ONF T-API 2.1.3 standard, facilitating the exchange of information such as network topology, available resources, and active connectivity services.

A key feature for future enhancement is the integration of the PCE server with a telemetry system based on MQTT. This integration would enable the asynchronous exchange of updated topology information, ensuring the PCE server has access to the most current network state for path computation.

Validations and Intermediate Results

The DRL-based latency-aware RSA approach was evaluated testing it on two different EON topologies: a metropolitan network in Barcelona City Network (BCNNet) and a national core network across Great Britain (BTNet), shown in Figure 3.4-. BCNNet includes 14 nodes and 42 bidirectional links, while BTNet encompasses 22 nodes and 70 links, as shown in Figure X. Both networks use identical frequency grids supporting 100 Frequency Slots (FSs) on optical fiber links. The dynamic service generation model assumes that connection requests arrive between random node pairs based on a Poisson process with a fixed inter-arrival time (λ) of 2 seconds. The duration of service for connections is modeled with an exponential distribution where the average service duration (μ) can be modified to create different traffic loads for testing. The model also considers various bandwidth demands and latency requirements for connections. Bandwidth demands are evenly distributed between requests for two, four, and eight frequency slots (FSs). Each FS represents a bandwidth of 12.5 GHz, which assumes a spectral efficiency of 1 b/s/Hz with BPSK modulation. Latency requirements are uniformly distributed between the shortest average path delay between all node pairs and twice that value. For BCNNet, latency varied between 0.30 ms to 0.60 ms, while for BTNet, it ranged from 1.25 ms to 2.5 ms.

Figure 3.4-15: Topologies used for the experimental evaluation.

The performance of the proposed DRL-based RSA was evaluated against the traditional k-Shortest Path First Fit (kSPFF) heuristic. This comparison specifically examines the Bandwidth Blocking Ratio (BBR), which measures the proportion of bandwidth requests that are blocked due to a lack of available resources on the BCNNet and BTNet topologies, both subjected to varying traffic loads. To train the DRL agent, the traffic load was set to 200 Erlangs. Figure 3.4- (a) shows that, as the number of training episodes for the DRL model increased, a noticeable improvement in BBR was observed compared to the kSPFF algorithm. This indicates the DRL model's capability to learn and optimize its decisions over time, progressively enhancing its decision-making policy to better accommodate diverse network conditions and connectivity demands.

The results shown in Figure 3.4- (b) reveal a clear advantage of the DRL-based approach over kSPFF, particularly as the traffic load increases. Under low traffic conditions (e.g., 50 Erlangs), both methods successfully accommodate all connection requests, resulting in a BBR of zero. However, as the traffic load rises, the limitations of kSPFF become apparent. The DRL agent consistently demonstrates a lower BBR compared to kSPFF, signifying its superior ability to efficiently allocate spectral resources. For instance, at a traffic load of 150 Erlangs on the BCNNet topology, the DRL agent achieves a BBR reduction of approximately 16.20% compared to kSPFF. This improvement becomes even more pronounced at higher loads, reaching an 18.35% reduction at 250 Erlangs. Similar trends are observed for the BTNet topology, indicating that the DRL agent's effectiveness extends across different network scales. This enhanced performance is attributed to the DRL agent's capacity to learn and adapt to varying network states and traffic dynamics. Through continuous training, the agent develops an optimized policy for path selection and spectrum allocation, maximizing resource utilization and minimizing blocking. In contrast, kSPFF relies on a simpler, less flexible heuristic that becomes increasingly inefficient as resource contention intensifies.

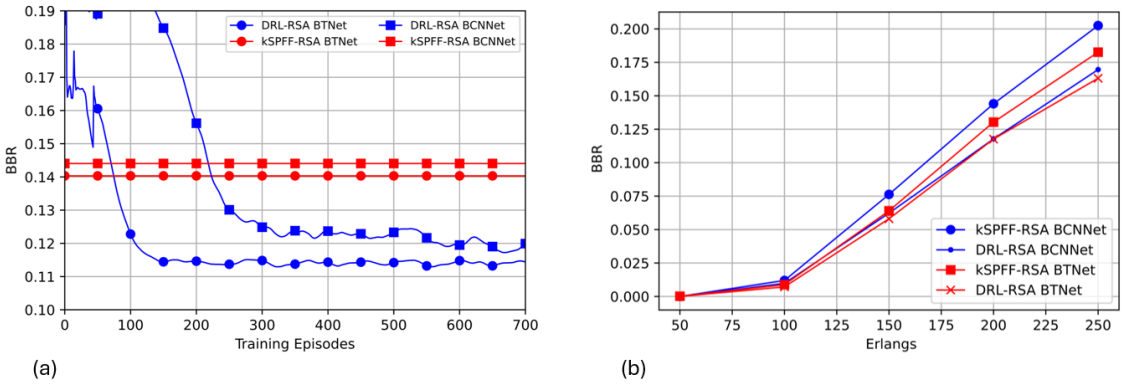


Figure 3.4-16: (a) Training phase of DRL RSA; (b) Performance benchmarking of kSPFF and DRL RSA.

In addition, the proposed DRL-based RSA solution for an EON was deployed in a proof of concept (PoC) using Docker containers to emulate a real-world optical network scenario. This implementation showcases a practical approach to integrating DRL-based RSA solutions into an SDN-controlled EON. The PoC showcases the interaction between the OSS/BSS, SDN controller, PCE, and the trained DRL model, to manage the provisioning of connectivity service requests autonomously.

The PoC, depicted in Figure 3.4-, includes the following key components:

- SDN Controller: The SDN controller acts as an Optical Line System (OLS) Controller and manages the overall service lifecycle within the emulated EON. This includes processing new service requests, allocating the necessary resources, configuring the network elements, and releasing those resources when a service is terminated.
- ML-Assisted PCE: The PCE hosts the trained DRL model and is responsible for making RSA decisions. It receives path computation requests from the SDN controller, retrieves the latest network topology and resource information from the controller, and uses the DRL model to determine the optimal path and spectrum allocation for the requested connection.
- Service Connection Generator: This component acts as an OSS and generates dynamic service connection requests that arrive and depart based on a predefined traffic load pattern. It interacts with the SDN controller to initiate new service connections and signal service termination.
- Graphical User Interface (GUI): The GUI provides an interactive way to request service provisioning and visualize the network's current state.

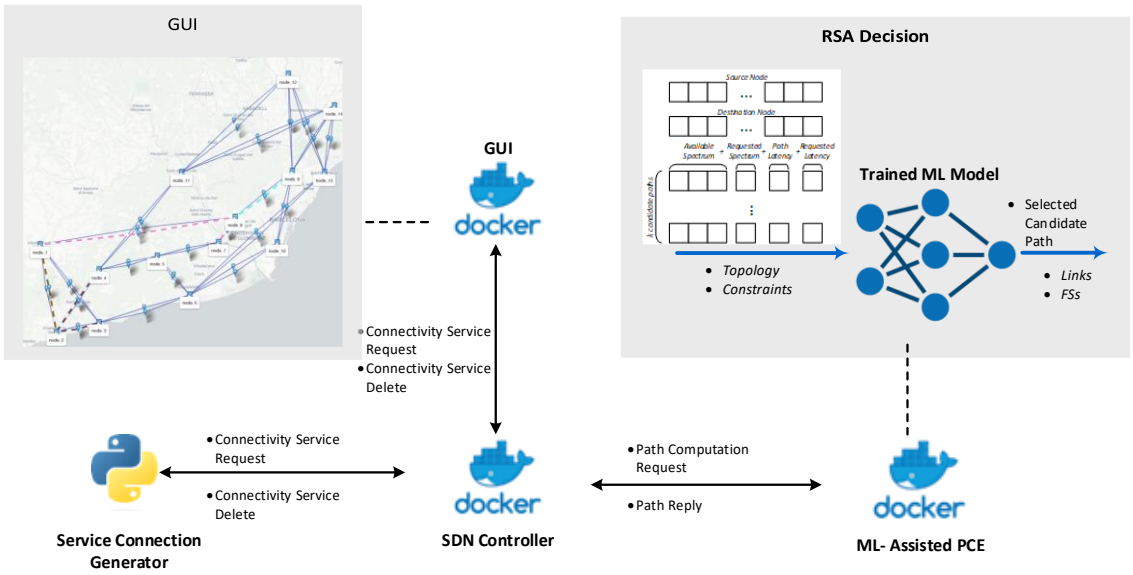


Figure 3.4-17: Illustrative layout of the PoC components.

4 TELEMETRY SYSTEM

4.1 TELEMETRY SYSTEM BASED ON MQTT

Description

Telemetry streaming and monitoring are essential for modern optical networks, enabling the collection of real-time performance data from various optical network elements and enabling more effective management strategies since operators can proactively address network issues, optimize resource allocation, and improve overall network reliability. This real-time data is critical for various applications, including training and updating machine learning (ML) models for network automation tasks like quality of transmission (QoT) estimation, traffic prediction, and resource allocation.

Telemetry streaming offers a more effective approach to network monitoring than traditional methods that rely on periodic or on-demand full snapshots of network status. Telemetry streaming provides a continuous flow of data, allowing for a comprehensive view of network dynamics and enabling the detection of subtle changes in performance that might be missed with less frequent updates. Consider a scenario where you need to know the current status of network resources and elements. Traditionally, this would involve requesting complete information from the SDN controller. However, with telemetry, SDN controllers can generate events corresponding to specific changes in network configurations. This event-driven approach means that only the information about updated network elements is transmitted, significantly reducing the amount of data exchanged. In addition, the event-driven nature of telemetry ensures that updates are transmitted immediately, providing a real-time view of network configurations. Moreover, focusing on specific changes allows for more efficient data processing and analysis, leading to quicker responses and improved network management.

For instance, when an SDN controller creates or deletes connections, it generates some events that describe the specific changes, such as a new connection establishment, the lightpath associated to this connection with the nodes and spectrum resources. This events data is then transmitted to the monitoring system, providing precise information about the updated network elements without requiring a full network status report. This targeted approach to data exchange is particularly important in large and dynamic networks where frequent configuration changes are common. The benefit of this approach is that network topology and the status of network elements and resources can be maintained based solely on these telemetry messages generated by the SDN controller.

The CTTC FlexOpt SDN controller is capable of exporting telemetry data using MQTT for simple topology synchronization. To take advantage of this, an MQTT subscriber was created. The subscriber processes received MQTT messages from the controller and uses this telemetry data to build and dynamically update a network topology database (TED). This TED is then exposed to an AI/ML Path Computation Server that uses TAPI models for path computation. Importantly, 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. This asynchronous topology updating allows the AI/ML path computation server to operate with a current representation of the network, leading to more accurate and effective path computation decisions. The interaction described above is shown in Figure 4.1-1.

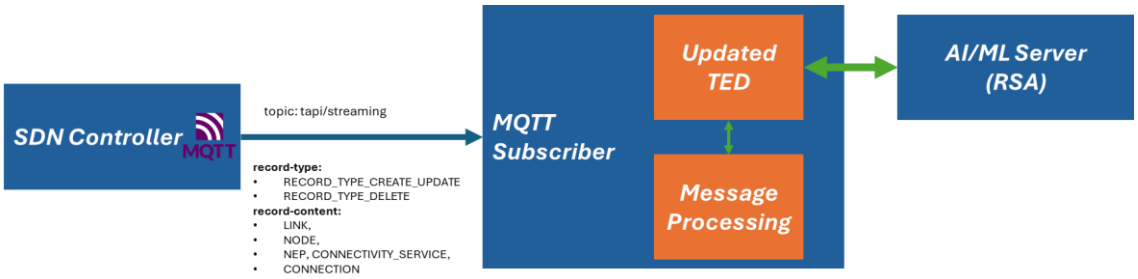


Figure 4.1-1: MQTT-based Telemetry.

APIs and Interfaces for integration

S.No	Interface	Description
1	Telemetry	The MQTT subscriber receives, and processes Telemetry records exported by the SDN Controller. Depending on the record-content (LINK, NODE, NEP, CONNECTION, CONNECTIVITY_SERVICE), the messages are processed to build and keep an updated topology.

Status, features under development

The MQTT subscriber has been successfully developed to process telemetry messages exported from the SDN controller. This functionality allows it to build and update a comprehensive topology database, ensuring that the network's topology and optical resources are accurately represented in real-time.

Subscriber capabilities will be extended to include the ability to build and maintain an updated database of active connectivity services. By tracking changes in the network topology and the status of active connectivity services, the control plane will be provided with a more holistic view of the operational status of the network. This information can be used by the Path Computation Server to further optimize network performance and resource allocation.

Validations

As mentioned before, the MQTT subscriber has been developed to receive telemetry messages exported from an SDN controller. Its functionality has been validated through extensive testing with the SDN controller, which exports telemetry messages in response to specific network events. When a new connection request is received or an existing connection is deleted, the SDN controller generates telemetry messages that trigger updates to various elements of the network topology. These messages indicate changes in the TAPI topology context within the SDN controller, such as updates to LINKS, NODES, and Node Edge Points (NEPs), as well as the creation or deletion of CONNECTIONS and CONNECTIVITY SERVICES. This dynamic interaction ensures that the MQTT subscriber maintains an accurate and up-to-date representation of the network topology and optical resources.

Figure 4.1-2 shows how the spectrum usage of a link is dynamically updated based on real-time telemetry messages received asynchronously from the SDN controller. These messages are generated only when there is a change in the topology, such as the establishment of a new connection, rather than polling the entire topology information periodically. This approach significantly reduces the amount of data transmitted from the SDN controller.

The MQTT subscriber processes these specific telemetry messages to update the spectrum usage of the affected link, reflecting the current state of the network.

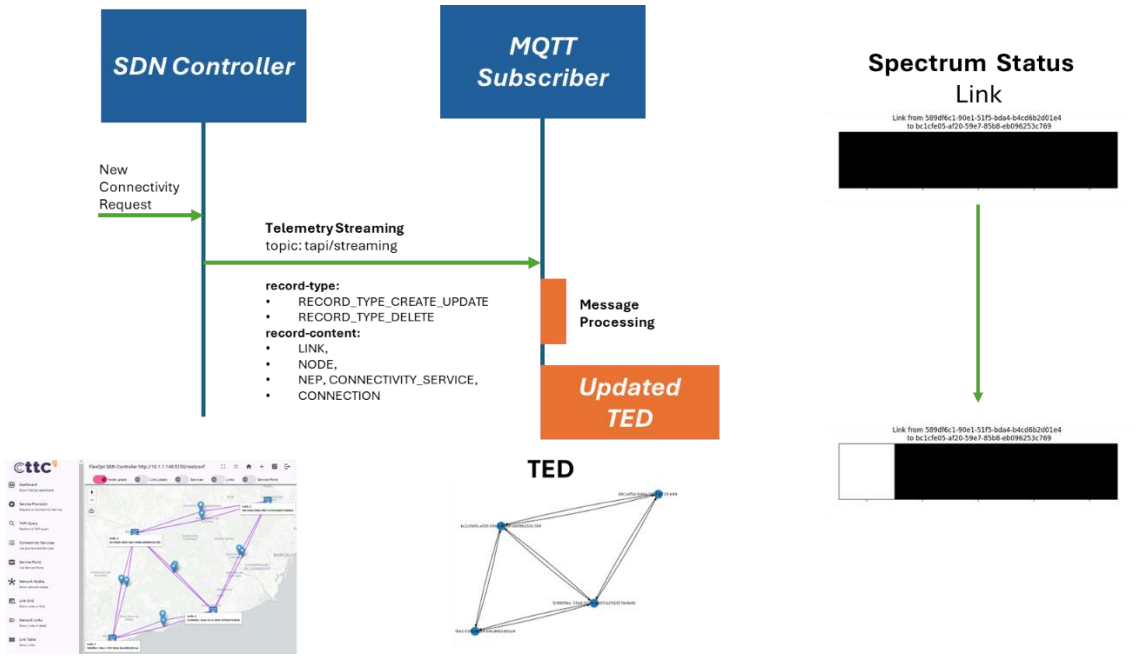


Figure 4.1-2: Real-time TED updating.

4.2 END-TO-END GRPC

Description

Performance data streaming in the network plays a vital part for many applications like monitoring, network management, closed loop automation, and much more. This is getting complex when it comes to optical transport networks, where streaming of performance data from multiple domains is required to manage the end-to-end optical service. Therefore, a systematic approach is required for telemetry data streaming from optical domain controllers to facilitate its utilization to northbound applications.

The ONF TAPI v2.5 is proposed with a telemetry streaming improvements and recommended respective YANG models and protocols to effectively support streaming of performance data from multiple domains. Following the TAPI v2.5 performance data streaming, it allows different data consumers to access the data from multiple domains with a unified data format. This systematic approach of telemetry data collection from multiple domains allows the orchestrator / service provider to have a common software platform to listen performance data and automate the network. Before TAPI v2.5, TAPI streaming was existing but with limited data streaming capabilities focusing alarm, fault, and configuration data to monitor the state of the network. The data exchange takes place over RESTCONF with JSON data encoding format, where the same strategy is also applied for performance data. However, the performance data in the network must be continuously monitored and anomalies in the data should be addressed promptly. This requires continuous streaming of performance metrics in an efficient manner as this adds huge load to network traffic. So, TAPI v2.5 addressed the difficulties in streaming the

performance data in section 6.1.1 to 6.1.11 in TAPI v2.5 reference implementation agreement. The key consideration includes volume of data, data source, efficient streaming functionality, measurement time, etc. In addition, a solution for streaming performance data using gNMI/GRPC is also recommended in section 6.2 in TAPI v2.5 reference implementation agreement. The solution recommends the use of GNMI connection protocol, PROTOBUF encoding format, and truncated log-storage strategy.

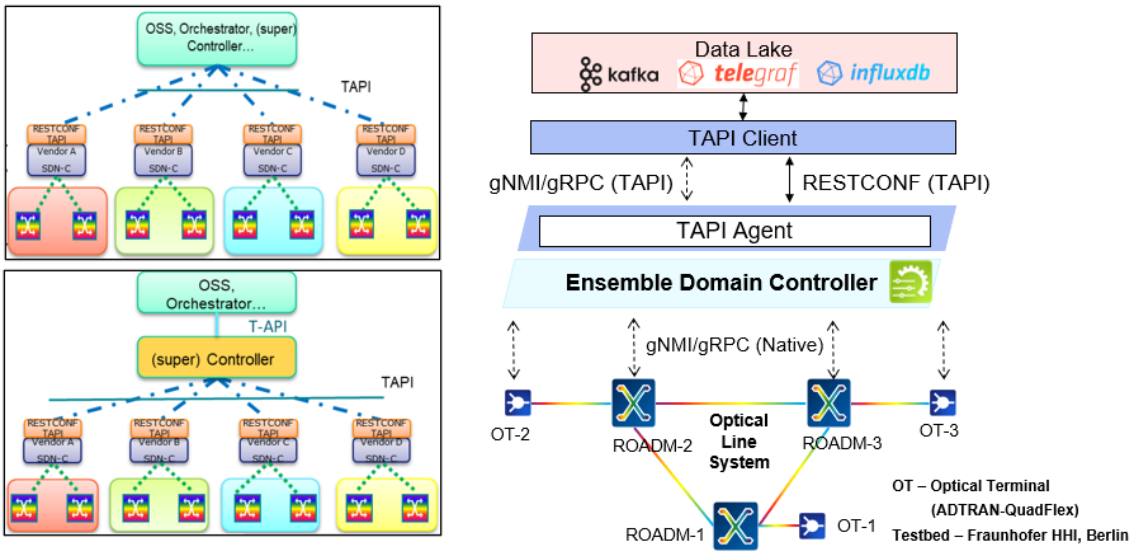


Figure 4.2-1: ONF-TAPI v2.5 – Reference Architecture (left) and Proposed Architecture (right).

API's and Interfaces for Integration:

NorthBound (Between Management Plane and Control Plane):

In TAPI v2.5, the YANG data model and the respective Protobuf file is defined. The YANG data model is defined in such a way, that a unified data structure to support streaming of data from multiple domains. This enables the operator to subscribe for the PM data without complexity and can able to monitor the E2E service spanning across multiple domains.

As seen in Figure 4.2-, referenced from section 1.2 in TAPI v2.5 reference implementation agreement, the controllers are equipped with a TAPI component such that performance data of each domain is retrieved using a common data model for the OSS. In our implementation as in figure 2, a similar TAPI proxy agent is attached which gets all the PM data from the network, and streams it to northbound using a unified TAPI model. The Transport Layer Security (TLS) v1.3 is used to establish communication between gNMI/gRPC server and client. In this Proof of Concept, the gNMI/gRPC based streaming is evaluated with REST best approach. So, the same data model is tested using REST by implementing the REST server and client. The throughput of both approaches are compared as seen in figure 3.

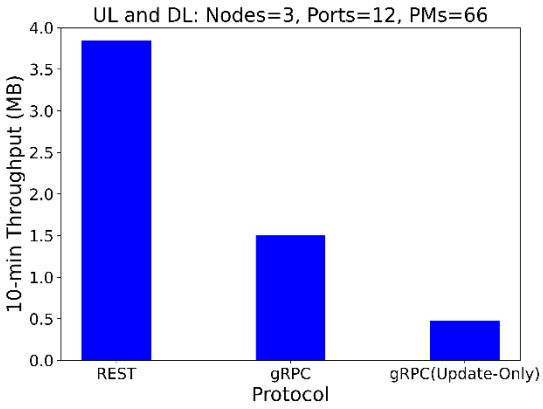


Figure 4.2-2: Telemetry Streaming - Northbound - RESTCONF vs gRPC.

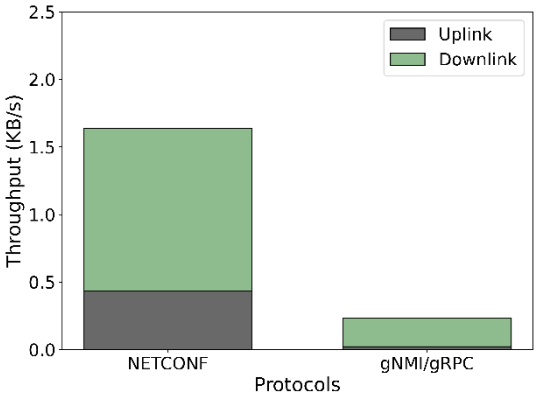


Figure 4.2-3: Telemetry Streaming - Southbound – NETCONF vs gRPC.

South Bound (Between Control Plane and Data Plane):

NETCONF is often used in the southbound API for telemetry data retrieval, utilizing text-based data serialization. However, this method is less efficient in terms of serialization latency, throughput, and compression ratio. In contrast, gRPC employs PROTOBUF, a binary-encoded protocol that offers better serialization and data compression.

Validation and Results:

NorthBound:

We have demonstrated our implementation on the networking testbed hosted by Fraunhofer HHI and performed a set of experiments to compare the performance of gNMI/gRPC with REST. For our experiments, the subscribed data in the northbound is pushed to the HHI Data Lake, which itself complies with telemetry system of ETSI ISG F5G. The gNMI/gRPC based streaming offers various advantages over REST based telemetry retrieval. The gNMI/gRPC offers a lightweight payload with 2.5 times smaller than REST in telemetry streaming as shown in Figure 4.2-. Also, the gRPC uses protocol buffer definitions and uses binary data transfer which is significantly faster than REST.

SouthBound:

In the testbed, gNMI/gRPC capabilities are configured in all OLS and OT components to support telemetry streaming encrypted over TLSv1.3. An agent application, as described in Figure 4.2-,

is installed alongside the OLS controller to retrieve data from devices using gNMI/gRPC and stream it northbound in a TAPI data model. Performance monitoring (PM) data from an OT is retrieved using both NETCONF and gRPC.

To ensure a fair comparison, NETCONF requests include an XML filter to retrieve only the specified PM data, while gRPC subscription requests use xPATHs for the same purpose. Over a 10-minute period with a 10-second sampling interval, NETCONF consumes about 0.43 KB/s and 1.2 KB/s of traffic during uplink and downlink, respectively. In contrast, gRPC consumes only 0.02 KB/s for uplink and 0.21 KB/s for downlink. This efficiency is due to gRPC's binary serialization, which reduces payload size and traffic utilization, making NETCONF 5.6 times larger.

4.3 RIC TELEMETRY SYSTEM

The RIC telemetry system is a key enabler of the O-RAN architecture. The Telemetry System interfaces with both the Near-Real-Time RIC (Near-RT RIC) and the Non-Real-Time RIC (Non-RT RIC) to enable data-driven decision-making.

Some of the key functions of the RIC Telemetry System include Data Collection, Data Transport, Data Processing and Telemetry Visualisation.

Telemetry sources include the O-RAN components (O-RUs, O-DUs, ands O-CUs) and RIC Applications (xApps and rAPPs). Non-standardised data sources can also be incorporated e.g., external/proprietary network management and monitoring tools.

Telemetry data can consist of:

1. **Performance Metrics**: Such as tthroughput, latency, packet loss, signal quality, and interference levels.
2. **Operational Data**: Such as resource utilization, spectrum usage, and power consumption.
3. **Fault and Alarm Data**: Such as information about errors, faults, and failures in the RAN.
4. **Event Logs**: Such as detailed logs of specific events like handovers, drops, or anomalies.

Efficient transport of telemetry data is critical for the RIC system to ensure scalability, low latency, and reliability. The O-RAN ecosystem uses modern telemetry transport protocols and architectures to handle this data.

Typical telemetry transport mechanisms include:

- **Kafka**: A distributed messaging system used for transporting telemetry data, especially in the Non-RT RIC. Kafka ensures scalable, reliable, and ordered data streams for processing and storage.
- **HTTP/REST APIs**: Used for Non-RT telemetry transport and integration with external systems. It offers flexibility for exchanging data but may not be ideal for real-time operations.
- **NETCONF/YANG**: A protocol for managing and monitoring RAN configurations. It is primarily used to retrieve configuration telemetry.
- **MQTT (Message Queuing Telemetry Transport)**: A lightweight messaging protocol more typically used in resource-constrained environments. This may be used for providing telemetry from low-power or IoT applications within the RAN.

- **gRPC (gRemote Procedure Call)**: Typically used in O-RAN for high-performance, low-latency communication. It supports streaming telemetry data between RAN components and the RIC. It provides efficient binary serialization (through the use of Protocol Buffers) for compact data transfer.
- **Proprietary Transport Protocols**: Non-standardised transport mechanisms can also be incorporated if required, but the O-RAN Alliance promotes open and interoperable solutions.

Telemetry in O-RAN based networks can be utilised to support:

1. **Real-Time Monitoring**: Telemetry data is collected in near real-time from various O-RAN components, such as the Radio Unit (O-RU), Distributed Unit (O-DU), and Central Unit (O-CU).
2. **Open Interfaces**: O-RAN specifies open interfaces, such as the O1, O2 and E2 interfaces (see Figure 3.2-), which facilitate telemetry data exchange between network components and management systems like the Service Management and Orchestration (SMO) framework.
3. **Performance Metrics**: Telemetry data includes critical metrics like:
    - Network Key Performance Indicators (KPIs)
    - Resource utilization (e.g., CPU, memory, and bandwidth usage)
    - Radio metrics (e.g., signal strength, throughput, latency)
    - Fault information and alarms
4. **AI/ML Integration**: Telemetry data in O-RAN is often used to feed AI/ML models within the Non-Real-Time RAN Intelligent Controller (Non-RT RIC) or Near-Real-Time RIC (Near-RT RIC). These models help with network optimization, predictive maintenance, and anomaly detection.
5. **Streaming Protocols**: O-RAN systems commonly use protocols like gRPC, NETCONF, or Kafka for telemetry data streaming, ensuring low latency and high reliability.
6. **Scalability**: Telemetry systems in O-RAN are designed to dynamically scale with network demands, enabling flexible and efficient monitoring in complex, multi-vendor environments.

Telemetry can provide real-time insights into the state of the network, allowing operators to make informed decisions regarding resource allocation, optimization, and overall quality of service. As network infrastructures become more complex and scalable, the need for comprehensive and efficient telemetry systems becomes critical to maintaining smooth operations and adaptability.

4.4 INTELLIGENT DATA AGGREGATION

Description

Figure 4.4- presents the reference network scenario, where an SDN architecture controls a number of optical nodes, specifically optical transponders (TP) and reconfigurable optical add-drop multiplexers (ROADM) in the data plane. Note that the SDN architecture might include a hierarchy of controllers, including optical line systems and parent SDN controllers. A centralized telemetry manager is in charge of receiving, processing, and storing telemetry data in a telemetry DB, which includes two repositories: *i*) the measurements DB is a time-series (TS) DB that stores measurements; and *ii*) the events DB is a free-text search (FT) engine. In addition, telemetry data can be exported to other external systems, e.g., through Kafka. Some data exchange between the SDN control and the telemetry manager is needed, e.g., the telemetry manager needs to access the topology DB describing the optical network topology, as well as the label-switched path (LSP) DB describing the optical connections (these DBs are not shown in the figure for the sake of simplicity).

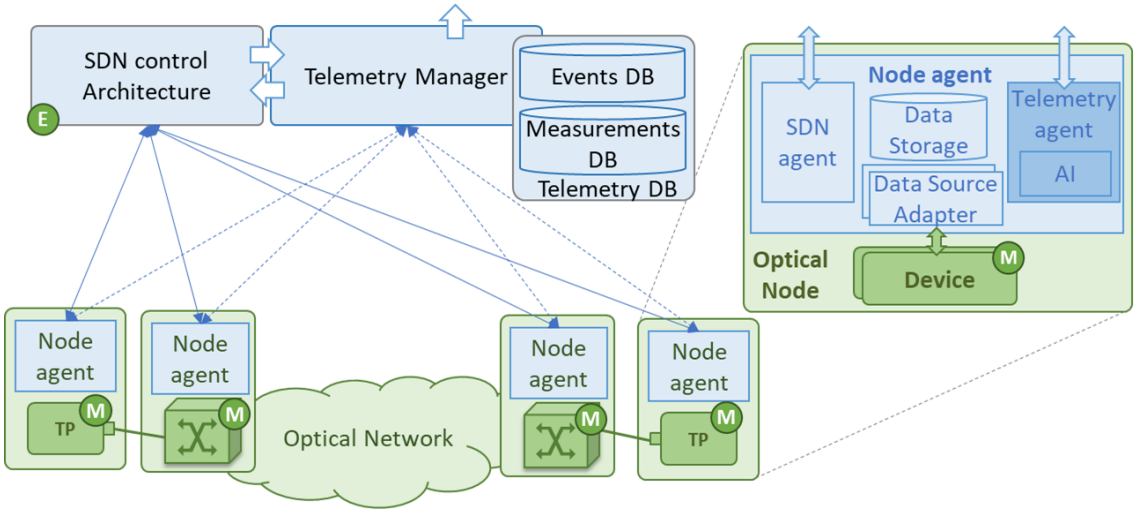


Figure 4.4-1: Overall network and proposed telemetry architecture.

Every node in the data plane is locally managed by a node agent (see some internal details in Figure 4.4-). The node agent translates the control messages received from the related SDN controller into operations in the local node. In addition, the node agent includes data source adaptors that collect measurements from observation points (labeled M) enabled in the optical nodes or in specific optical devices, like OSAs, as well as a telemetry agent that processes and exports telemetry data to the telemetry manager. In addition, events can be collected from applications and controllers (labeled E).

The internal architecture of telemetry agents inside node agents is presented in Figure 4.4-, which consists of five main components: i) a manager module configuring and supervising the operation of the rest of the modules; ii) a security manager in charge of security aspects, like key management; iii) a number of algorithms for data processing that include dimensionality reduction and data veracity checking; iv) a number of interfaces, e.g., gRPC, to communicate with other systems. Additionally, interfaces take care of the security of telemetry data, e.g., to ensure data privacy, authentication, and integrity; and v) a Redis DB that is used in publish-subscribe mode to communicate the different modules among them, i.e., no direct

communication is allowed. This facilitates the definition of specific workflows for telemetry data and provides an agile, reliable, and secure environment that simplifies communication, as well as integration of new modules.

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

The internal architecture of the telemetry manager is the same that the one for the agents. The difference between them is on the algorithms and the interfaces that they run. E.g., the telemetry manager includes interfaces to the telemetry DBs and to export data to external systems.

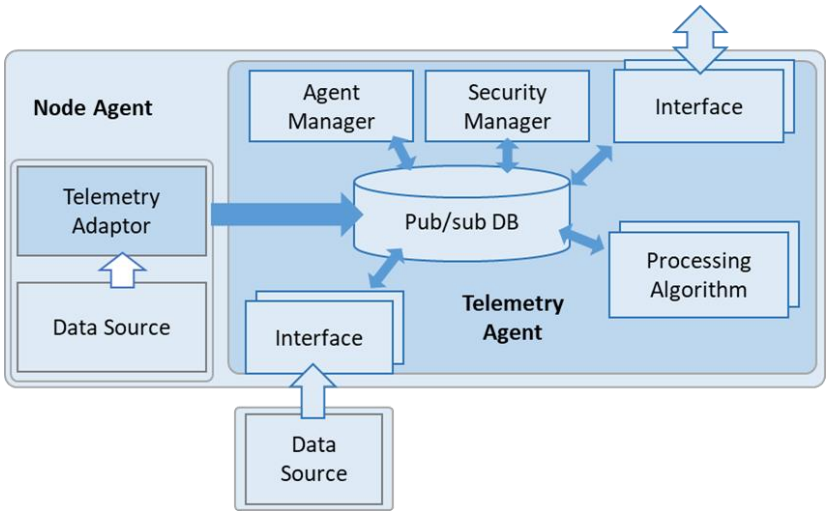


Figure 4.4-2: Telemetry agent architecture.

APIs and Interfaces for Integration

A data source example has been developed and it is available for the rest of the partners in the project.

Status, features under development

Intelligent data aggregation algorithms have been implemented in Python and deployed in the telemetry agent and manager.

The following strategies have been developed:

A) Supervised feature extraction: A simple but effective dimensionality reduction technique is supervised FeX. This technique is intended to generate the set of features Φ(M) that characterize a measurement sample M.
B) Data compression: intelligent telemetry data compression is performed at telemetry agents before data are sent to the telemetry manager. The compression technique is based on the use of Autoencoders (AE). An AE is a type of neural network with two components: the encoder, which maps input data into a lower-dimensional latent space,

and the decoder, which gets data from the latent space and reconstructs the original data back. The proposed compression method is compatible with any data serialization and compression engine built on common protocols, like gRPC, and distribution platforms like Kafka.

C) Data Summarization: High frequency of telemetry data collection entails a large volume of data being conveyed to the telemetry manager. However, this is not needed in general in normal conditions, where the collected values of the counters are stable. Data summarization consists on measuring variations in the computed features to decide whether a new sample M or a representation of it needs to be sent to the telemetry manager. In case of no significant variations are found, the telemetry agent can send averaged values of the features with a much lower frequency, thus reducing the volume of telemetry data being conveyed.

Validations and Intermediate Results

For evaluation, three data sources have been developed. Representation of the JSON objects are shown in Figure 4.4-. The spectrum data source emulates spectrum samples collection from an Optical Spectrum Analyzer (OSA). The OSA measures the whole C band (4.8 THz) and an algorithm processes the measurement and selects the spectrum for each channel separately. Therefore, each measurement S for the spectrum of a 75GHz channel consists of a list of 75 <f,p> pairs, i.e., 600 bytes assuming 32-bit scalars. The telemetry adaptor in the data source publishes samples S encoded as JSON objects of size 1,207 bytes (input a in Figure 4.4-).

The constellations data source emulates IQ constellation samples collection from an observation point in a Transponder. Specifically, two sizes of constellation samples X are considered, containing k=2,048 and k=10,000 symbols, respectively from a 16-QAM optical signal. The size of each raw sample X in a scalar representation is 2×k×4, i.e., 16,384 bytes and 80,000 bytes, assuming that every symbol is represented with two scalars (I and Q). The telemetry adaptor publishes raw samples X encoded as JSON objects (input b in Figure 4.4-) of size 75,783 and 370,007 bytes, respectively.

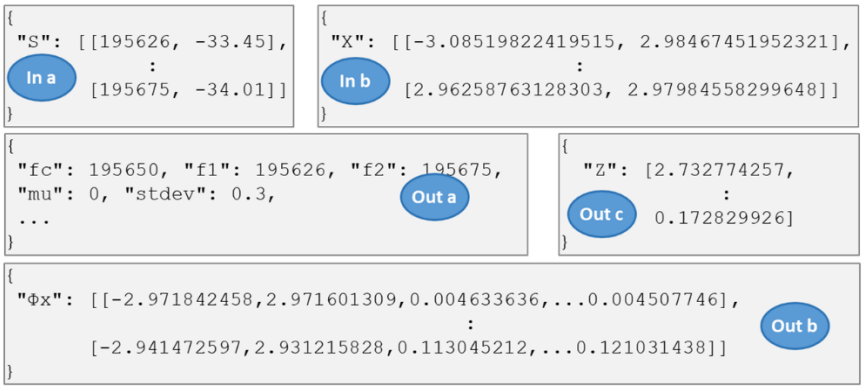


Figure 4.4-3: JSON representations of received and processed samples.

A) Feature extraction

In the case of supervised FeX from the optical spectrum of a lightpath, the algorithm in the telemetry agent generates features ΦS in a JSON object with 184 characters (output a in Figure 4.4-), which is then serialized before being conveyed through the gRPC interface. As for IQ constellations, the algorithm in the telemetry agent applies GMM fitting to every

constellation sample X received and generates features ΦX encoded as a JSON object with 1,000 characters (output b in Figure 4.4-).

B) Data compression:

Data compression using AEs has been evaluated using IQ constellation samples with 10,000 symbols.

We need to firstly determine the resolution p that allows an accurate representation of the original constellation. Specifically, we found that negligible error (~1%) for p > 35,000, so we selected p = 36,864, which entails splitting the IQ constellation in a 192x192 grid i.e., each of the 16 constellation points is mapped on a 48x48 grid.

With such value, we have obtained an overall reconstruction accuracy of 95% and compression ratio of 322.6 by compressing 10,000 symbols into 32 latent space features, which are output as JSON objects (output c in Figure 4.4-), resulting in 391 characters in total for the JSON object.

C) Data summarization

Data summarization on the extracted features has been evaluated for both IQ constellation using samples with 10,000 symbols and 75 GHz optical spectrum samples from a lightpath of 1,000 km operating under normal conditions. A remarkable summarization (<10% of total telemetry measurements) has been achieved for both IQ constellations and optical spectrum samples.

5 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.

At the time of writing, 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 5-1 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 5-1: 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.	Not started.
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.	Not started.
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	Preliminary results available in T2.3 (by simulations), experimental results expected in WP5
6.1	40% CAPEX reduction by collapsing computing, IP networking, and usage of high-speed coherent optical transmission in a single element (i.e., DPU) not designed for the Telecom market but for much wider computing markets and verticals (e.g., automotive).	First partial assessment considering ROADM-free networks, submitted to ComMag. Reference SEASON scenario presented at ECOC as Invited Talk.
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	Progressing well. 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.
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.	Progressing well. 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.	Assessed 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 πarallelism 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 under study in WP3 evaluating monitoring solution in the lab, development and regression testing within WP4, and testing and demo in WP5. 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 infrastrcture configurationa and service configuration. Inital results are evaluated are proposed for CNSM 2024.
7.3	Configuration of the MBoSDM node prototype agents for simple operations from the SDN control plane of O(100ms).	
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)	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 underdefinition 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)	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.
8.3	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)	Partially addressed 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)	Not started.

6 CONCLUSIONS

This deliverable, *D4.2 Second year design of control plane infrastructure*, extends the previous deliverable (D4.1) that presented the overall SEASON solution and control plane architecture. In this document, the different functional elements have been identified, the requirements regarding the interworking and interfaces and their initial validation.

We have reported on the different software components by grouping them into four main categories: i) SDN Agents; ii)single domain controllers; iii) multi-domain orchestrators and over-arching controllers; and iv) algorithms in support of network operation, planning, and control. Similarly, we have detailed the different telemetry systems used across the proof of concepts and validations.

The list of components has been provided to WP5 for integration in the two main demonstration activities, with the possibility of targeting additional specific demonstrations. As part of the initial WP5 work, components are mapped to demos with specific scenarios in mind. Consequently, software development will continue to address missing functionality and ensure that the identified workflows associated with the main demonstrations can be reproduced.

7 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*
MBoSDM	*Multi-Band over Space Division Multiplexing*
MEC	*Mobile Edge Computing*
MH	*Madhul*
ML	*Machine Learning*
NBI	*Northbound Interface*
NE	*Network Element*
Near-RT	*Near-Real Time*
NETCONF	*Network Configuration protocol*
NetDevOps	*Network Development Operations*
NFV	*Network Function Virtualization*
NGMN	*Next Generation Management Network*
NIC	*Network Interface Card*
Non-RT	*Non-Real Time*
ODU	*Outdoor Unit*
OLS	*Optical Line System*
OLT	*Optical Line Terminal*
ONAP	*Open Network Automation Platform*
ONF	*Optical Networking Foundation*
ONT	*Optical Network Termination*
ONU	*Optical Network Unit*
OPEX	*Operational Expenditure*
O-RAN	*Open RAN*
OSA	*Optical Spectrum Analyzer*
OSM	*Open-Source MANO*
OSNR	*Optical Signal-to-Noise Ratio*

OSS	*Operations Support Systems*
OTN	*Optical Transport Network*
P2MP	*Point-to-Multipoint*
P2P	*Point-to-Point*
PON	*Passive Optical Network*
QAM	*Quadrature Amplitude Modulation*
QoS	*Quality of Service*
QoT	*Quality of Transmission*
RAN	*Radio Access Network*
RIC	*Radio Intelligent Controller*
RL	*Reinforcement Learning*
ROADM	*Reconfigurable Optical Add/Drop Multiplexer*
Rx	*Receiver*
RSCA	*Routing and Spectrum and Core Assignment*
SBI	*Southbound Interface*
SBA	*Service Based Architecture*
SDM	*Space Division Multiplexing*
SDN	*Software Defined Networking*
SLA	*Service Level Agreement*
TAPI	*Transport API*
TFS	*TeraFlowSDN*
VNF	*Virtual Network Function*
WDM	*Wavelength Division Multiplexing*
WSS	*Wavelength Selective Switch*

8 REFERENCES

[Cos21] L. R. Costa, et. al., “Energy Efficiency in Sliceable-Transponder Enabled Elastic Optical Networks,” IEEE Transactions on Green Communications and Networking, vol 5, no. 2, 2021.

[Cug23] F. Cugini, C. Natalino, D. Scano, F. Paolucci, and P. Monti, “P4-based telemetry processing for fast soft failure recovery in packet-optical networks”, in 2023 Optical Fiber Communications (OFC) Conference, 2023

[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

[Dav24] R. P. Davey et. al., “ZR 400 Gbit/s and 800 Gbit/s use cases, trials, deployments, and future prospects”, Journal of Optical Communications and Networks, vol. 16, no. 1, Jan 2024

[Mar24] R. Martínez, et. al., “Enhancing Network Performance and Reducing Power Consumption in Elastic Optical Networks with Deep Reinforcement Learning”, in Proc. of ONDM 2024.

[Mos24] D. de la Osa Mostazo, et. al., “Lessons learned from IP routers power measurements and characterization”, in proc. of NOF 2024

[ORAN24] O-RAN architecture. (2024) (online) https://docs.o-ran-sc.org/en/latest/architecture/architecture.html here.

[Tapi23] ONF TR-547 TAPI Reference Implementation Agreement, v3.1, (2023).