



# 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 D5.2

### Final report of demo 1

**Editor** Muhammad Rehan Raza (HHI)

**Contributors** HHI, CNIT, CTTC, ADVA, TID, UPC, WIN

**Version** 2.0

**Date** December 31, 2025

**Distribution** PU

## DISCLAIMER

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

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

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

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

## REVISION HISTORY

<b>Revision</b>	<b>Date</b>	<b>Responsible</b>	<b>Comment</b>
1.0	October 29, 2025	HHI	Initial ToC version.
1.1	November 7, 2025	HHI	First set of contributions.
1.2	November 28, 2025	HHI	Final contributions.
1.3	December 1, 2025	HHI	Start quality review.
1.4	December 12, 2025	NOK-P	Quality check (C. Pinho).
2.0	December 17, 2025	HHI	Final version for submission

## LIST OF AUTHORS

<b>Partner</b>	<b>Name Surname</b>
Adtran (ADVA)	Achim Autenrieth, Vignesh Karunakaran, Nikhil DSilva
CNIT	Bashar Ali, Alessio Giorgetti, Andrea Sgambelluri, Filippo Cugini
CTTC	Ramon Casellas, Ricardo Martinez, Laia Nadal, Luca Vettori, J. Vilchez, Carlos Hernández-Chulde.
HHI	Muhammad Rehan Raza, Abdelrahmane Moawad
TID	Pablo Armingol, Oscar Gonzalez
UPC	Luis Velasco, Marc Ruiz, Jaume Comellas, Salvatore Spadaro, Fernando Agraz, Joan Gené
WIN	Vasileios Tsekenis, Sokratis Barmounakis, Panagiotis Demestichas

## EXECUTIVE SUMMARY

This deliverable, along with D5.3, presents the experimental validation of the SEASON solution by integrating data plane components from WP3 with control plane components from WP4. While other demonstrations are presented in the deliverable D5.3, this deliverable focuses on two demonstrations, hosted at HHI and CTTC, which include different Multi-Band over Spatial Division Multiplexing (MBoSDM) prototypes developed in this project.

The HHI demo highlights self-healing in an IP-over-WDM network with dual protection for video streaming traffic, combining advanced control-plane elements with a testbed built from commercial hardware and an in-house MBoSDM prototype. The CTTC demo features a different MBoSDM prototype that incorporates active components like wavelength selective switches. For both demos, this deliverable describes their respective setups, workflows, and experimental validation through screenshots of demo execution and some measurement results.

The purpose of this document is to emphasize the major accomplishments achieved in the project, presented through fully integrated demonstrations. It highlights the strong collaboration among the partners and different work packages of this project.

# TABLE OF CONTENTS

1	Introduction.....	5
2	Demo 1: Self-Healing in IPoWDM Network with Dual Protection Using MBoSDM and NetDevOps Solutions.....	6
2.1	Demo Architecture and Components.....	6
2.1.1	Control Plane .....	6
2.1.2	Data Plane.....	10
2.2	Workflow.....	13
2.3	Experimental Validation .....	14
3	Demonstration of Dynamic Multiband and SDM Service Provisioning.....	24
3.1	Demo Architecture and Components.....	24
3.1.1	Control Plane .....	24
3.1.2	Data Plane.....	25
3.2	Experimental Validation .....	26
3.2.1	Initialization phase .....	26
3.3	Workflow.....	28
4	Conclusion .....	37
	Glossary .....	38
	References.....	40

# 1 INTRODUCTION

---

The deliverables D5.2 and D5.3 are complementary documents that present the experimental demonstration and validation of the SEASON solution, while describing the integration of data plane components developed in WP3 and control plane components created in WP4. Although other demonstrations are covered in D5.3, this deliverable presents two integrated demonstrations performed at HHI and CTTC. Both demonstrations are covered in this deliverable as both depict the Multi-Band over Spatial Division Multiplexing (MBoSDM) technology, i.e., one of the core concepts of SEASON solution.

The second chapter describes one of the final demos of SEASON project, i.e., self-healing in an IP-over-WDM network with dual protection for video streaming traffic, that was carried out at HHI's premises. The novel control plane components developed in the project that were integrated together in the demo are described. This includes service/network orchestrators, IP/optical controllers, Open Line System (OLS) controller, telemetry system as well as Network and Development Operations (NetDevOps) which is one of the key highlights of this demo. For the data plane, the testbed at HHI is presented which includes commercial hardware and MBoSDM prototype developed at HHI. Moreover, the use case and demo workflow is described in detail. The experimental validation is presented by including the screenshots of demo execution, and some results on timing measurements performed in the demo.

The third chapter presents second integrated demo of the SEASON project that includes the MBoSDM node prototype developed at CTTC's premises. The main difference between this prototype and the one used in first integrated demo (from HHI) is the inclusion of active components, such as wavelength selective switches (WSSs). The differences between the two prototypes are described in WP3 deliverables in more detail. This chapter also follows a similar structure as the previous chapter, i.e., it describes the data and control plane components integrated in the demo which is followed by the description of workflow and experimental validation using screenshots taken from the demo execution.

## 2 DEMO 1: SELF-HEALING IN IPOWDM NETWORK WITH DUAL PROTECTION USING MBoSDM AND NETDEVOPS SOLUTIONS

This chapter describes the final demo of SEASON project that was performed at HHI.

### 2.1 DEMO ARCHITECTURE AND COMPONENTS

Figure 2.1-1 depicts the architecture of HHI demo with both data and control planes. All the demo components are described in detail in the following sub-sections.

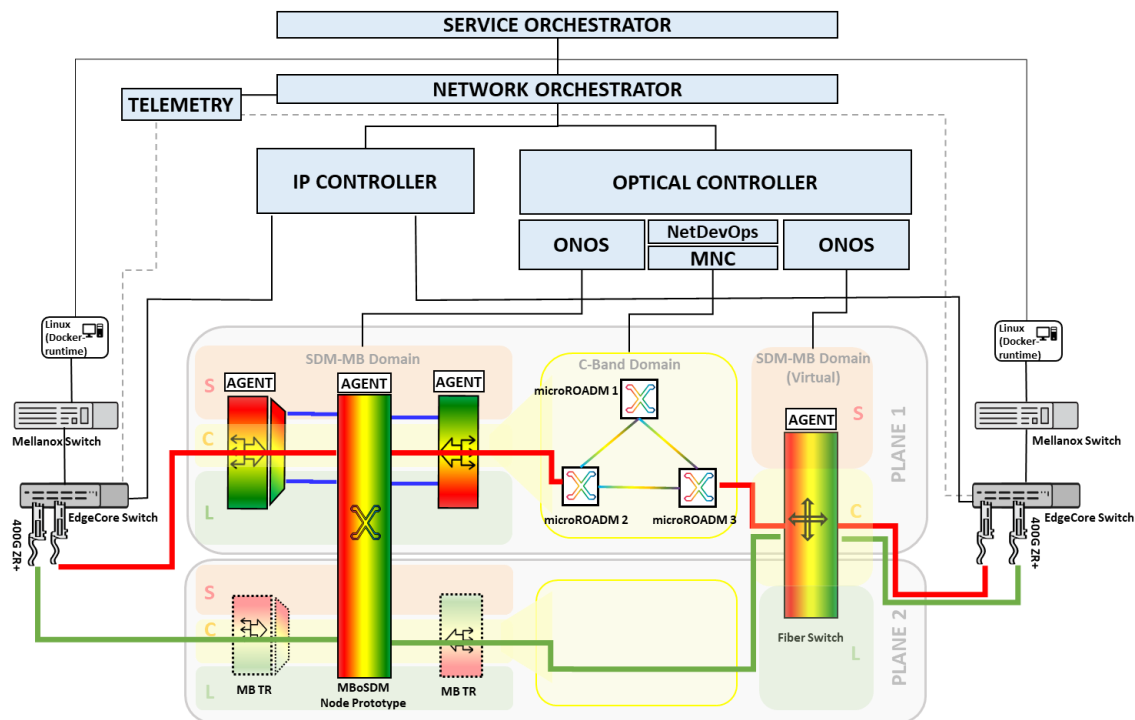


Figure 2.1-1: HHI Demo Architecture.

#### 2.1.1 Control Plane

The control plane comprises the following components:

##### Service Orchestrator

The service orchestrator provides the control-plane logic that instantiates, scales, and supervises the media-processing workloads for the demo. Concretely, the service orchestrator is implemented as a Python module coupled with a Flask + Flask-SocketIO web application, exposing both REST APIs and Socket.IO events for live control and telemetry. The transmission

control protocol (TCP) streamer container sends framed video frames over TCP to a receiver container, which aggregates bandwidth and forwards sampled frames and statistics to the dashboard. It runs a-top Kubernetes (K8s) and deploys the UE-receiver as a StatefulSet with host networking, assigning per-replica host ports (e.g., 9990 + ordinal). This preserves stable network identities and enables horizontal scale-out across worker nodes. The orchestrator ensures dependent resources (services, a headless service for StatefulSet DNS, and the workload) are present and consistent; gates streamer start-up until at least one receiver is “Ready” to avoid connection errors; and accelerates scale-down by setting “podManagementPolicy: Parallel” with a short termination grace period. Operator interaction happens through the WINGS Service Management and Orchestration Dashboard, which exposes streamer controls, inter-node network tests (host/K8s RTT/BW), a live topology view, and logs in a single pane. The dashboard connects to the backend via Socket.IO and REST. It offers a “Number of Streamers” control plus an “Emit every Nth frame” parameter, a dynamic grid of “Video Streams” tiles fed by video\_frame events, tables for pod status and bandwidth per stream, and dedicated charts for “Inter-node Net Test” (ping + iperf3 host vs. in-Kubernetes paths), “Receiver Autoscaling” (HPA status), and “Receiver Load” (aggregate Mbps from receiver\_stats). A “Topology” panel uses a vis-network graph driven either by TeraFlowSDN (TFS) topology (/api/tfs/topology) or by a local topology file.

The autoscaling is handled via K8s HPA. The orchestrator applies an HPA targeting CPU utilization while respecting platform constraints. The HPA resource is attached to the receiver workload using the autoscaling/v2 API, with explicit up/down scale behaviors, min/max replicas, and current CPU metrics exposed through the /api/k8s/hpa endpoint for visualization in the “Receiver Autoscaling” chart. When the desired streamer count is driven to zero, the orchestrator explicitly deletes the HPA and scales the StatefulSet to zero to avoid the controller racing it back to minReplicas. This “scale to zero” path is triggered by set\_streamers(count=0), which (i) ensures the receiver stack exists, (ii) removes the HPA, (iii) scales the receiver replicas to 0, and (iv) cleans up any surplus UE-streamer-\* pods, with all actions logged and streamed to the dashboard. Operator control and observability are exposed through the dashboard where the user can set the number of streamers, observe live video tiles, and track receiver load and bandwidth charts fed by periodic statistics from the streamers. Bandwidth and load are computed in the receiver process, which tracks per-connection bytes and emits bw\_update events (per stream) and receiver\_stats events (aggregate Mbps and connection count per receiver instance) roughly once per second. These events feed both the per-stream “Bandwidth (Mbps)” table and the aggregate “Receiver Load” chart in the UI.

Finally, the orchestrator has been structured to accommodate inputs from the TFS (e.g., topology/telemetry) which has the role of the network orchestrator. While the demo primarily exercises Kubernetes-native scaling and readiness policies, the design leaves room to consume external telemetry and drive policies such as adapting receiver replicas based on upstream network conditions so the service orchestrator can act as the convergence point between higher-level intent and concrete workload lifecycle actions in the cluster.

## Network Orchestrator

The network orchestrator provides the control plane logic that translates and executes the intents received from the services. It consists of a combination of two components: IETF Network Slice Controller (NSC) [Con25], which translates the intents, performs the path computation necessary to determine the equipment and parameters involved in the services to be created, and generates the necessary services for each domain; and TFS, which communicates with the various domain controllers, receives the services generated by the NSC, and sends the corresponding services to each controller in the required format.

The NSC has a REST API that acts as a northbound interface (NBI) for the service orchestrator. This interface exposes the configured intents and allows intents to be created or deleted. From TFS, there is a southbound interface (SBI) that implements different interfaces. For the IP controller, it is based on the IETF format, while for the optical controller, it is based on Transport API (TAPI).

## IP Controller

The IP controller provides the control plane logic that configures the IP layer from the network orchestrator services. It's based on TFS and interacts with IP devices. The IP controller provides a YANG-based layer for exposing and manipulating IP/MPLS services in a vendor-agnostic manner. This includes services such as Layer-2 VPNs [Bar22], Layer-3 VPNs [Bar23], peering services, access control lists, and routing policies. Its main role is to translate the service from the network orchestrator and generate the needed services to configure the IP devices.

The interface towards the network orchestrator is RESTCONF, following the IETF service models for L2 and L3. The network orchestrator (and, by extension, other components such as the graphical user interface) can query the state of the IP network, or request inventory data from the equipment, among other features.

## Optical Controller

The FlexOpt optical controller/orchestrator interacts with different domain controllers to provide an end-to-end optical service via an abstracted and standards based (i.e., TAPI) interface. Its main role is to abstract the underlying topological and control details and provide a unified interaction to the higher layers and aggregating workflows enabling, as needed, multi-domain connections. For this, it implements domain-specific SBIs to the different domains and creates a single topological view of the optical network (including the discrete multiband Tx/Rx) elements.

The interface towards the ONOS domains is ONOS-native, defined by the ONOS REST API, extended as needed. The interface towards the ADTRAN OLS domain is based on TAPI 2.1. The network orchestrator (and, by extension, other components such as the graphical user interface)

can query the state of the optical network, or request data connectivity services between the Service Interface Points (SIPs) at different supported layers.

### Telemetry

The telemetry collector continuously receives performance data from EdgeCore switches in the data plane. These devices monitor several optical transmission parameters, including BER, OSNR, and both Tx and Rx power levels, which are essential for identifying early signs of fiber degradation or equipment malfunction. Once collected, all telemetry is stored locally in a time-series InfluxDB instance, allowing not only to perform detailed analysis but also to maintain a historical record of the network's state. For real-time visualization, the stored measurements can be displayed through a Grafana dashboard. The collector also processes and periodically forwards this information to the network orchestrator through a Redis database, ensuring that the network orchestrator always has access to the last updated performance metrics. When any performance issue is detected, the network orchestrator can automatically trigger the necessary optimization actions to maintain the expected Quality-of-Service (QoS) and ensure stable network operation.

### Adtran OLS Controller

The optical controller proposed in SEASON for Optical Line System (OLS) control is based on the Adtran Mosaic Network Controller (MNC) software solution. It offers a northbound ONF TAPI v2.1 interface. The OLS components—including WSS, splitters, boosters, and pre-amplifiers—are controlled via SBIs using native YANG models and employ NETCONF/SNMP for network operations. The controller manages the provisioning of optical services, which includes port configuration, channel frequency assignment, channel setup, and other OLS-related operations. The OLS controller supports TAPI v2.1.4 as its northbound interface. Based on ONF TAPI 2.1.4 models, the Adtran TAPI implementation supports a flat abstraction model that collapses all layers into a single multilayer topology. The TAPI northbound interface of the controller is used to provision the service. The novel aspect of this work is the integration of NetDevOps with the OLS controller to enable version-controlled network state management.

### NetDevOps

NetDevOps is integrated with the optical controller using TAPI-RESTCONF, enabling automated create, delete, and update operations for services in the OLS network. NetDevOps applies DevOps principles to networking, ensuring changes are small, frequent, automated, and reliable. DevOps is a software development strategy and culture that bridges the gap between Development (Dev) and Operations (Ops) teams to build, test, and release software faster and more reliably.

A Python web server acts as both the NetDevOps entry point and a TAPI proxy. When the optical controller sends a service-creation request: (i) The proxy receives the request via REST API, (ii) commits the corresponding change to a Git repository, and (iii) triggers the NetDevOps pipeline, which executes in three stages:

- **State Gathering:** Queries each relevant network device for its current configuration and operational state, and retrieves service information from the TAPI interface via the proxy.
- **Execution and Tracking:** Applies the committed configuration to create or update service definitions across the infrastructure. During execution, it logs both device state and service state into a MongoDB database, preserving a full snapshot of the network at the end of the run.
- **History and Rollback Readiness:** Captured snapshots become part of the pipeline history. If rollback or transition to a different network state is required, the system references these historical snapshots—linked to specific Git commits or tags—to restore or reapply a known good state.

## ONOS

In the SEASON control-plane framework, ONOS serves as the controller for the optical domain and acts as the interface between the upper control layer and the underlying optical devices. ONOS receives the service-related requests generated by the orchestrator and translates them into NETCONF configurations for the optical elements involved in a given connection. Once a request is received, ONOS identifies the transponders and switching matrices that must be configured and sends the necessary updates to their NETCONF agents, ensuring that service setup and release operations are carried out in a consistent and coordinated way. Before any service can be established, ONOS runs a discovery phase where it connects to each device, retrieves its capabilities, and reconstructs the domain topology. Through this process, the controller learns the available nodes and links, the switching functions supported by each optical element, and the current status of the spectrum—information that is continuously updated and made available to the higher control layer for path computation and wavelength assignment.

To support the multi-band optical infrastructure used in SEASON, the standard ONOS distribution was extended with additional functions. These include **support for band switching** (allowing ONOS to configure optical components operating across the S and L bands), **fiber switching** (enabling the controller to manage port-to-port connections within the optical matrices), and **improved discovery mechanisms** that correctly identify and represent multi-band devices. With these extensions, ONOS provides a unified and technology-agnostic view of the optical domain, hiding device-specific details while enabling the orchestrator to operate using abstracted resources.

### 2.1.2 Data Plane

The data plane comprises the following components:

#### C-Band Domain

In the demo, three ROADMs—ROADM1, ROADM2, and ROADM3—are connected in a ring topology, each 80 km apart (see Figure 2.1-1). The C-band signal output from the MBoSDM node

is connected to ROADM2 and routed toward the virtual SDM-MB domain through ROADM3. A C-band signal with 100 GHz bandwidth is transmitted through the OLS network by provisioning a service across the nodes.

### SDM-MB Domain

The signal from C-band domain is routed through a node prototype, that was developed in WP3, which employs a fixed, static optical band-selective filtering architecture combined with a fully configurable optical cross-connect (optical matrix switch). This enables the node to operate as a hierarchical switching element, offering fast route-and-select functionality at the band level and at the fiber level as well. The design integrates eight multiband multiplexers together with a 12×12 optical matrix switch, providing flexible and dynamic band-routing capabilities. To support comprehensive system visibility, monitoring ports are included at every input and output, enabling the connection of measurement equipment to facilitate both prototype validation and continuous operational monitoring. The fully assembled prototype is depicted in Figure 2.1-2. More details can be found in the deliverables D3.2 and D3.3. Furthermore, the MB Tx/Rx are also routed through the MBoSDM node, to showcase the band switching capability of the prototype.

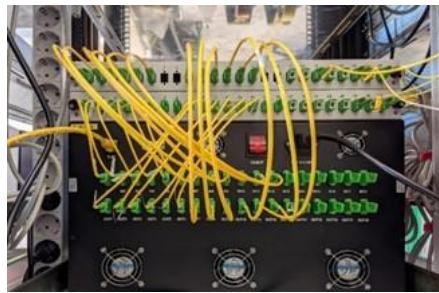


Figure 2.1-2: MBoSDM node prototype after assembly.

### SDM-Virtual Domain

A virtualized SDM domain (on the right side of Figure 2.1-1) also exists for fiber switching. This domain is controlled by a separate ONOS instance, and is used to switch two C-band signals from the C-band domain to the Edgcore switch.

### Edgcore Switches with Pluggable Transceivers

Two IPoWDM boxes have been added to the Edgcore switches to perform an inter-pop communication. These devices enable packet-optical traffic management, thanks to the usage of Coherent Pluggable Transceiver ZR+, with 400G/s of supported bit-rate, transmitting in the C-band (i.e, D-WDM) with coherent detection and Tx power in the range [+2, -8] dBm. SonicOS is used as Operative System, enabling the support of Common Management Interface Specification (CMIS) for the control/management of the device and the pluggable transceivers. In order to enable the SEASON control-plane in those devices, a NETCONF agent and an ad-hoc REST-based device driver have been designed and realized.

Considering the NETCONF agent, the OpenConfig model [OpenConfig] has been adopted to abstract and reproduce the IPoWDM elements, including in the components: (i) ports, (ii) transceivers and (iii) optical-channels. The agent has been equipped with a REST client to enable the communication with the device driver and configure the node and the pluggable transceivers.

Considering the device driver, a python framework has been implemented, based on Flask package to implement the REST API for both configuration and monitoring purposes. Figure 2.1-3 shows the view of the REST APIs from the swagger access.

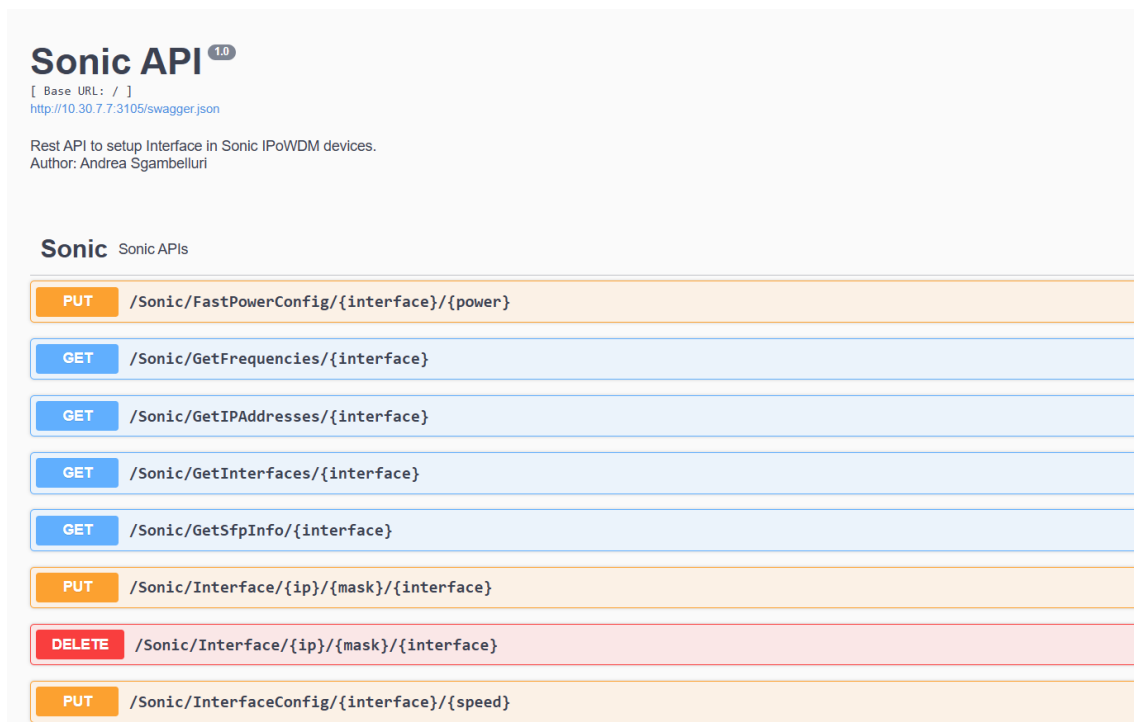


Figure 2.1-3: REST-based IPoWDM driver view.

## Aggregation Switches

The Mellanox switches are used as aggregation switches in this demo to aggregate the traffic in the network.

## Linux Boxes

The Linux boxes provide the general-purpose compute endpoints used to terminate, transform, and validate media streams in the demo data plane. Deployed as Ubuntu LTS VMs in the HHI lab, they are dual-homed toward the demo's management and service networks and host lightweight services required for end-to-end validation. A key role is the video receiver node, which runs NGINX with the RTMP module to ingest an RTMP feed (e.g., from a 5G UE or test encoder) and publish it as HLS for browser playback. For robustness during integration, HLS is exposed on HTTP with sensible defaults (segmenting and playlist settings), and, where direct

reachability is constrained, an alternate exposure via port 8080 or SSH port-forwarding is used to guarantee operator access from the dashboard host. The boxes also run FFmpeg for synthetic test streams and quick on-site troubleshooting.

Operationally, the Linux boxes complement the K8s workloads by providing stable, host-networked termination points and simple, observable file-system artifacts (HLS playlists and segments). They are configured with minimal firewall openings (HTTP and RTMP), file permissions for the HLS directory, and log collection (NGINX access/error logs) to speed fault isolation. The orchestrator's dashboard uses these hosts as data-plane anchors: it verifies playlist availability, and plays the HLS stream in-page (via HLS.js). This separation (K8s for elastic media processing and Linux hosts for protocol termination and last-mile delivery) keeps the overall system simple to operate while remaining flexible during the demo. It also decouples lifecycle concerns: K8s resources (StatefulSets, streamers, iperf3 pods) can be restarted or rescaled freely by the service orchestrator, while the Linux boxes remain long-lived, well-known termination points that network operators can manage.

## 2.2 WORKFLOW

The demo workflow is depicted in Figure 2.2-1. The workflow begins when an admin sends an end-to-end service creation request to the service orchestrator. The service orchestrator asks for instantiation of Docker containers for video streaming in both Linux boxes and requests the network orchestrator to provide connectivity between them (Steps 1-2). The network orchestrator requests IP and optical controllers to provision the red path in Figure 2.1-1 with direct connection between ROADM2 and ROADM3 in the C-band domain (Steps 3-4). The IP controller configures the Edgecore switches, whereas the optical controller configures three domains by interacting with both ONOS instances and NetDevOps/MNC (Steps 5-9). A failure (referred to as *Failure-1* in Figure 2.2-1) is introduced in the C-band domain on the direct path between ROADM2 and ROADM3 which disrupts the traffic on the red path. The telemetry system collects metrics (such as BER) from the Edgecore switches. When the BER exceeds a predefined threshold, the telemetry system sends a failure notification to the network orchestrator (Steps 10-11). The network orchestrator asks the optical controller to provide alternate path in C-band domain between ROADM2 and ROADM3 via ROADM1 using NetDevOps/MNC, which results in traffic flowing again on the red path (Steps 12-15). Another failure (referred to as *Failure-2* in Figure 2.2-1) is introduced in the C-band domain on the alternate path between ROADM2 and ROADM3 (via ROADM1) which disrupts the traffic again on the red path. The telemetry system again detects the failure and informs the network orchestrator about it (Steps 16-17). Considering that there is no path available in the C-band domain after both of these failures, the network orchestrator (via IP and optical controllers) provisions the green protection path that bypasses the C-band domain, and switches the traffic to the green path (Steps 18-22). During the recovery phase (not shown in Figure 2.2-1 due to space constraints), *Failure-2* in C-band domain gets fixed, and the network orchestrator switches the traffic back to the red path using alternate path between ROADM2 and ROADM3 via

ROADM1 following the same procedure (Steps 3-9). Finally, *Failure-1* in C-band domain gets fixed, and the network orchestrator reverts the traffic on the red path to the direct connection between ROADM2 and ROADM3 (Steps 12-15).

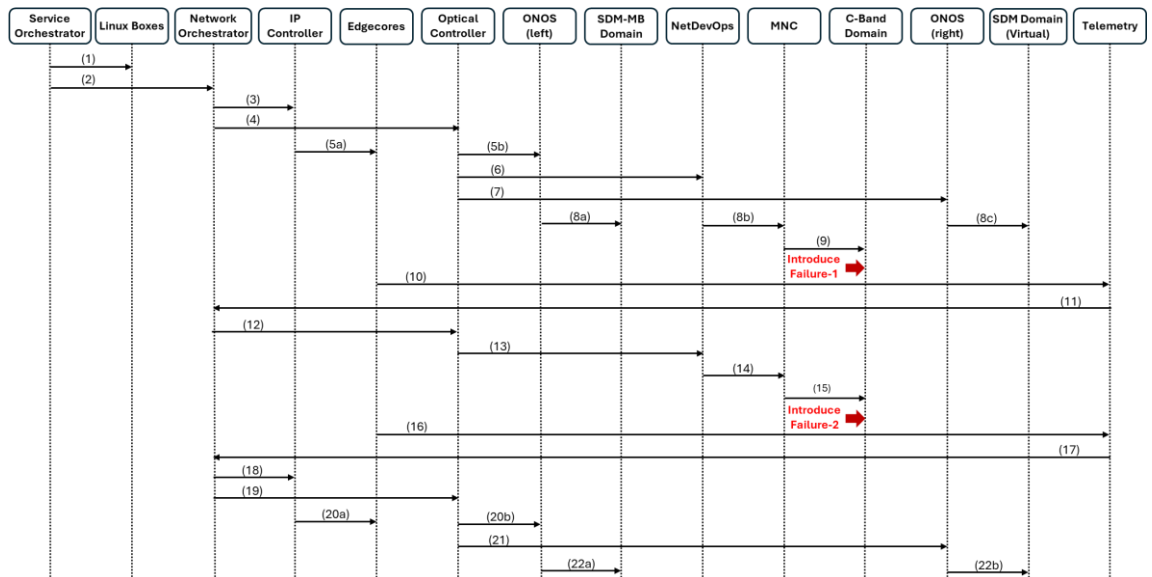


Figure 2.2-1: Demo workflow. Each request is associated with a response; but the responses are not depicted here due to space constraints.

### 2.3 EXPERIMENTAL VALIDATION

This sub-section shows some screenshots of the demo execution for experimental validation of our proposed setup. The detailed video of the demo will also be made available in the project.

Figure 2.3-1 depicts a snapshot of service orchestrator dashboard when two videos are being streamed over the network after establishing the connectivity. Figure 2.3-2 shows a streaming session where the receiver is actively ingesting traffic (aggregate Mbps rises), with the log pane recording HPA apply and delete events, and the pods table confirming that the UE-receiver is running.

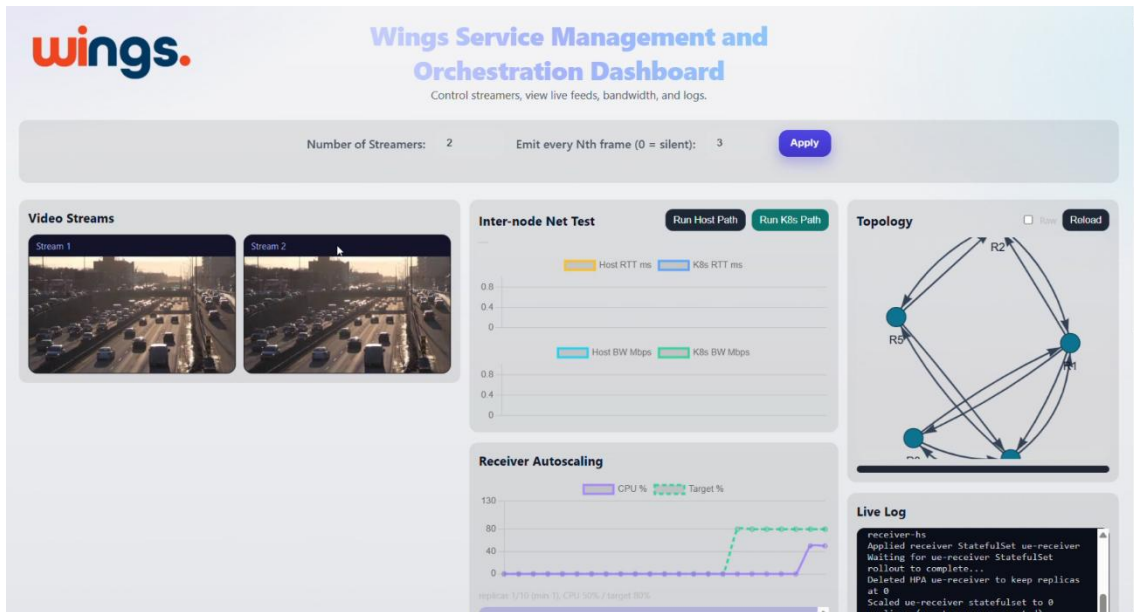


Figure 2.3-1: Service Orchestration Dashboard: Operator view exposing streamer controls, inter node network tests (host/K8s RTT/BW), network topology, and live logs.

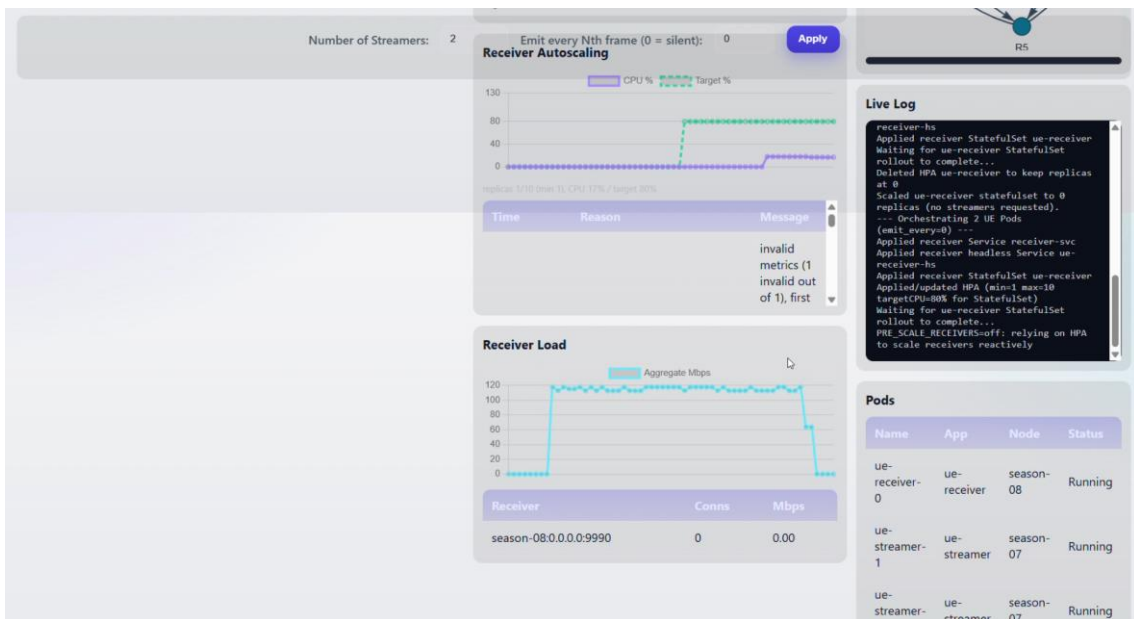


Figure 2.3-2: Receiver autoscaling and load during streaming: Autoscaling traces (CPU vs. target), orchestration logs, pods status, and 'Receiver Load' aggregate Mbps confirming active ingestion.

Figure 2.3-3 shows the logs from NSC component of the network orchestrator. The NSC is used to monitor the status of intents, showing which path computation has been chosen, the different services generated by each controller, and the total time taken to create the services. Figure 2.3-4 depicts the Services tab of TFS WebUI where each service generated can be tracked, with a list of the configuration parameters sent, the type of service, and the controller to which the service is sent.

Dissemination Level	PU
---------------------	----

```

-----NEW REQUEST-----
INFO - IETF intent received
INFO - Generating intent between 10.10.1.1 and 10.10.2.1
INFO - Rules generated by planner:
{
  "network-slice-uid": "8cc025-short-path-18881943-3995-442a-8a9d-4a5ca3a44hd", "viability": true, "actions": [{"type": "PROVISION_MEDIA_CHANNEL_OLS_PATH", "layer": "OPTICAL", "content": [{"src-sip-uid": "975f0b9a-389f-51d6-989f-18c7c3b0829", "dest-sip-uid": "f71a545e-7555-53ce-861e-113db1853a2c", "direction": "UNIDIRECTIONAL", "layer-protocol-name": "PHOTONIC_MEDIA", "layer-protocol-qualifier": "tapi-photonic-media:PHOTONIC_LAYER_QUALIFIER_OTS", "route-objective-function": "UNSPECIFIED", "service": "94033e13-d54d-4e6e-858e-da809cbfb51e"}], "tenant-uid": "req-left-fiber1", "url": "http://172.24.36.54:4900/restconf/data/tapi-common:context/tapi-connectivity:connectivity-context"}, {"type": "PROVISION_MEDIA_CHANNEL_OLS_PATH", "layer": "OPTICAL", "content": [{"src-sip-uid": "e1963c84-fbdd-529b-9836-b1909845c795", "dest-sip-uid": "8ca602b1-73c5-586a-917d-793a9191c1f2", "direction": "UNIDIRECTIONAL", "layer-protocol-name": "PHOTONIC_MEDIA", "layer-protocol-qualifier": "tapi-photonic-media:PHOTONIC_LAYER_QUALIFIER_OTS", "route-objective-function": "UNSPECIFIED", "service": "d8c197a-f5a8-4692-b629-a82728ca07"}, {"type": "PROVISION_MEDIA_CHANNEL_OLS_PATH", "layer": "OPTICAL", "content": [{"src-sip-uid": "2cbb691-c0d9-5683-b407-52c3d1208358", "dest-sip-uid": "9ef7d6f7-89bc-5911-beab-9d6e617fcf86", "direction": "UNIDIRECTIONAL", "layer-protocol-name": "PHOTONIC_MEDIA", "layer-protocol-qualifier": "tapi-photonic-media:PHOTONIC_LAYER_QUALIFIER_OTS", "route-objective-function": "UNSPECIFIED", "service": "dfdb31e7-8b68-4f7d-99b9-606e73aa9131"}], "tenant-uid": "req-right-fiber1", "url": "http://172.24.36.54:4900/restconf/data/tapi-common:context/tapi-connectivity:connectivity-context"}, {"type": "PROVISION_MEDIA_CHANNEL_OLS_PATH", "layer": "OPTICAL", "content": [{"src-sip-uid": "798c54e4-40aa599d-af8b-803d0c7c941", "dest-sip-uid": "f02d2d9-8ab8-54c8-8d49-429739b5f21d", "direction": "UNIDIRECTIONAL", "layer-protocol-name": "PHOTONIC_MEDIA", "layer-protocol-qualifier": "tapi-photonic-media:PHOTONIC_LAYER_QUALIFIER_OTS", "route-objective-function": "UNSPECIFIED", "service": "e366802d-0390-4a3c-b80c-1f882794c0c"}, {"type": "PROVISION_MEDIA_CHANNEL_OLS_PATH", "layer": "OPTICAL", "content": [{"src-sip-uid": "95b9727-c673-5297-86a7-faaae7ea39cb", "dest-sip-uid": "d3d68ec-43f2-50bc-baf4-ab0fffd475f4", "direction": "UNIDIRECTIONAL", "layer-protocol-name": "DSR", "layer-protocol-qualifier": "tapi-dsr:digital_signal_type_unspecified", "capacity": "100-Gbps", "service": "14c8dc97-c58a-460d-9cec-bd4eff6a4e5", "tenant-uid": "req-left-multiband", "url": "http://172.24.36.54:4900/restconf/data/tapi-common:context/tapi-connectivity:connectivity-context"}, {"type": "PROVISION_MEDIA_CHANNEL_OLS_PATH", "layer": "OPTICAL", "content": [{"src-sip-uid": "60888310-0800-0800-0800-080808080808", "dest-sip-uid": "00000010-0000-0000-0000-000000000001", "direction": "BIDIRECTIONAL", "layer-protocol-name": "PHOTONIC_MEDIA", "layer-protocol-qualifier": "tapi-photonic-media:PHOTONIC_LAYER_QUALIFIER_OCH", "capacity": "50-GHz", "lower-frequency-mhz": "19375000", "upper-frequency-mhz": "193125000", "adjustment-granularity": "6.6_25GHz", "grid-type": "FLEX", "service": "95948296-ca2c-48f7-9549-e59a00f64a03"}, {"type": "PROVISION_MEDIA_CHANNEL_OLS_PATH", "layer": "OPTICAL", "content": [{"src-sip-uid": "95948296-ca2c-48f7-9549-e59a00f64a03", "dest-sip-uid": "95948296-ca2c-48f7-9549-e59a00f64a03", "direction": "BIDIRECTIONAL", "layer-protocol-name": "PHOTONIC_MEDIA", "layer-protocol-qualifier": "tapi-photonic-media:PHOTONIC_LAYER_QUALIFIER_OTS", "route-objective-function": "UNSPECIFIED", "service": "94033e13-d54d-4e6e-858e-da809cbfb51e"}]}]}]}
INFO - Selected Realizer: OPTIC
INFO - Processing rule 8 for optical slice
INFO - Rule content: {"src-sip-uid": "975f0b9a-389f-51d6-989f-18c7c3b0829", "dest-sip-uid": "f71a545e-7555-53ce-861e-113db1853a2c", "direction": "UNIDIRECTIONAL", "layer-protocol-name": "PHOTONIC_MEDIA", "layer-protocol-qualifier": "tapi-photonic-media:PHOTONIC_LAYER_QUALIFIER_OTS", "route-objective-function": "UNSPECIFIED", "service": "94033e13-d54d-4e6e-858e-da809cbfb51e"}
INFO - Service to send: {
  "services": [
    {
      "service_id": {
        "context_id": {
          "context_uid": {
            "uid": "admin"
          }
        }
      },
      "service_uid": {
        "uid": "94033e13-d54d-4e6e-858e-da809cbfb51e"
      }
    }
  ],
  "service_type": 12,
  "service_status": {
    "service_status": 1
  },
  "service_endpoint_ids": [
    {
      "device_id": {
        "device_uid": {
          "uid": "OPTICAL-CONTROLLER"
        }
      }
    }
  ]
}

```

Figure 2.3-3: Network Orchestrator: NSC logs.

## Services

+ Add New Service 6 services found in context admin

UUID	Name	Type	End points	Status
14c8dc97-c58a-460d-9cec-bd4eff6a4e5	14c8dc97-c58a-460d-9cec-bd4eff6a4e5	TAPI_LSP	<ul style="list-style-type: none"> <li>ROADM-2/VECH-1-13-CS-1 / Device: <a href="#">OPTICAL-CONTROLLER</a></li> <li>mgmt / Device: <a href="#">TFS-PACKET</a></li> </ul>	ACTIVE
94033e13-d54d-4e6e-858e-da809cbfb51e	94033e13-d54d-4e6e-858e-da809cbfb51e	TAPI_LSP	<ul style="list-style-type: none"> <li>ROADM-2/VECH-1-13-CS-1 / Device: <a href="#">OPTICAL-CONTROLLER</a></li> <li>mgmt / Device: <a href="#">TFS-PACKET</a></li> </ul>	ACTIVE
95948296-ca2c-48f7-9549-e59a00f64a03	95948296-ca2c-48f7-9549-e59a00f64a03	TAPI_LSP	<ul style="list-style-type: none"> <li>ROADM-2/VECH-1-13-CS-1 / Device: <a href="#">OPTICAL-CONTROLLER</a></li> <li>mgmt / Device: <a href="#">TFS-PACKET</a></li> </ul>	ACTIVE

Figure 2.3-4: Network Orchestrator: TFS WebUI Services Tab.

Figure 2.3-5 shows the Grafana dashboard created by the telemetry system for monitoring different parameters from the Edgework switches.



Figure 2.3-5: Edgecore monitoring dashboard.

Figure 2.3-6 shows the network topology retrieved by the optical controller that comprises of three domains: SDM-MB domain, C-band domain, and SDM domain (Virtual). Figure 2.3-7 shows six different services that are established in the multidomain network. Furthermore, the graphical user interface (GUI) of optical controller also shows the spectrum usage in the data plane and channel allocation as depicted in Figure 2.3-8.

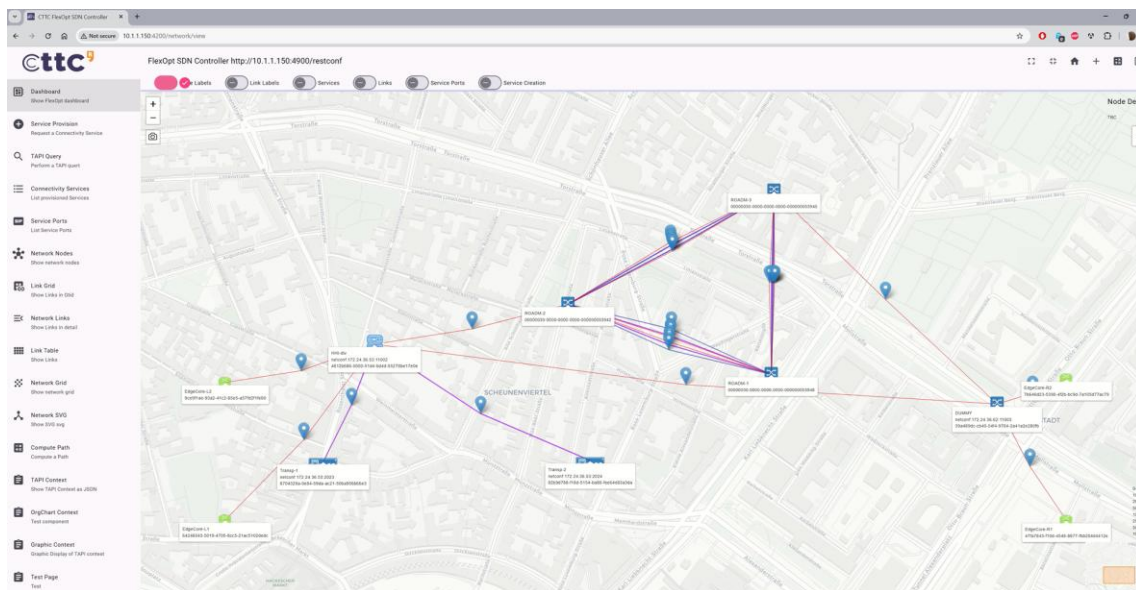


Figure 2.3-6: Multidomain optical network composed of three domains as retrieved by the optical controller using TAPI standard NBI.

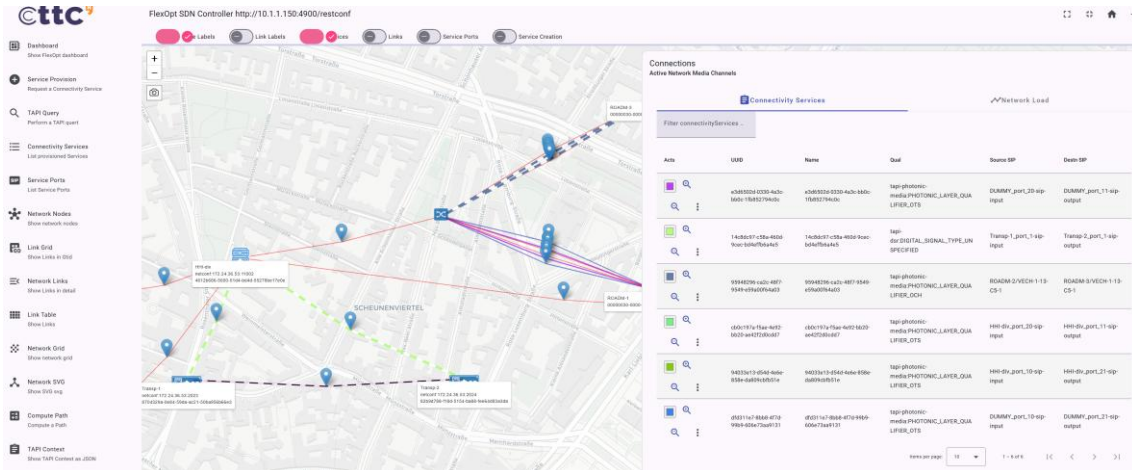


Figure 2.3-7: View of six different services established in the multidomain optical MBoSDM network.

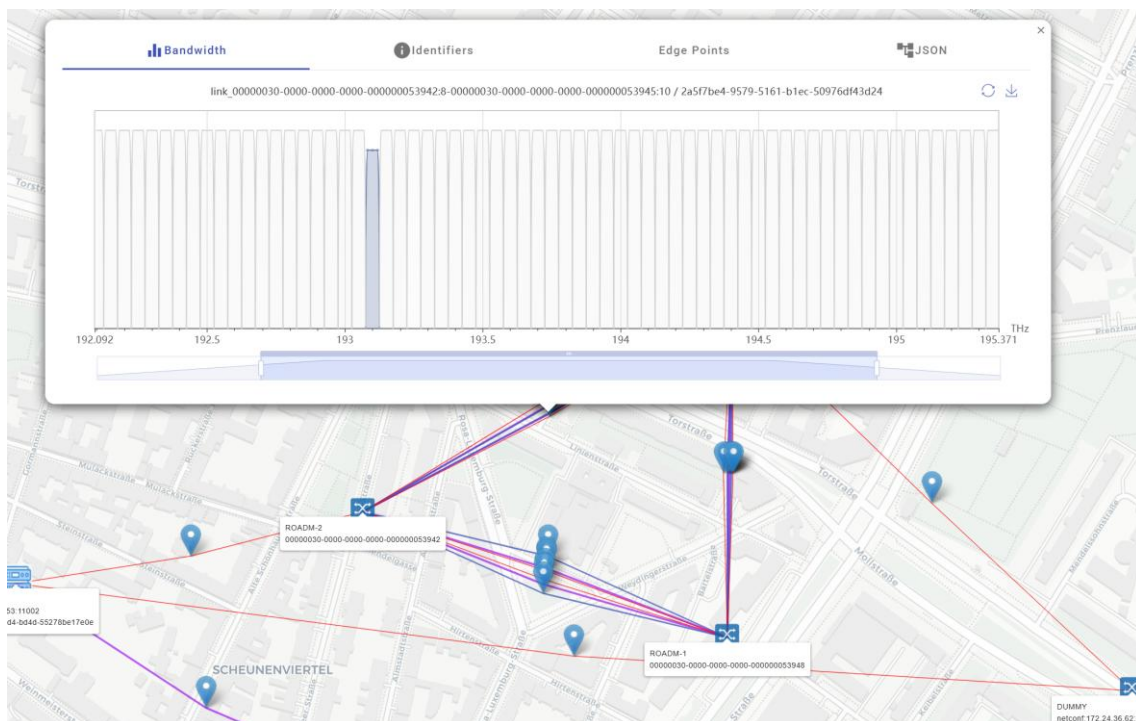


Figure 2.3-8: View of the spectrum usage between ADTRAN ROADMs and the allocation of the channel.

Figure 2.3-9 presents the resulting optical-domain topology as discovered by ONOS, including the NETCONF devices and their logical links. Figure 2.3-10 shows the flows installed by ONOS on a NETCONF-managed optical device, together with the NETCONF messages exchanged during configuration.

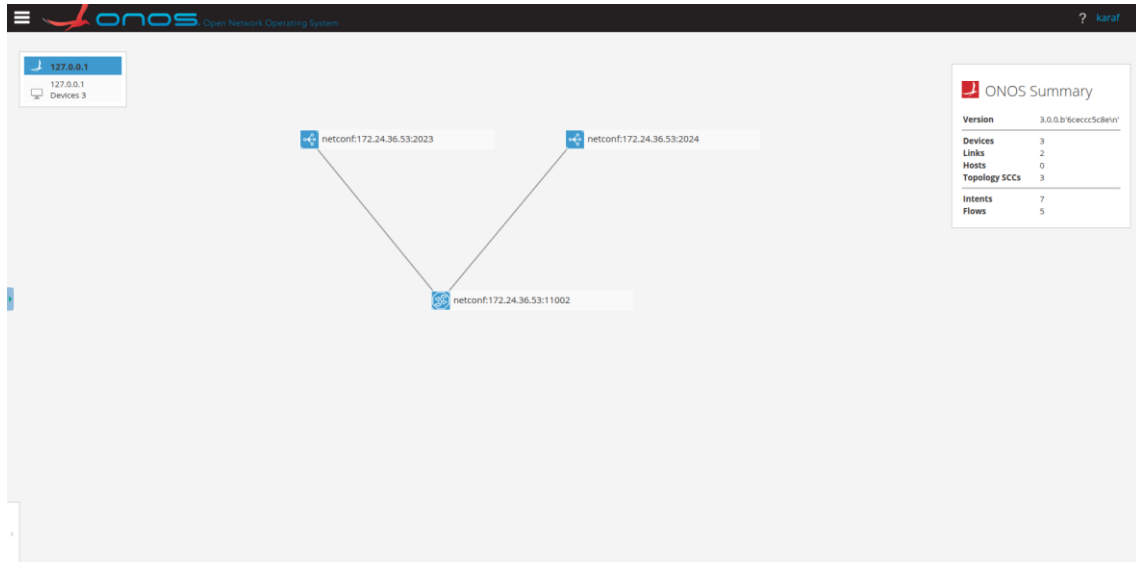


Figure 2.3-9: ONOS Topology View.

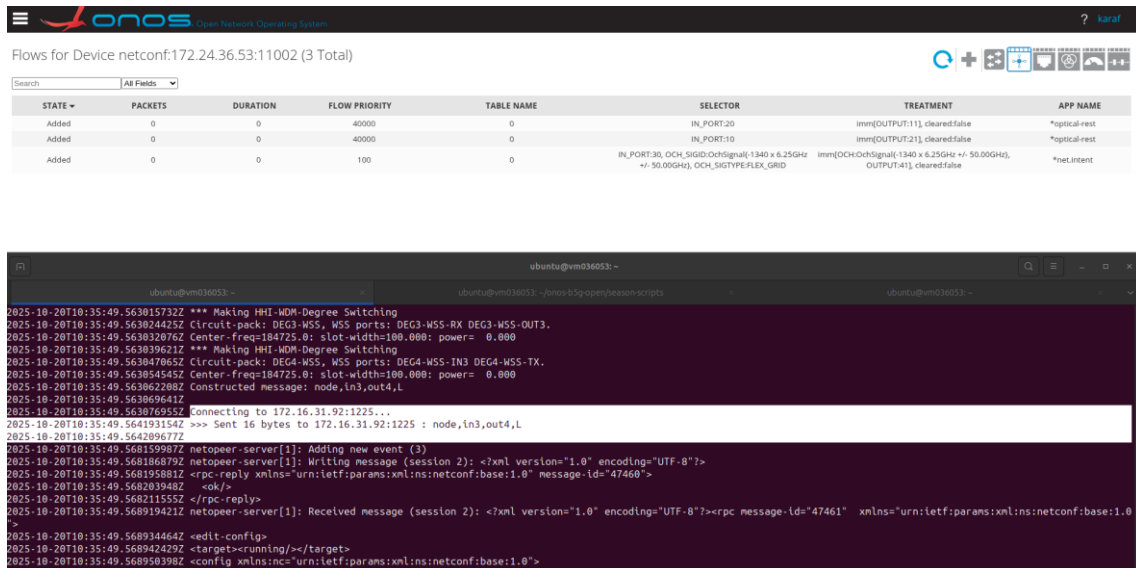


Figure 2.3-10: ONOS Flow View and NETCONF Activity.

Figure 2.3-11 shows the NetDevOps pipelines for creating, deleting, and rolling back a service, running in a GitLab environment.

Status	Pipeline	Created by	Stages	Actions
<p>Passed</p> <p>00:01:10</p> <p>39 seconds ago</p>	<p>Commit: Rollback from the Commit-id: 9...</p> <p>#2715075 P main 44e0817f</p> <p>latest branch</p>			
<p>Passed</p> <p>00:00:57</p> <p>4 minutes ago</p>	<p>Commit: {'create': 'SEASON_DEMO'}</p> <p>#2715054 P main 2ce0873a</p> <p>branch</p>			
<p>Passed</p> <p>00:00:35</p> <p>5 minutes ago</p>	<p>Commit: {'delete': 'SEASON_DEMO/SEA...'}</p> <p>#2715051 P main 35583ef3</p> <p>branch</p>			
<p>Passed</p> <p>00:00:34</p> <p>6 minutes ago</p>	<p>Commit: {'create': 'SEASON_DEMO'}</p> <p>#2715037 P main 98a84489</p> <p>branch</p>			

Figure 2.3-11: NetDevOps pipelines (Create, Delete, and Rollback service) and stages in execution of each pipeline.

After the optical controller sends a service request to NetDevOps, the MNC controller provisions an optical channel between ROADM2 and ROADM3 (via direct link) to support end-to-end video streaming, as shown in Figure 2.3-12. To illustrate the use case, a dashboard has been prepared (see Figure 2.3-13), which enables the following functionalities:

**Failure Injection:** Introduces a failure in the OLS on the existing service, as shown in Figure 2.3-12, resulting in a drop in performance data from ROADM2 toward ROADM3.

**Service Reroute:** Deletes the existing service between ROADM2 and ROADM3 and creates an alternate path via ROADM2–ROADM1–ROADM3 (160 km). This operation is performed using NetDevOps, and the change in service path is visible in the controller (see Figure 2.3-14). For every network operation, NetDevOps captures a snapshot of the network state at that time using a commit ID, ensuring uninterrupted service even in case of failure.

**Failure Fix:** The dashboard provides an option to resolve the issue triggered between ROADM2 and ROADM3.

**Rollback to Original State:** If the operator wishes to revert to the original state, the commit ID corresponding to the network state with the service between ROADM2 and ROADM3 is sufficient. NetDevOps automatically performs all device and service configurations according to the stored network state which can be seen in Figure 2.3-11. The demo showcases rollback to the original 80 km path between ROADM2 and ROADM3.

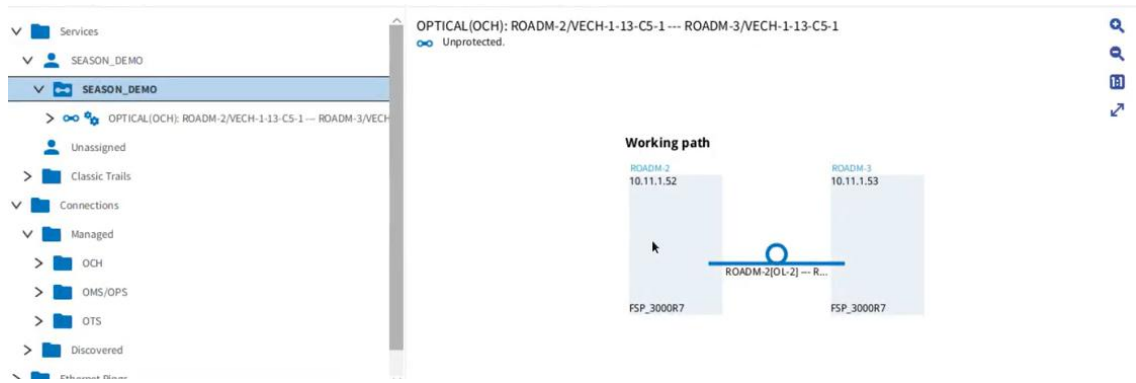


Figure 2.3-12: MNC controller: service provisioning between ROADM2 and ROADM3 via direct link.

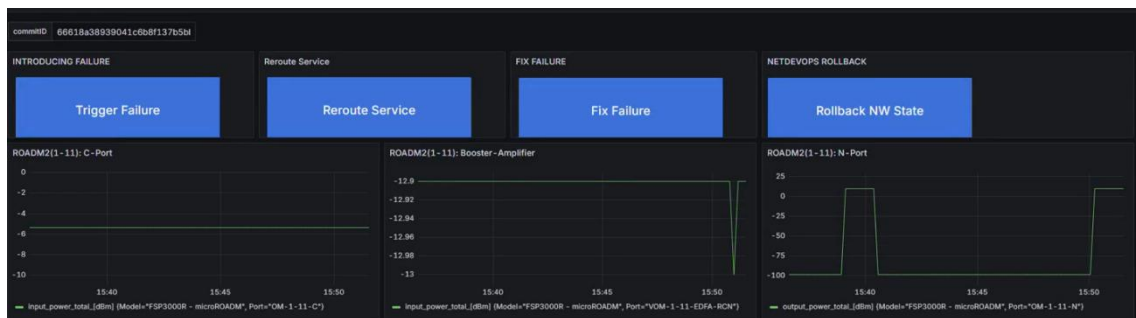


Figure 2.3-13: Grafana Dashboard to trigger failure, reroute-, fix-, and rollback-service in the network.

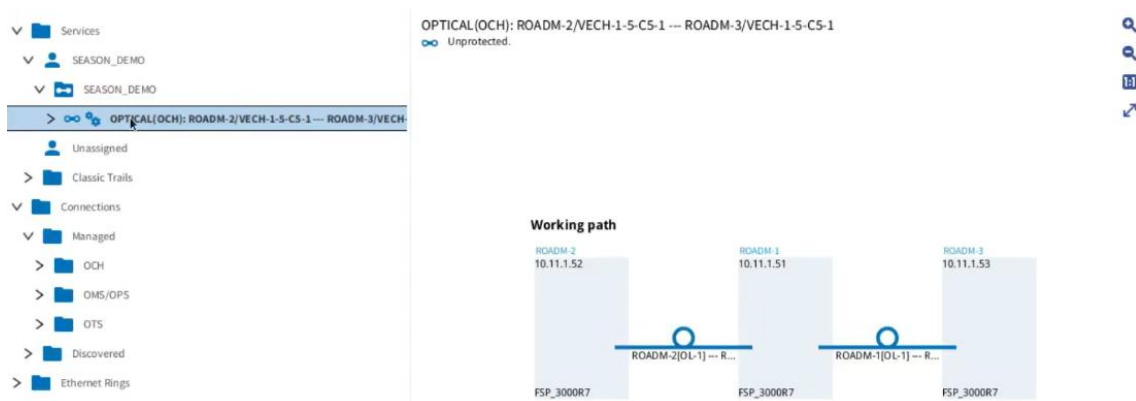


Figure 2.3-14: MNC controller: service provisioning between ROADM2 and ROADM3 via ROADM1 due to failure.

Figure 2.3-15 shows the commands received from the ONOS instance controlling the MBoSDM node, in the running MATLAB script for configuring the node. The third command represents the MB study case where the L-band was routed from input3 of the node to output4 establishing the band-selective routing capability. Commands 1 and 2 represent the fiber switching case, where the service was established in the red path and the green bath according to the demo scenario. A screenshot of the Optical Spectrum Analyzer (OSA) showing the signal from the Edgcore is shown in Figure 2.3-16.

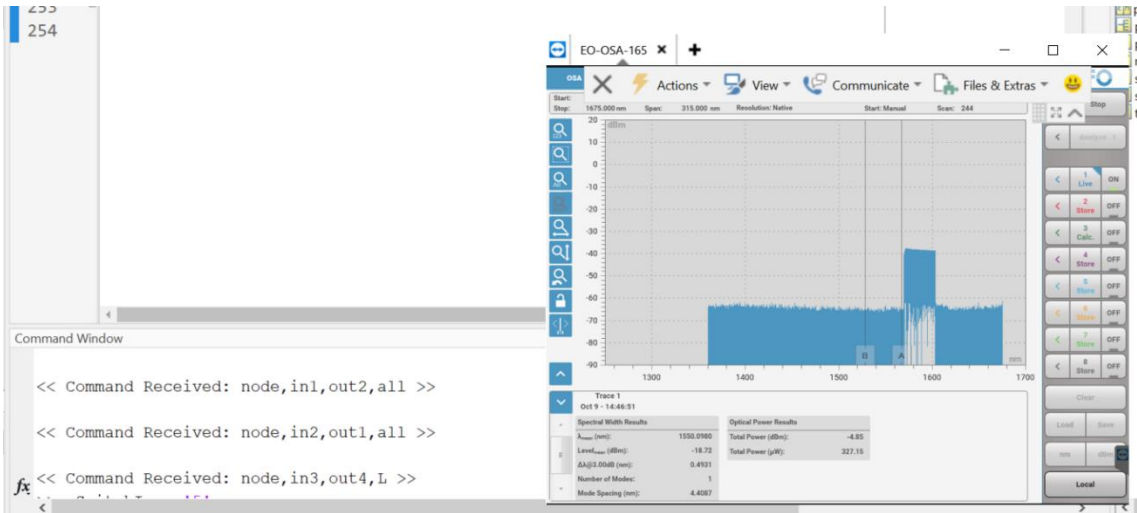


Figure 2.3-15: Commands received and MB routing.

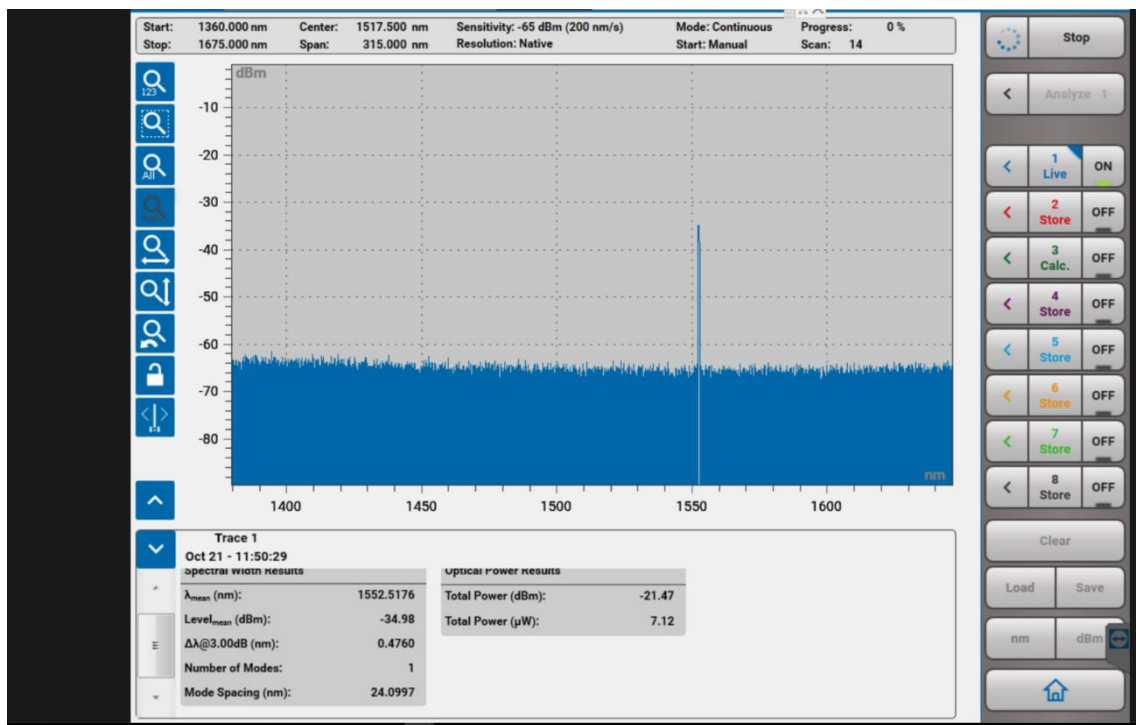


Figure 2.3-16: OSA screenshot of the signal routed from the Edgecore through the MBoSDM node.

During the demo experiment, two main operations have been considered: the configuration of the pluggable transceivers (i.e., setting frequency and Tx power level) via NETCONF/REST, and the monitoring capability to check the behavior of the end-to-end optical service. Figure 2.3-17 shows the command line interface (CLI) view of the second Edgecore switch on the right (see Figure 2.1-1), highlighting two main details: the status of the interface and the performance of the connection that is transmitted via telemetry to the collector. To the top of the figure, interface Ethernet16, the only one equipped with a ZR+ QSFP-DD pluggable transceiver is operationally up. To the bottom of the figure, the telemetry details are shown, highlighting that

the Rx power level is 3.12 dBm, the pre-FEC BER is in the order of 7.6E-3, the OSNR is around 24.7 dB and the electrical SNR is 14.3 dB.

```

admin@EDGE-SW-SEASON2:~$ show int stat
-----
Interface          Speed  MTU  Oper  FEC      Alias  Vlan  Oper  Admin  ProtoDown  Eff Admin  Type  Asym FEC
-----
Ethernet0          73, 74, 75, 76, 77, 78, 80  4000  9100  rs      Eth1(Per11)  routed  down  up      False      up          N/A    N/A
Ethernet10         50, 60, 67, 68, 69, 70, 71, 72  4000  9100  rs      Eth2(Per12)  routed  down  up      False      up          N/A    N/A
Ethernet16         81, 82, 83, 84, 85, 86, 87, 88  4000  9100  none    Eth3(Per13)  routed  down  up      False      up          N/A    N/A
Ethernet17         260  100  9100  none    Eth3(Per14)  routed  down  up      False      up          N/A    N/A
admin@EDGE-SW-SEASON2:~$ show int stat
-----
Interface          Speed  MTU  Oper  FEC      Alias  Vlan  Oper  Admin  ProtoDown  Eff Admin  Type  Asym FEC
-----
Ethernet0          73, 74, 75, 76, 77, 78, 80  4000  9100  rs      Eth1(Per11)  routed  down  up      False      up          N/A    N/A
Ethernet10         50, 60, 67, 68, 69, 70, 71, 72  4000  9100  rs      Eth2(Per12)  routed  down  up      False      up          N/A    N/A
Ethernet16         81, 82, 83, 84, 85, 86, 87, 88  4000  9100  rs      Eth3(Per13)  routed  down  up      False      up          N/A    N/A
Ethernet17         260  100  9100  none    Eth3(Per14)  routed  down  up      False      up          N/A    N/A
Ethernet18         99, 99, 91, 92, 93, 94, 95, 96  4000  9100  rs      Eth4(Per14)  routed  down  up      False      up          N/A    N/A
Ethernet19         97, 98, 99, 100, 101, 102, 103, 104  4000  9100  rs      Eth5(Per15)  routed  down  up      False      up          N/A    N/A
Ethernet20         105, 106, 107, 108, 109, 110, 111, 112  4000  9100  rs      Eth6(Per16)  routed  down  up      False      up          N/A    N/A
Ethernet21         113, 114, 115, 116  1000  9100  none    Eth7(Per17)  routed  down  up      False      up          N/A    N/A
Ethernet22         111, 112, 113, 114, 115, 116, 117, 118  4000  9100  rs      Eth8(Per18)  routed  down  up      False      up          N/A    N/A
Ethernet23         81, 82, 83, 84, 85, 86, 87, 88  4000  9100  rs      Eth9(Per19)  routed  down  up      False      up          N/A    N/A
Ethernet24         33, 34, 35, 36, 37, 38, 39, 40  4000  9100  rs      Eth10(Per10)  routed  down  up      False      up          N/A    N/A
Ethernet25         89, 89, 81, 82, 83, 84, 85, 86  4000  9100  rs      Eth11(Per11)  routed  down  up      False      up          N/A    N/A
Ethernet26         57, 58, 59, 60, 61, 62, 63, 64  4000  9100  rs      Eth12(Per12)  routed  down  up      False      up          N/A    N/A
Ethernet27         129, 130, 131, 132, 133, 134, 135, 136  4000  9100  rs      Eth13(Per13)  routed  down  up      False      up          N/A    N/A
Ethernet28         137, 138, 139, 140, 141, 142, 143, 144  4000  9100  rs      Eth14(Per14)  routed  down  up      False      up          N/A    N/A
Ethernet29         145, 146, 147, 148, 149, 150, 151  4000  9100  rs      Eth15(Per15)  routed  down  up      False      up          N/A    N/A
Ethernet30         153, 154, 155, 156, 157, 158, 159, 160  4000  9100  rs      Eth16(Per16)  routed  down  up      False      up          N/A    N/A
Ethernet31         169, 170, 171, 172, 173, 174, 175, 176  4000  9100  rs      Eth17(Per17)  routed  down  up      False      up          N/A    N/A
Ethernet32         181, 182, 183, 184, 185, 186, 187, 188  4000  9100  rs      Eth18(Per18)  routed  down  up      False      up          N/A    N/A
Ethernet33         177, 178, 179, 180, 181, 182, 183, 184  4000  9100  rs      Eth19(Per19)  routed  down  up      False      up          N/A    N/A
Ethernet34         185, 186, 187, 188, 189, 190, 191, 192  4000  9100  rs      Eth20(Per20)  routed  down  up      False      up          N/A    N/A
Ethernet35         193, 194, 195, 196, 197, 198  4000  9100  rs      Eth21(Per21)  routed  down  up      False      up          N/A    N/A
Ethernet36         9, 10, 11, 12, 13, 14, 15, 16  4000  9100  rs      Eth22(Per22)  routed  down  up      False      up          N/A    N/A
Ethernet37         17, 18, 19, 20, 21, 22, 23, 24  4000  9100  rs      Eth23(Per23)  routed  down  up      False      up          N/A    N/A
Ethernet38         25, 26, 27, 28, 29, 30, 31, 32  4000  9100  rs      Eth24(Per24)  routed  down  up      False      up          N/A    N/A
Ethernet39         33, 34, 35, 36, 37, 38, 39, 40  4000  9100  rs      Eth25(Per25)  routed  down  up      False      up          N/A    N/A
Ethernet40         103, 104, 105, 106, 107, 108, 109, 110  4000  9100  rs      Eth26(Per26)  routed  down  up      False      up          N/A    N/A
Ethernet41         117, 118, 119, 120, 121, 122, 123, 124  4000  9100  rs      Eth27(Per27)  routed  down  up      False      up          N/A    N/A
Ethernet42         129, 130, 131, 132, 133, 134, 135, 136  4000  9100  rs      Eth28(Per28)  routed  down  up      False      up          N/A    N/A
Ethernet43         137, 138, 139, 140, 141, 142, 143, 144  4000  9100  rs      Eth29(Per29)  routed  down  up      False      up          N/A    N/A
Ethernet44         145, 146, 147, 148, 149, 150, 151, 152  4000  9100  rs      Eth30(Per30)  routed  down  up      False      up          N/A    N/A
Ethernet45         153, 154, 155, 156, 157, 158, 159, 160  4000  9100  rs      Eth31(Per31)  routed  down  up      False      up          N/A    N/A
Ethernet46         169, 170, 171, 172, 173, 174, 175, 176  4000  9100  rs      Eth32(Per32)  routed  down  up      False      up          N/A    N/A
Ethernet47         181, 182, 183, 184, 185, 186, 187, 188  4000  9100  none    Eth33(Per33)  routed  down  up      False      up          N/A    N/A
Ethernet48         241, 242, 243, 244, 245, 246, 247, 248  4000  9100  none    Eth34(Per34)  routed  down  up      False      up          N/A    N/A
Ethernet49         260  100  9100  none    Eth34(Per34)  routed  down  up      False      up          N/A    N/A
admin@EDGE-SW-SEASON2:~$
-----
admin@EDGE-SW-SEASON2:~$
-----
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.84, 'sig-rx-power': -17.01, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.77, 'sig-rx-power': -17.03, 'BER': 0.5, 'osnr': 0.0, 'esnr': 0.0 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.82, 'sig-rx-power': -3.12, 'BER': 0.00738, 'osnr': 24.3, 'esnr': 14.3 }
Telemetry thread 1 --> { 'tx-power': 0.0, 'rx-power': 2.82, 'sig-rx-power': -3.12, 'BER': 0.00762, 'osnr': 24.7, 'esnr': 14.3 }

```

Figure 2.3-17: CLI view of the Edgecore switch.

Finally, some timing measurements were also performed while running the demo as given below:

- Initial service provisioning time on red path between ROADM2 and ROADM3 (direct) was 1 minute 30 seconds.
- When the first failure occurred, the service reconfiguration time on red path between ROADM2 and ROADM3 (via ROADM1) was 1 minute 15 seconds.
- When the second failure occurred, the service reconfiguration time on green path was 45 seconds.
- On fixing the failures, the service recovery time for rolling back to the original path was 50 seconds.

Dissemination Level	PU
---------------------	----

## 3 DEMONSTRATION OF DYNAMIC MULTIBAND AND SDM SERVICE PROVISIONING

---

This chapter describes the MBoSDM demo of SEASON project that was performed at CTTC.

### 3.1 DEMO ARCHITECTURE AND COMPONENTS

The second complementary demonstration regarding to MBoSDM dynamic service provisioning aims at validating the node prototype that has been developed in the SEASON project.

#### 3.1.1 Control Plane

The implemented control plane has several components that we describe next.

- **MBoSDM SDN Controller:** the MBoSDM SDN controller is based on the CTTC FlexOpt SDN controller, properly extended to support multiple switching layers. This component has been developed in the scope of WP4 and was demonstrated in D5.1, with a large topology such as Telefonica network. In the first reporting period, the SDN Controller had not yet been integrated with the hardware. In the second reporting period, we have completed the integration.
- **MB(oSDM) S-BVT agent:** a dedicated S-BVT SDN agent has been designed and implemented. It is based on the usage of the netopeer framework to support the NETCONF protocol with the OpenConfig Terminal Device and Platform optical YANG models.
- **MBoSDM node agent:** likewise, an SDN agent for the MBoSDM node prototype has been developed. It also uses the netopeer framework supporting the NETCONF protocol but in this case, it utilizes a SEASON-defined YANG model that allows specific operations to be performed at e.g. the SDM/core or multi-band flexi-grid level. The YANG model is the reference and contract between the SDN controller and the agent, and supports the identified Create/Read/Update/Delete (CRUD) operations on cross-connections, with constraints in terms of input ports/output ports and type/layer of cross-connection.

The following snippet shows the YANG model in tree format.

```

module: season-mbosdm-node
  +--rw node
  |   +--rw address      inet:ip-address
  |   +--rw node_id     yang:uuid
  |   +--ro ports
  |   |   +--ro port* [id]
  |   |   |   +--ro id      uint32
  |   |   |   +--ro name?   string
  |   |   |   +--ro direction? season-mbosdm-types:port-direction
  |   |   |   +--ro type?   season-mbosdm-types:port-type
  |   +--rw connections
  |   |   +--rw connection* [name]

```

	+-rw name	string
	+-rw input_port	uint32
	+-rw output_port	uint32
	+-rw input_core	uint32
	+-rw output_core	uint32
	+-rw band?	enumeration
	+-rw n	int16
	+-rw m	uint16
+--ro info		
	+--ro software_version?	string
	+--ro hardware_version?	string

### 3.1.2 Data Plane

- **MB(oSDM) S-BVT prototype (WP3 component C3.4)**

The MB(oSDM) S-BVT prototype, which operates in the C+L+S-bands, has been setup for the demonstration activities. This component has been developed within WP3, and all the details and experimental validation are included in the corresponding WP3 deliverables (D3.1-D3.3). In the second reporting period, the transceiver prototype has been integrated with the MBoSDM SDN controller and corresponding agent (see previous section 3.1.1).

- **MBoSDM node prototype (WP3 component C3.1)**

In the SEASON project, an MBoSDM node prototype has been designed and implemented towards addressing the growing capacity and traffic demands of future optical networks. More details and features of the proposed switching solution can be found in deliverable D3.2 and D3.3. This component has been developed and validated in WP3. In the second reporting period, it has been integrated with the MBoSDM controller and related agent (see previous section 3.1.1).

## 3.2 EXPERIMENTAL VALIDATION

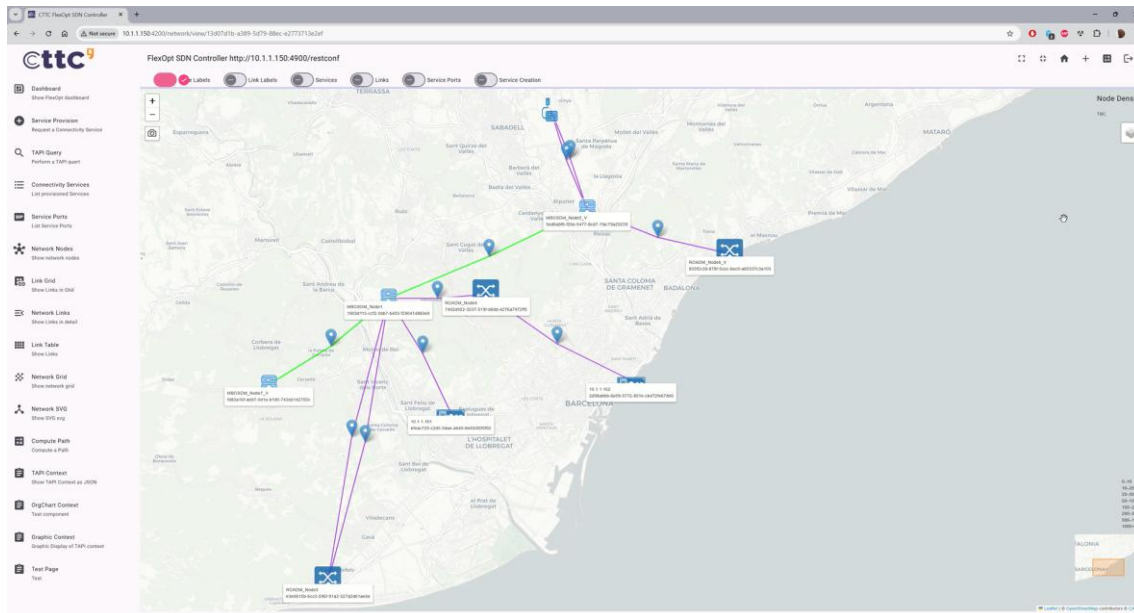


Figure 3.2-1: Scenario for the CTC demo with the MBoSDM node.

The experimental validation was conducted in the ADRENALINE<sup>®</sup> testbed to demonstrate the dynamic provisioning capabilities of the SEASON MBoSDM architecture and validate the integration of control and data plane components. The assessed MBoSDM scenario includes real hardware elements, such as real nodes (*MBoSDM node 1*, *ROADM4*, *SBVT Tx*, *SBVT Rx*) as well as additional (virtual) nodes to illustrate more complex cases without the need for (unavailable) hardware. Specifically, *MBoSDM node 1* is the SEASON node prototype, developed in the framework of WP3. *ROADM4* is a node from the ADRENALINE<sup>®</sup> testbed available at CTC premises, where the above-mentioned SEASON node prototype has been also deployed (see deliverable D3.3). *SBVT Tx* and *SBVT Rx* are two nodes where the respective SEASON transmitter and receiver MB prototypes are available. The MB(oSDM) S-BVT has been also developed as a SEASON prototype in WP3 context. Two different use cases have been demonstrated for considering both an SDM and a multiband flexi-grid services.

### 3.2.1 Initialization phase

During the initialization phase, the SDN controller establishes NETCONF sessions with all devices in the scenario and performs a comprehensive discovery process. This includes retrieving device capabilities, enumerating available components and ports, and identifying supported operational modes and configuration parameters. The collected information is used to build an accurate topology and resource inventory, ensuring that subsequent service provisioning operations are consistent with the actual hardware and software capabilities. As detailed in Section 3.1.1, the SDN controller establishes connections to the various SDN agents within the

scenario via the NETCONF protocol. The operational status and information of these agents are also reflected in the controller's logs. A visualization of this communication and the supported operational modes of the S-BVT transceiver is provided in Figure 3.2-2.

```

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <get>
    <filter type="subtree">
      <terminal-device xmlns="http://openconfig.net/yang/terminal-device" />
    </filter>
  </get>
</rpc>
]]>]]>
<?xml version="1.0"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <terminal-device xmlns="http://openconfig.net/yang/terminal-device">
      <operational-modes>
        <mode>
          <mode-id>0</mode-id>
          <state>
            <mode-id>0</mode-id>
            <description>Laser OFF</description>
          </state>
        </mode>
        <mode>
          <mode-id>100</mode-id>
          <state>
            <mode-id>100</mode-id>
            <description>Laser ON</description>
          </state>
        </mode>
      </operational-modes>
    </terminal-device>
  </data>
</rpc-reply>
]]>]]>

```

Figure 3.2-2: Supported modes of the S-BVT.

Figure 3.2-1 shows SDN controller interface during the initialization phase, where the controller has successfully discovered the network topology and device capabilities. Detailed information about the selected node, including its network ports and associated attributes such as operational state, supported modes, and graphical representations of available cores and spectrum slots can be seen. This visualization, depicted in Figure 3.2-3, confirms that the controller has retrieved all necessary resources and capabilities, enabling accurate path computation and service provisioning in subsequent steps.

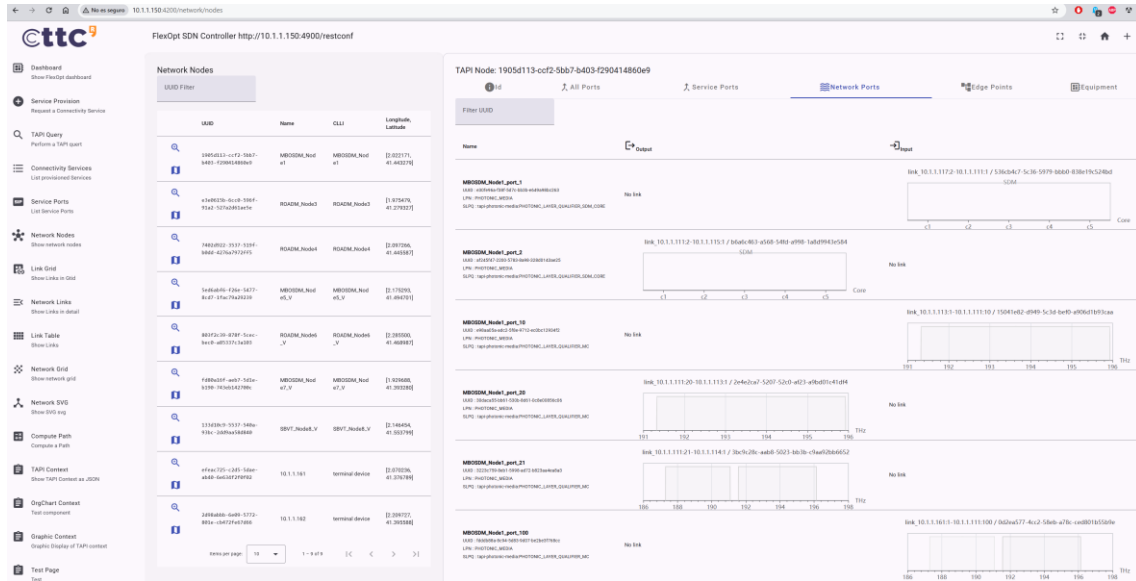


Figure 3.2-3: List of nodes in the CTTC scenario including the detail for the MBoSDM node.

### 3.3 WORKFLOW

In this section, the demo workflow is described. The demo is implemented in a video that is companion to this document. The **first use case** involves setting up an **SDM service** between two virtual *MBoSDM\_Node7\_v* and *MBoSDM\_Node5\_v* nodes, traversing the physical MBoSDM node. This process demonstrates the controller’s ability to compute the route, allocate resources at the core level, and configure the necessary cross-connections dynamically.

1. The workflow starts with the creation of an SDM service between the two virtual nodes through the SDN controller’s GUI, as depicted in Figure 3.3-1. The interface provides a structured form where key parameters for service provisioning including service data (i.e. service name, type, ingress, egress and bandwidth) can be included. The controller establishes NETCONF sessions with all devices in the scenario, including the physical MBoSDM node, and performs a discovery phase, as detailed before, to retrieve: (i) available ports and cores; (ii) supported operational modes and (iii) spectrum resources. This ensures that the controller has an accurate view of the topology and resources before provisioning.

### Service Create

Service Data
Explicit Params

Service name

d71187bc-2947-11e8-b467-0ed

Unique name for the connection

36 / 256

Bidir

Service Qualifier\*

tapi-photonic-media:PHOTONIC\_LAYER\_QUALIFIER\_SDM\_CORE

Service Qualifier

Ingress SIP\*

b2c9ea71-8b6b-51b1-ab3b-90e10b1f827b

Name: MBOSDM\_Node7\_V\_port\_2-source

IfId: 10.1.1.117:2

↑

Egress SIP\*

fb7da22a-5b60-5258-b1e6-53c0c775dfd5

Name: MBOSDM\_Node5\_V\_port\_1-sink

IfId: 10.1.1.115:1

↑

Objective Function\*

FlexOpt DSR/NMC (10000)

Bandwidth\*

1

Units\*

cores

Submit
Reset

Figure 3.3-1: Graphical User Interface for the SDM service creation.

- The SDN controller computes the optimal route and allocates the available cores for the service to meet its requirements. The GUI, depicted in Figure 3.3-2, reflects the selected service connection.

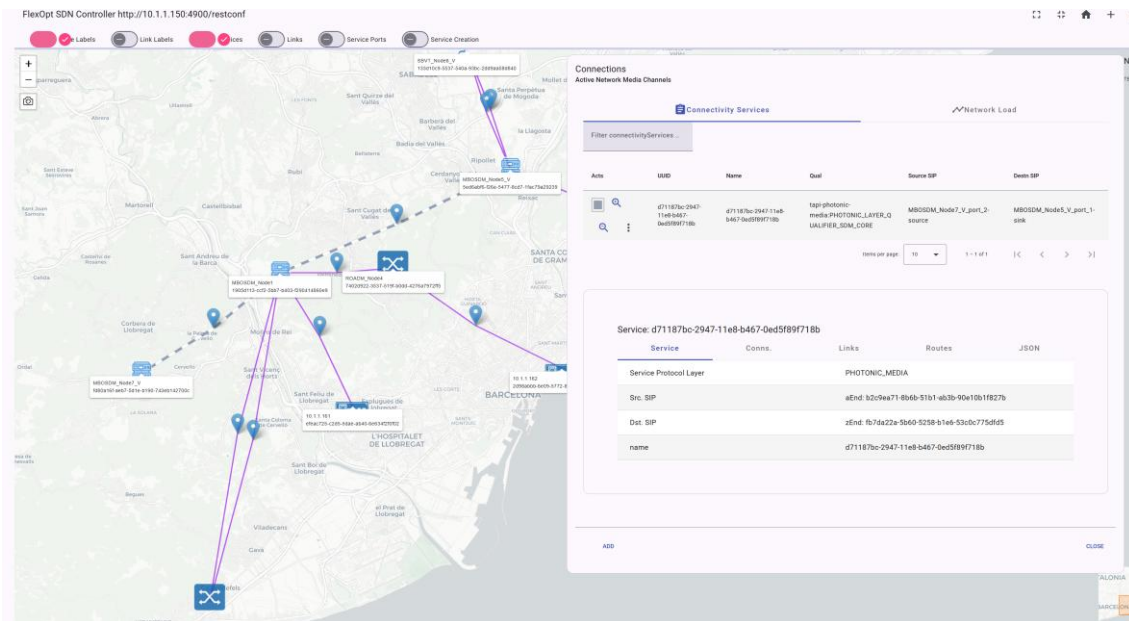


Figure 3.3-2: View of the SDM service from the GUI.

- Once the path and resources are determined, the controller sends NETCONF <edit-config> messages to the MBoSDM node using the SEASON-defined YANG model. These messages configure the necessary cross-connections at the core level, specifying: (i) input and output ports; (ii) input and output cores; (iii) additional parameters for SDM switching. Examining the logs in Figure 3.3-3, we can see the NETCONF exchange messages for the configuration of a network node connection. The figure illustrates a successful NETCONF configuration update where a new connection is merged into the device configuration. The <ok /> response confirms that the edit was applied without errors.

```
[13:00:19.250281] 3191371 [T] NETCONF Exchange - on_message
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <edit-config>
    <target>
      <running />
    </target>
    <config>
      <node xmlns="urn:season:yang:mboosdm-node" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <connections>
          <connection nc:operation="merge">
            <name>d71187bc-2947-11e8-b467-0ed5f89f718b</name>
            <input_port>1</input_port>
            <output_port>2</output_port>
            <input_core>3</input_core>
            <output_core>1</output_core>
          </connection>
        </connections>
      </node>
    </config>
  </edit-config>
</rpc>
]]]]>
<?xml version="1.0"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <ok />
</rpc-reply>
]]]]>
```

Figure 3.3-3: Logs for the NETCONF exchange for MBoSDM node configuration.

4. After configuration, the SDM service is confirmed. Figure 3.3-4 illustrates the allocation of available cores within the network links, specifically showing cores c3 and c1 assigned to the connection. The MBoSDM node 1 is actively performing core switching, as indicated by the link resource details displayed on the right panel. This panel provides a visual representation of the spectrum allocation (in THz) and the corresponding core assignments for each link, confirming how resources are distributed across the optical network.

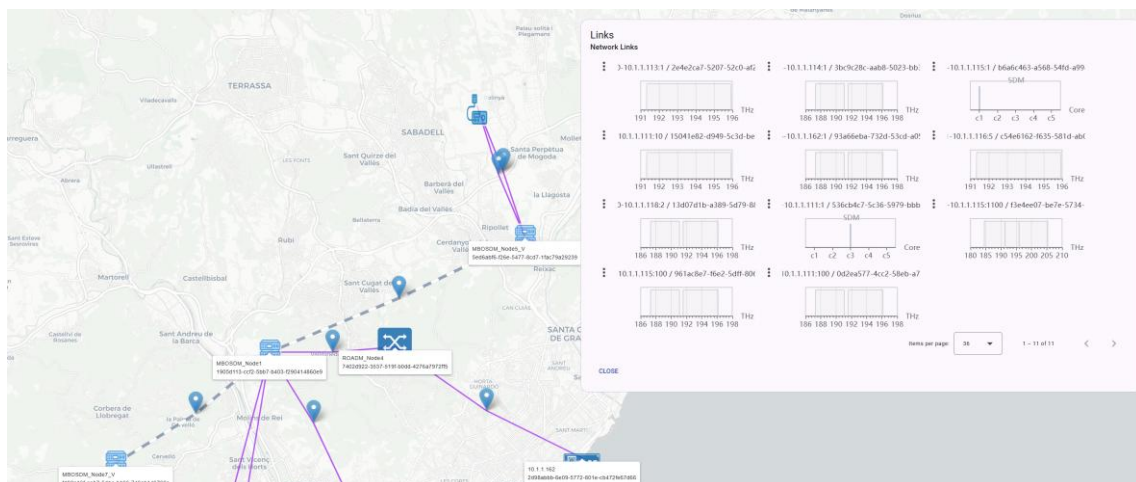


Figure 3.3-4 View of the assigned core and spectral resources of the optical network.

Dissemination Level	PU
---------------------	----

Finally, a **second service** has been established: a **multiband flexi-grid service** designed to demonstrate MB operation, as shown in Figure 3.3-5. In this scenario, the controller provisions a flex-grid connection that spans the SBVT Tx node, *MBoSDM node 1, ROADM4* (supporting both C and L bands operation) and the *S-BVT Rx node*. A multiband flow is transmitted at the *SBVT Tx node*, and the L-band component is successfully received at the *SBVT Rx node*, validating the multiband capability of the system. Mainly, the workflow includes different steps:

1. The workflow begins when the SDN controller receives a service creation request for a multiband flex-grid connection. The controller initiates NETCONF sessions with all involved devices, including the SEASON MBoSDM node 1 and the S-BVT nodes and RESTCONF for the ROADM4, and performs a comprehensive calculation phase. During this phase, the controller retrieves device capabilities, supported operational modes, available cores, and spectrum slots. This information is used to compute an optimal route and allocate resources across multiple bands (C and L).

The screenshot displays the 'Service Create' interface with the following details:

- Service Data** / **Explicit Params** tabs.
- Service name:** d71187bc-2947-11e8-b467-0ed
- Unique name for the connection:** 36 / 256
- Service Qualifier\*:** tapi-dsr:DIGITAL\_SIGNAL\_TYPE\_UNSPECIFIED
- Ingress SIP\*:** ada84e9e-41ae-5f5b-b149-cf9d89c345ba  
Name: 10.1.1.161-PORT-A-Line-1-floating-input-sip  
IfId: 10.1.1.161:4294967294
- Egress SIP\*:** fb417055-eda4-5861-88c8-930f70aaef81  
Name: 10.1.1.162-PORT-A-Line-1-floating-output-sip  
IfId: 10.1.1.162:4294967294
- Objective Function\*:** FlexOpt DSR/NMC (10000)
- Bandwidth\*:** 50
- Units\*:** GBPS
- Bidir:** Toggle switch (currently off).
- Buttons:** Submit, Reset.

Figure 3.3-5: Graphical User Interface for the MB flexi-grid service creation.

2. Once the topology and resource inventory are collected, the controller computes the path. The path used crosses ROADM4 of ADRENALINE® testbed which includes support for C+L-bands, so the controller can provision a flex-grid connection that leverages the expanded spectral resources across these bands, enabling higher capacity and flexibility for advanced optical services (depicted in Figure 3.3-6).

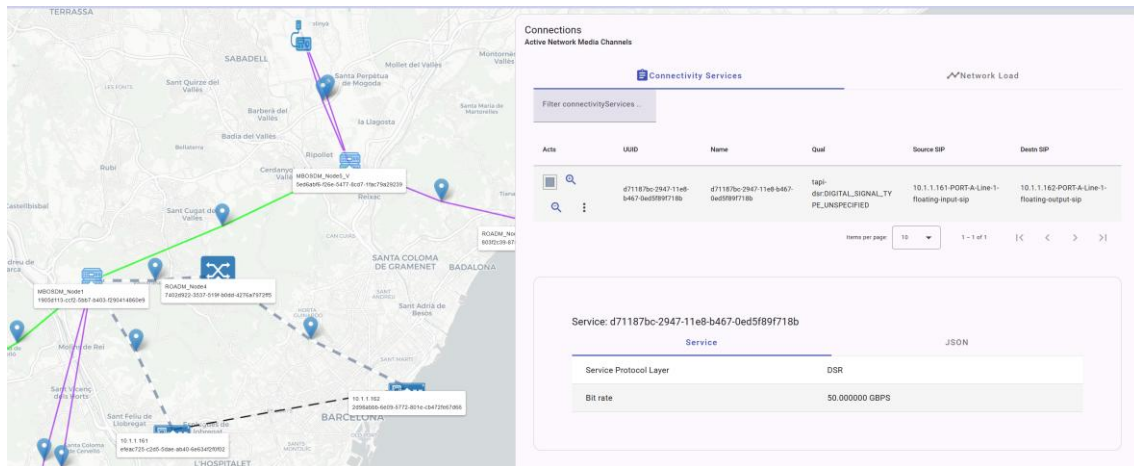


Figure 3.3-6: View of an active MB flexi-grid service.

3. The controller establishes the service connection by sending NETCONF <edit-config> messages to the devices involved:
  - a. *MBoSDM Node 1*: This node was configured using the SEASON-defined YANG model to establish multiband cross-connections. The corresponding NETCONF <edit-config> message for this operation (see Figure 3.3-7) can be compared to the previous use case (Figure 3.3-3). A key distinction is the absence of the "n" and "m" parameters in the earlier message, which is logical as these parameters are specific to the requirements of this multiband flexi-grid service scenario.
  - b. *SBVT Tx* and *SBVT Rx*: Configured using OpenConfig Terminal device YANG models to set parameters such as frequency, target output power, and operational mode (see Figure 3.3-8).

These configurations enable the creation of a multiband optical channel that spans both C and L bands, leveraging the extended spectral resources for higher capacity.

```

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <edit-config>
    <target>
      <running />
    </target>
    <config>
      <node xmlns="urn:season:yang:mboosdm-node" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <connections>
          <connection nc:operation="merge">
            <name>d71187bc-2947-11e8-b467-0ed5f89f718b</name>
            <input_port>100</input_port>
            <output_port>21</output_port>
            <input_core>0</input_core>
            <output_core>0</output_core>
            <n>-920</n>
            <m>8</m>
          </connection>
        </connections>
      </node>
    </config>
  </edit-config>
</rpc>
]]]]>
<?xml version="1.0"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <ok />
</rpc-reply>
]]]]>

```

Figure 3.3-7: NETCONF <edit-config> message for configuring an MBoSDM node connection.

```

[13:01:37.937872] 3191371 [T] [efead725-c2d5-5dae-ab40-6e634f2f0f02] sending message <?xml version="1.0" encoding="
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running />
    </target>
    <config>
      <components xmlns="http://openconfig.net/yang/platform">
        <component operation="merge">
          <name>0CH-A-Line-1</name>
          <oc-opt-term:optical-channel xmlns:oc-opt-term="http://openconfig.net/yang/terminal-device">
            <oc-opt-term:config>
              <oc-opt-term:frequency>187350000</oc-opt-term:frequency>
              <oc-opt-term:target-output-power>6</oc-opt-term:target-output-power>
              <oc-opt-term:operational-mode>100</oc-opt-term:operational-mode>
            </oc-opt-term:config>
          </oc-opt-term:optical-channel>
        </component>
      </components>
    </config>
  </edit-config>
</rpc>

```

Figure 3.3-8: NETCONF <edit-config> message for configuring an optical channel using OpenConfig YANG models.

4. After successful configuration, the multiband flow is transmitted across the provisioned path. The allocated spectral resources can be seen in Figure 3.3-9.

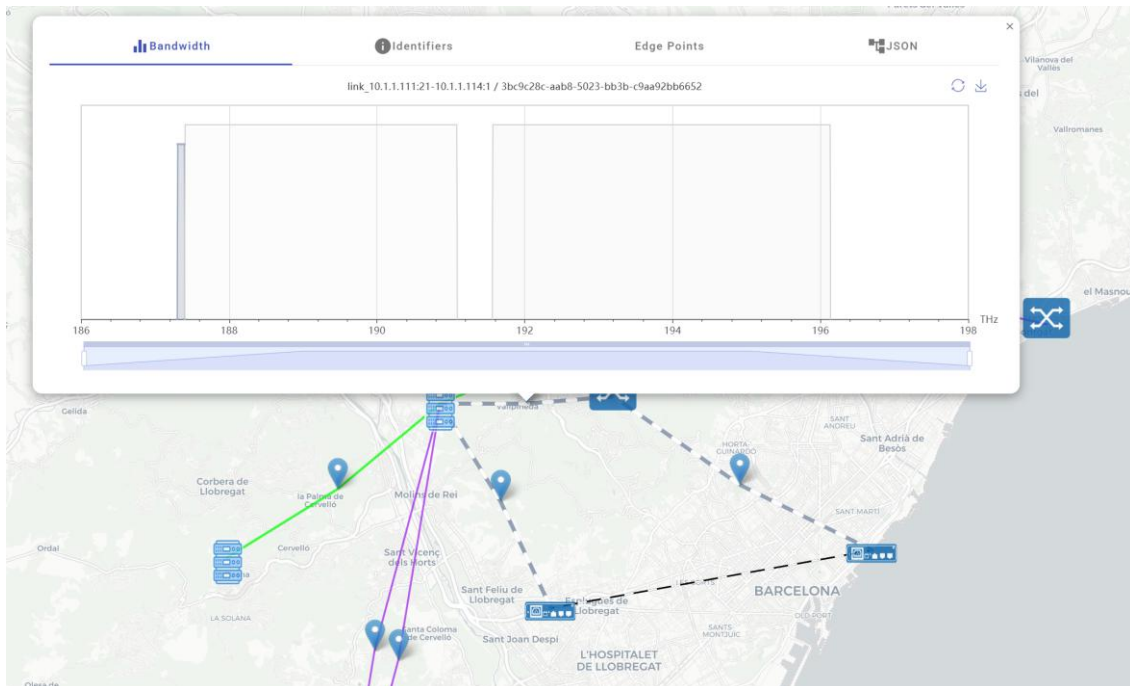


Figure 3.3-9: Visualization of link bandwidth allocation in the optical network within the L-band.

The L-band component is received at *S-BVT-Rx*, validating the multiband capability of the system. Performance metrics such as SNR per subcarrier and BER are collected and visualized (see Figure 3.3-10). This validates the ability to configure and monitor multiband flows dynamically and demonstrates the second selected use case.

```

C:\Windows\system32\cmd.exe
3895246, 45.14028639525508, 45.16918540215566, 32.90682775148972, 37.11499907385
99344, 40.4746220214097, 32.15412181336472, 40.04561532653369, 32.18093258110318
20.51045908441258, 36.61835129843874, 33.51256863434823, 23.97776877627196, 3
5.51864392977425, 35.57543241921788, 39.378822327626466, 26.61621137319002, 27.1
0149887305123, 17.937686198778636, 19.74211716232784, 24.559384415016495, 10.234
4234808990177, 15.515626930146681, 7.263074600207676, 20.042668611763303, 12.457
724711704355, 13.2140894361810929, 18.5113323172139, 16.95139718971036, 15.130833
547353873, 17.061048871179896, 18.00328682772289, 14.503451909531549, 15.2500596
79051319, 9.540335157704002, 9.934092002648330, 11.374430069622937, 10.661732208
61236, 9.446518920277712, 7.621441176264485, 8.164662785339712, 1.78083918456899
46, 4.2497426918144245, 2.072788236601956, 4.964555142446471, 1.223937574491817
6, 0.3265742561408947, 3.7475987253380136, 1.8140646987905251, 0.71233709581434,
2.210115725593549, 0.09212394354919122, 2.866274266284088, 0.1274159367195227, 1
.613747771292146, 0.02024565691799639, 1.0380491513852764, 1.1077197103050068
1.070750723901306, 3.194322146723145, 0.613934397273066, 0.5152455976407447, 0
.19294978143075195, 6.741715843597409, 1.4469803125402092, 4.337190286828503, 1.3
99755797253936, 5.4203891350795015, 0.046128443106095, 2.1236909115817566, 4.17
4303692492528, 1.3543299570440977, 2.655524000277395, 9.038418240578487, 12.3943
28970266974, 5.96493936506364, 11.946237601916891, 8.797820361570791, 10.056478
402912777, 16.537994984818007, 10.328778945381366, 8.55275923635703, 11.0426031
930878, 15.58501033534406, 15.417672254234366, 24.48320261230008, 20.58229213230
1383, 15.239481397657256, 17.614606018853898, 22.16280235623642, 28.369911374069
06, 8.792373382976756, 20.091332071090208, 23.0232585063512, 26.07983270483405,
29.891727914345689, 13.2955449428049, 31.02517656621194, 16.151698805833703, 33
.62812898150835, 29.554209039044014, 38.24276520365455, 25.087150490305717, 31.91
099751539825, 34.253914597192896, 28.176548993548863, 34.10732325000907, 20.8684
51272663314, 20.613195829199036, 41.31023004222971, 37.71413156905622, 17.767954
78955512, 30.68086597765875, 41.50624934022727, 49.18996746469402, 43.0433832
4809615, 53.01998946761314, 43.560977151931026, 41.26621051249909, 31.002229796
63392, 57.170744470094014, 42.116823740581806, 55.14132412620225, 57.43237283473
244, 53.150873611981645, 52.216765900012255, 51.40007988639828, 35.8479990651193
9, 75.9418069045686, 49.39394897086566, 61.20894930053458, 42.30751566747105,
36.705624800256935, 19.150454796786870, 68.19190789920548, 51.205208291053414, 3
0.33996342688351, 35.815411087656145, 43.49883189273226, 52.49582109399316, 40.2
792822556570, 51.480637418168904, 36.5069601309399, 29.972035107894964, 17.4080
06836745497, 39.40762655074676, 34.1484334252741, 36.20470020049034, 38.6972505
5884295, 35.68581968099415, 19.036619580226084, 24.02521422272085, 38.011635452
61837, 41.37446642187228, 39.1130205439323, 23.066456309799555, 34.8114388854461
4, 27.642522092999034, 35.15278314081183, 44.63446009175752, 19.089072467487306,
23.81783423511141, 30.13024022357144, 20.55601316050347, 32.453742116557105, 25
.9721470214680856, 24.15542202593542, 23.25708086846378, 22.640711406600517, 27.3
938002600747658, 29.798175546546872, 28.509822244265816, 20.54703415710966, 9.325
857482848802, 24.742031670511924, 16.33428055159501, 13.942572123929894, 23.2639
04084764228, 24.90772739104933, 22.161530574963912, 18.57279357471206, 11.74659
061547310, 22.21056450664814, 26.36580222639073, 10.739425519961706, 17.0351407
1550448, 21.231007611399857, 23.929226494500433, 19.245355250412242, 11.52669778
0480617, 12.54394209669525, 9.328580243428823, 14.300245748347859, 16.9625842098
22435, 6.7545085631446095, 18.422031056419453, 12.054306595406539, 7.83954778602
9876, 15.00264252657301, 9.862120072806134, 8.119895752638444, 4.181304697607419
7, 7.565449613709431, 6.233067893800265, 9.615104142559655, 8.344602767910917, 3
.209217490134396, 6.0026407716160261, 0.0744527180989583291
INFO:werkzeug:10.1.1.162 -- [06/Nov/2025 13:24:56] "POST /api/osc HTTP/1.1"
10m 200 -

```

Figure 3.3-10: Performance metrics for the received signal, including SNR and BER values.

Dissemination Level	PU
---------------------	----

## 4 CONCLUSION

---

This chapter presents the conclusions of this deliverable by summarizing all the work reported in the document. To be precise, the summary of each of the two demos of the SEASON project related to MBoSDM is presented here.

First demo reported in this deliverable is the final demo of SEASON that was performed at HHI's premises. The demo showcased self-healing in an IPoWDM network with dual protection, i.e., handling two failures that occur simultaneously in the network. The use case with dual protection was considered in order to demonstrate the capabilities of MBoSDM node prototype developed at HHI. The integration of different control and data plane components developed in the SEASON project was successfully demonstrated. The results showed that the proposed control plane was able to take remedial actions in the network to keep the video streaming traffic ongoing even when two simultaneous failures occur in the network. The timing measurements were also done in the demo which showed that the service provisioning time was less than 1.5 minutes, and the healing time was less than 1 minute. These values are well ahead of the target specified in KPI 2.3 of the project, requiring sub-3-minute network connectivity setup across multiple network domains.

Second demo reported in this deliverable was performed at CTTC's premises, and it also showcased the benefits of MBoSDM technology by using another MBoSDM node prototype that was developed at CTTC. Contrary to HHI's node prototype, this one used active components like WSSs. Since the purpose of this demo was to show only the capabilities of MBoSDM node, a small-scale control plane (with respect to HHI final demo) was utilized. Two different use cases were demonstrated for considering both an SDM and a multiband flexi-grid service. The experimental validation results demonstrated a successful integration of different components.

## GLOSSARY

---

Acronym	Description
<b>API</b>	Application Programming Interface
<b>BER</b>	Bit Error Rate
<b>BVT</b>	Band Variable Transceiver
<b>BW</b>	Bandwidth
<b>CLI</b>	Command Line Interface
<b>CMIS</b>	Common Management Interface Specification
<b>CPU</b>	Central Processing Unit
<b>CRUD</b>	Create/Read/Update/Delete
<b>DD</b>	Direct Detection
<b>FEC</b>	Forward Error Correction
<b>GUI</b>	Graphical User Interface
<b>HLS</b>	HTTP Live Streaming
<b>HPA</b>	Horizontal Pod Autoscaling
<b>IETF</b>	Internet Engineering Task Force
<b>IPoWDM</b>	IP over Wavelength Division Multiplexing
<b>K8s</b>	Kubernetes
<b>MB</b>	Multi-Band
<b>MBoSDM</b>	Multi-Band over Spatial Division Multiplexing
<b>MNC</b>	Mosaic Network Controller
<b>NBI</b>	North Bound Interface
<b>NETCONF</b>	Network Configuration Protocol
<b>NetDevOps</b>	Network and Development Operations
<b>NSC</b>	Network Slice Controller
<b>OLS</b>	Open Line System
<b>ONOS</b>	Open Network Operating System
<b>OSA</b>	Optical Spectrum Analyzer
<b>OSNR</b>	Optical Signal-to-Noise Ratio
<b>QoS</b>	Quality of Service
<b>QSFP</b>	Quad Small Form-factor Pluggable
<b>REST</b>	Representational State Transfer
<b>RESTCONF</b>	Representational State Transfer Configuration Protocol
<b>RESTful API</b>	Representational State Transfer Application Programming Interface
<b>ROADM</b>	Reconfigurable Optical Add-Drop Multiplexer
<b>RTMP</b>	Real-Time Messaging Protocol

<b>RTT</b>	Round Trip Time
<b>Rx</b>	Receiver
<b>SBI</b>	South Bound Interface
<b>S-BVT</b>	Sliceable Bandwidth Variable Transceiver
<b>SDM</b>	Spatial Division Multiplexing
<b>SDM-MB</b>	Spatial Division Multiplexing - Multi-Band
<b>SDN</b>	Software-Defined Networking
<b>SIP</b>	Service Interface Point
<b>SNMP</b>	Simple Network Management Protocol
<b>SNR</b>	Signal-to-Noise Ratio
<b>SONiC</b>	Software for Open Networking in the Cloud
<b>TAPI</b>	Transport API
<b>TCP</b>	Transmission Control Protocol
<b>TFS</b>	TeraFlow SDN
<b>Tx</b>	Transmitter
<b>UE</b>	User Equipment
<b>UI</b>	User Interface
<b>VPN</b>	Virtual Private Network
<b>WDM</b>	Wavelength Division Multiplexing
<b>WSS</b>	Wavelength Selective Switch
<b>YANG</b>	Yet Another Next Generation

---

## REFERENCES

---

- [BAR22] S. Barguil and L. Munoz. «RFC 9291: A YANG Network Data Model for Layer 2 VPNs.» (2022). [Online]. Available in: <https://datatracker.ietf.org/doc/rfc9291/>
- [BAR23] S. Barguil, O. G. de Dios, M. Boucadair, L. Munoz and A. Aguado «RFC 9182: A YANG Network Data Model for Layer 3 VPNs.» (2022). 2024. [Online]. Available in: <https://www.rfc-editor.org/rfc/rfc9182.html>
- [Con25] IETF. (2025). draft-ietf-teas-ns-controller-models-06: Network Slice Controller Models. Internet Engineering Task Force. <https://datatracker.ietf.org/doc/draft-ietf-teas-ns-controller-models/>
- [OpenConfig] OpenConfig, “Openconfig web site. [online] available: <http://www.openconfig.net/>,” (2018).
-