This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016608.

BEYOND 5G – OPTICAL NETWORK CONTINUUM
(H2020 – Grant Agreement Nº 101016663)

Deliverable D4.4

Software release of different components

Editor	C. Matrakidis (OLC-E)
Contributors	TID, UC3M, TIM, CTTC, INF-P, NBLF, CNIT, CNR, UPC, OLC-E, ELIG, PLF
Version	1.0
Date	July 31, 2024
Distribution	PUBLIC (PU)

DISCLAIMER

This document contains information, which is proprietary to the B5G-OPEN (Beyond 5G – OPtical nEtwork coNtinuum) consortium members that is subject to the rights and obligations and to the terms and conditions applicable to the Grant Agreement number 101016663. The action of the B5G-OPEN 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 B5G-OPEN 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 B5G-OPEN consortium members as a whole, nor a certain B5G-OPEN consortium member warrant that the information contained in this document is suitable for use, nor that the use of the information is accurate or free from risk, and accepts no liability for loss or damage suffered by any person using this information.

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

REVISION HISTORY

Revision	*Date*	*Responsible*	*Comment*
0.1	*October 10, 2024*	*Chris Matrakidis*	*First Release*
0.2	*October 28, 2024*	*Lutz Rapp*	*Quality Check*
1.0	*November 07, 2024*	*WPL*	*Approval*

LIST OF AUTHORS

Partner ACRONYM	Partner FULL NAME	Name & Surname
TID	Telefonica I+D	Oscar González de Dios
UC3M	Universidad Carlos III de Madrid	José Alberto Hernández, Alfonso Sánchez-Macián, Gonzalo Martínez
TIM	Telecom Italia	Roberto Morro
CTTC	Centre Tecnològic de Telecommunicacions de Catalunya	Ramon Casellas, Laia Nadal, Michela Svaluto Moreolo, Fco. Javier Vilchez, Ricardo Martinez
CNIT	CNIT	Filippo Cugini
CNR	CNR	Alessio Giorgetti
ELIG	E-lighthouse Network Solution	Pablo Pavón Mariño, Enrique Fernández Sánchez, Francisco Javier Moreno Muro
NBL	Nokia Bell Labs	Fabien Boitier, Patricia Layec, Petros Ramantanis, Isaia Andrenacci
UPC	Universitat Politecnica de Catalunya	Luis Velasco, Marc Ruiz, Jaume Comellas, Salvatore Spadaro, Davide Careglio, Josep Prat, Morteza Ahmadian
Adtran	Adtran Networks SE	Achim Autenrieth, Nikhil Dsilva, Vignesh Karunakaran, Jasper Müller
OLC-E	OpenLightComm Europe s.r.o.	Alexandros Stavdas, Evangelos Kosmatos, Chris Matrakidis
PLF	pureLiFi Ltd	Rui Bian
INF-P	Infinera Unipessoal Lda	João Pedro António Eira

GLOSSARY

Acronyms	Description	Acronyms	Description
5G	Fifth Generation	OMS	Optical Multiplex Section
6G	Sixth Generation	ONF	Open Networking Foundation
AI	Artificial Intelligence	ONOS	Open Network Operating System
AP	Access Point	ONU	Optical Network Unit
API	Application Programming Interface	ONP	Open Network Planner
ASIC	Application Specific Integrated Circuit	OPCE	Optical Path Computation Element
B5G	Beyond 5G	OptC	Optical Controller
B5G-ONP	Beyond 5G - Open Network Planner	OSA	Optical Spectrum Analysers
BBF	Broadband Forum	OSNIR	Optical Signal-To-Noise plus Interference Ratio
BGP	Border Gateway Protocol	OSNR	Optical Signal-To-Noise Ratio
CD	Chromatic Dispersion	OSPF	Open Shortest Path First
CLI	Command Line Interface	OSS	Operation Support Systems
CMIS	Content Management Interoperability Services	OTS	Optical Transport Section
DB	Database	OTSi	Optical Tributary Signal
DNN	Deep Neural Network	P2MP	Point-To-MultiPoint
DSP	Digital Signal Processing	PCE	Path Computational Engine
DSR	Digital Signal Rate	PckC	Packet Controller
E2E	End-To-End	PDL	Polarisation Dependent Loss
FEC	Forward Error Correction	PDF	Probability Density Function
gNMI	gPRC Network Management Interface	PL	Physical Layer
gRPC	gRPC Remote Call Procedure	PLA	Physical Layer impairment Aware
GUI	Graphical User Interface	PLI	Physical Layer impairment
HrC	Hierarchical Controller	PMD	Polarisation Mode Dispersion
HTTP	Hypertext Transfer Protocol	PON	Passive Optical Network
IBN	Intent-Based Networking	PSD	Power Spectral Density
IDS	International Data Spaces	QoE	Quality of user Experience
IETF	Internet Engineering Task Force	QoS	Quality of Service
IGP	Interior Gateway Protocol	QoT	Quality of Transmission
IP	Internet Protocol	REST	Representational State Transfer
IPoWDM	IP over WDM	RMSA	Routing, Modulation and Spectrum Assignment
IS-IS	Intermediate System to Intermediate System	ROADM	Reconfigurable Optical Add/Drop Multiplexer
IT	Information Technology	RPC	Remote Procedure Call
ITU	International Telecommunication Union	RSA	Routing and Spectrum Assignment
JSON	JavaScript Object Notation	SAI	Switch Abstraction Interface
K8s	Kubernetes	SBI	Southbound Interface

KPI	*Key Performance Indicator*	**SDK**	*Software Development Kit*
LiFi	*Light Fidelity*	**SDN**	*Software Defined Networking*
LKM	*Loadable Kernel Modules*	**SIP**	*Service Interface Point*
LLDP	*Link Layer Discovery Protocol*	**SMA**	*Spectral and Modulation Assignment*
LSP	*Label Switched Path*	**SNMP**	*Simple Network Management Protocol*
MB	*Multi-Band*	**SNR**	*Signal-To-Noise Ratio*
MDA	*Monitoring and Data Analytics*	**SONiC**	*Software for Open Networking in the Cloud*
ML	*Machine Learning*	**SSH**	*Secure Shell*
		SSID	*Service Set Identifier*
MPLS	*Multiprotocol Label Switching*	**TAPI**	*Transport API*
MQTT	*Message Queuing Telemetry Transport*	**TDM**	*Time Division Multiplexing*
NBI	*North Bound Interface*	**TLS**	*Transport Layer Security*
NF	*Noise Figure*	**TP**	*Transponders*
NOS	*Node Operating System*	**Tx**	*Transmission*
O/E/O	*Optical to Electrical to Optical*	**WDM**	*Wavelength Division Multiplexing*
OA	*Optical Amplifier*	**WiFi**	*Wireless Fidelity*
OCH	*Optical Channel*	**WP**	*Work Package*
ODN	*Optical Distribution Network*	**WRN**	*Wavelength Routed Network*
OIF	*Optical Internetworking Forum*	**XGS**	*Symmetric 10G-PON*
OLS	*Open Line system*	**XML**	*Extensible Markup Language*
OLT	*Optical Line Terminal*	**XR Optics**	*Infinera Technology for point to muti-point optics*
OMB	*Optical Multi-Band*	**YANG**	*Yet Another Next Generation*

EXECUTIVE SUMMARY

This deliverable reports the high-level description of the software and sub-components implemented for B5G-OPEN. This is presented in two parts, a summary of the key elements of the architecture and tables for the sub-components, showing their status, availability and other information.

Specifically, the document summarizes the **Infrastructure Control and Service Management platform** architecture implemented in the B5G-OPEN Control, Orchestration and Telemetry System (referred to as the control plane, for short). The key aspects of the architecture are:

- Multi-Band operation: Provision services using available bands out of the O-, E-, S-, C-, L-band in optical fibres.
- Optical continuum: Allow optical slicing based on service requirements and crossing network segments (i.e. access, metro, core, etc.)
- Integrated access: Operate and control service regardless of the access technology (Mobile, Fixed, WiFi, LiFi)
- E2E network orchestration: Operate service and network operations from the Access Point to the Cloud node, which may include monitoring and AI/ML
- Autonomous operation: Based on Intent-based and zero-touch networking paradigms, autonomous operation is built using closed-control loops at various levels, from device to network.

The major parts of the architecture include: service orchestration and planning, packet optical-integration systems, telemetry and intent-based networking. As presented in D4.1, the main control plane innovations are:

- **[multiband control**] Control of optical multi-band network, this means being able to exploit the multiband capabilities of optical devices such as transmission or switching.
- [**transparent multi-domain**, *domain-less*] The ability to setup connections in a transparent manner, across multiple domains and network segments.
- [**Packet/optical integration**] Moving forward from current hierarchical architectures for the SDN control plane of the control plane that consider the IP/MPLS layer largely decoupled from the control plane of the optical layer
- [**physical layer impairments, PLI**] Accounting for PLI is critical to efficiently plan and operate optical networks and high data rates, with increasing non-linear effects.
- [**telemetry**] The scope of the SDN extends to optical monitoring and telemetry, a key enabler for advanced functions such as autonomous/autonomic networking via hierarchical and coordinated closed loops.
- [**external planning tools**] Planning tools, including QoT estimators or path computation and validation systems need efficient access (in terms of retrieval, storage and processing) to collected and managed data.
- [**network automation**] Aspects related to automation, zero touch networking and Intent Based Networking (IBN) are developed in the areas of service deployment, network planning and overall network operation.

The interfaces for such a modular architecture, which must rely on standard and open interface definitions between the control plane functions and towards the devices. Those were presented in D4.2 together with components required for the service orchestration and infrastructure control system. Similarly, the generic telemetry platform enabling straightforward adaptations of devices or systems as data sources was defined.

The presentation of this work is concluded in the second part of this document, where the final status of each component is described. This is summarized in the following table that provides the list of components that have been implemented and the key performance indicators for each one.

Component	Partner	Purpose
B5G-ONP	ELIG	The B5G-OPEN Optical Network Planner (B5G-ONP) component orchestrates both IT and network resources. Within the B5G-OPEN project, the B5G-ONP serves as the hub and provides design, optimization, and planning tools to deploying, managing, and configuring services and resources. The KPIs measured are: 1. Topology Discovery Time, the time elapsed from the moment a network operator sends the finding command until all layer of the topology is imported into B5G-ONP, with target values from less than 5 seconds up to 30 seconds. The average values measured were: - TAPI Optical Network Orchestrator: 300 ms - IP SDN Controller: 4 seconds - Kubernetes Cluster: 2 seconds 2. Connectivity service provisioning latency, the time to provision connectivity service starting when B5G-ONP receives a request for it and ending when the service is properly established, with target values of less than 10 minutes with hardware and less than 1 minute with emulation. The following mean baseline values were recorded: - DSR provisioning: 237 ms - OLS path provisioning: 110 ms - IP BGP adjacency provisioning: 4 seconds - Reconfiguring OLS path provisioning: 200-300 ms - Kubernetes Deployment: 150 ms - Kubernetes Service: 100 ms 3. Optical Path computation element latency, the time it takes to perform a path and is directly proportional to the number of network elements involved in the path computation and the traffic load on the network. Values measured in the project experiments: - PCE: average of 200 ms - Multiband-PCE: average of 600 ms, depending directly on the time consumed by the MB-PCE when requesting the optical context.
TAPI-enabled Optical Network Orchestrator with	CTTC	The TAPI-enabled Optical Network Orchestrator is responsible for: i) providing a uniform, open and standard view and interface to the higher levels and components; ii) Composing a complete Context to be consumed by B5G-OPEN network planner and additional consumers combining information retrieved from subsystems and sub-controllers; iii) Enabling a single entry point for provisioning

externalized Path Computation		DSR and Photonic Media services, including externalized path computation and iv) providing an event telemetry data source that reports events that happen asynchronously in the network. The KPIs measured are: 1. Service Provisioning Overhead – In terms of messages, message size, encoding, etc. This includes a characterization of the protocol overhead (e.g., HTTP, RESTCONF, etc). Values measured in an experiment: - Discovery context: 931 ms - Single-domain provision: 309 ms - Multi-domain provision: 515 ms 2. Path Computation Latency – measured as the time it takes to perform a path computation with a dedicated PCE. This is to be evaluated. In different experiments the following values were obtained: - PCE latency: 0.5 - 0.6 seconds - MB-PCE: PCE latency :1.8 - 3.2 seconds
Multi-Band Path Computation Engine (MB-PCE)	OLC-E	The Multi-Band Path Computation Engine (MB-PCE) is based on a multi-band routing engine which ensures that: i) routing is implemented by means of an efficient spectrum and modulation-format assignment; and ii) the impact of physical layer effects over the selected optical paths is estimated and the results are benchmarked against QoT target values (BER, OSNIR, OSNR, etc). The KPIs measured are: 1. Path Computation Latency, is the time it takes to MB-PCE to compute the path of a new service request, with a target value of < 40s. - Several scenarios were executed. The MB-PCE Path Computation Latency was measured to be in the range between 1.8 – 3.2 seconds 2. TAPI Topology retrieval and parsing Latency, the time it takes for MB-PCE to retrieve and parse the optical network topology context, with a target value of < 20s. - Several scenarios were executed. The MB-PCE Topology retrieval and parsing Latency was measured to be in the range between 0.9 – 2.5 seconds
ONOS Optical Controller	CNIT	The optical controller is based on the ONOS open-source project [ONOS] that, besides the control of optical devices, also provides a suitable environment for the control of packet devices (e.g., based on OpenFlow or P4Runtime protocols). The main roles of the optical controller in the B5G-OPEN project are: (i) retrieve device descriptions from data plane and abstract them toward the upper control layers; (ii) receive the service configuration requests by the upper control layers and translate such requests in a set of configuration messages to be forwarded to each involved device.

		Set of component KPIs that have be measured during experimental demonstrations: 1. Time required for network discovery: - Measured in the order of tens of seconds, always lower than one minute. 2. Physical activation delay for an optical intent: - Measured in the order of one-two minutes mostly depending on the utilized transceivers. 3. Time elapsed in the controller: - Measured in the order of one second. 4. Time elapsed for configuration of devices: - Devices typically confirm the reception of a configuration message in the order of few hundreds of milliseconds.
Access Controller / PON Controller	OLC-E	The Access Controller is responsible for: a) monitoring the PON network and receiving any requests for PON reconfiguration; b) translating these requests into high level traffic requests that will be reported to the B5G-ONP App; c) executing the appropriate actions in the PON Controller in order to support the new requests. In addition, the Access Controller will communicate with the LiFi Controller for retrieving any connection/traffic requests The KPIs measured are: 1. PON Reconfiguration Latency, the time it takes for the actual reconfiguration of the PON network, with a target value of < 20s. - The latency for retrieving the configuration and status information from different elements in the PON network was measured between 1.2 to 1.8 seconds - In addition, the PON Reconfiguration Latency (SLA update and application) was measured between 450 - 600 ms 2. Access Controller Latencies, the time it takes to Access Controller to execute different functionalities. Here are the values reported in the Berlin Demo (WP5): - Authentication: 234 - 790 ms - PON SLA creation: 187 - 781 ms - ONU SLA configuration: 316 - 529 ms - Logout: 71 - 514 ms - Send access traffic descriptor to B5G-ONP app: 51 - 79 ms In addition, the response from B5G-ONP app was measured between 53 ms and 7.26 seconds (for different B5G-ONP app configurations).
LiFi Controller	PLF	The LiFi controller serves as the central component responsible for managing LiFi APs in the network. It is strategically positioned between the PON controller and the LiFi AP. This specific positioning ensures seamless communication and enhanced coordination between the optical network layer and the wireless LiFi communication layer

		The KPIs measured are: 1. Device Management Accuracy, assessing how accurately the controller works and manages device states. - No failure was observed in normal working conditions. 2. Scalability, the controller's ability to scale by managing an increasing number of LiFi devices without performance degradation. - It has been tested for up to three devices in lab conditions. 3. Latency, time for the controller responds to user's request. - Measured between 10 and 70 ms (excluding any NETCONF sessions accessing the LiFi agent).
LiFi agent	PLF	The LiFi Agent acts as a central hub in the architecture of the LiFi AP. With the advancement of NETCONF capabilities, the agent provides a seamless way for the AP to interact with other components, offering a structured interface for configurations and management The KPIs measured are: 1. Device Recognition Success Rate, how often the agent successfully recognizes and interfaces with devices. - No failure was observed on recognising the LiFi APs. 2. Configuration Accuracy, how accurately the agent applies the configuration changes to the devices. - No failure observed. 3. Stability, how reliable and stable the agent is with multiple LiFi AP devices. - Has been tested with up to three LiFi APs in lab conditions. 4. Latency for accessing the LiFi AP - Measured between 5 ms and 20 ms, - The latency for having a NETCONF session is typically 30 and 71 ms. - With the light communication path partially blocked, 150 ms has been observed.
OpenROADM agent	TIM	The OpenROADM agent is an implementation of a NETCONF server controlling optical network elements using OpenROADM device models The KPIs measured are: 1. Start-up delay, for different datastore sizes (4 to 9 degrees). - Measured between 437 and 477 ms. 2. Discovery Latency, the time required by ONOS SDN controller to discover the ROADM and its port capabilities. - Measured between 30.2 and 43.7 ms 3. Connection Latency, the time required for the configuration change to create a cross-connection between two roadm degrees. - Measured between 172 ms and 2.5 seconds.

OpenConfig agent	CTTC CNIT	OpenConfig agent is an implementation of an SDN agent using NETCONF/YANG with the OpenConfig data models. It implements a subset of the data models, namely the OpenConfig platform and optical transport as well as some extensions devised in the context of B5G-OPEN to report details about the transceiver operational mode The KPIs measured are: 1. Instantiation *delay and footprint*: when the agent is running as a containerized application, characterize aspects related to instantiation of the agent, as well as aspects related to memory usage. - Measured from 17 ms to 1.230 seconds. The variability is due to retrieving operational data from the devices. 2. Discovery *latency*: measure the time and the control plane overhead (in terms of bytes, and throughput) it takes for an SDN controller to discover the components of the transceiver upon a NETCONF get operation. - The discovery latency was measured as 475 ms 3. Operational *Mode characterization*: measure the time and the control plane overhead (in terms of bytes, and throughput) it takes for an SDN controller to discover the details of a given operational mode - The measured value was 300ms 4. Transaction *delay*: the time it takes to send a configuration change, and this is reflected in the datastore. The focus shall be to change an Optical channel frequency, power and operational model. This KPI will be evaluated with and without hardware - A total setup time of around 300s is needed to perform all the required OpenConfig operations to set up the connection.
SONiC based packet optical node	CNIT	The Software for Open Networking in the Cloud, i.e., SONiC, [SONIC] is considered as the Network Operating System (NOS) to be deployed on packet-optical IPoWDM nodes operated in metro/aggregation networks. Within the B5G-OPEN project SONiC has been extended with several components provided in the form of docker containers. The KPIs measured are: 1. Time required at the data layer for enforcing a modification of the central frequency of the optical coherent transceivers, has measured using ZR and ZR+ transceivers: - Control plane time from 9.38 to 10.37 seconds - ZR transceiver output time from 67,34 to 71.06 seconds - ZR+ transceiver output time from 13.2 to 19.1 seconds 2. The spectrum width required to preserve the quality of transmission of a 400 Gbps channel with 16-QAM modulation format: - 68 GHz required for no QoT degradation - 60 GHz with QoT degradation.

AI/ML models for PSD and Power Management	NOKIA	Machine learning application towards augmented optical networks and is called "Automatic power correction" The KPIs measured are: 1. Gain/loss of performance: The difference between the signal-to-noise ratio before and after using this AI/ML component. - The SNR gain after correction is up to 0.5 dB for 1dB power correction and 1dB for 2.5dB power correction. - The maximum loss of performance was 0.15dB 2. Scalability in terms of number of input points: The number of points required for the optical PSD to get a given accuracy - For 20 PSD points, the error standard deviation is 1.67dB - For 100 PSD points, the error standard deviation is 1dB - For 315 PSD points, the error standard deviation to 0.41 dB.
Telemetry System	UPC	B5G/OPEN distributed telemetry system integrates measurements and event data collection and supports intelligent data aggregation nearby data collection, so agents receive and analyse measurements before sending to a centralized manager. The KPIs measured are: 1. Optical Constellation Data Compression with Autoencoders, a compression technique applied to optical constellation measurements. - Achieved 625:1 compression with reconstruction error < 2% 2. Spectrum Dimensionality Reduction with Feature Extraction, a technique applied to spectrum measurements. - The compression rate achieved was 7.5:1
FlexTelemetry Agent	Adtran	The FlexTelemetry Agent periodically requests and collects performance measurements from optical transport network devices. It utilizes NETCONF and supports both open (OpenConfig) and proprietary data models to ensure comprehensive data collection. It features a modular plugin system that provides a Northbound Interface (NBI) capable of delivering stable telemetry streams to various mediums. This modular approach allows Flex-Telemetry to seamlessly integrate with diverse data storage and processing systems, facilitating efficient, scalable access to performance data across different platforms.
Mesarthim – Failure management Using a SNR Digital Twin	UPC	MESARTHIM compares the QoT measured in the transponders with the one estimated using a QoT tool. Deviations can be explained by changes in the value of input parameters of the QoT model representing the optical devices, like noise figure in optical amplifiers and reduced Optical Signal to Noise Ratio in the Wavelength Selective Switches. By applying reverse engineering, MESARTHIM estimates the value of those modelling parameters as a function of the observed QoT of the lightpaths The KPIs measured are:

		1. Estimate the most likely modelling configuration, with relative average error of the modelling parameters estimation < 8%. - At each step, the module was able to explain the increment in the SNR of the lightpath by a reduction in the bandwidth in the related WSS or the NF of the OA. $R^2 > 0.986$ 2. Anticipation of soft failures, higher than 15% through the estimation of modelling parameters w.r.t. SNR analysis. - P-max degradation anticipated 15% - NF degradation anticipated 45% - WSS degradation anticipation 27% 3. Severity estimation, anticipating > 40% - P-max estimation 42.8%, - NF estimation 62.8% - WSS estimation 49.6%
Ocata - Digital Twin for the Optical Time Domain	UPC	OCATA is a deep learning-based digital twin for the optical time domain that is based on the concatenation of deep neural networks (DNN) modelling optical links and nodes, which facilitates representing lightpaths. The DNNs model linear and nonlinear noise, as well as optical filtering The KPIs measured are: 1. Lightpath modelling error, comparing the optical constellations generated with OCATA with real ones for the same lightpath. - Distribution mean error under 2% independently of the link length. - Distribution standard deviation error under 15% for longer paths. - The reconstruction of the features of the non-selected constellation points showed an accuracy of 97%. - Average error for lightpath estimation < 5% for lightpaths over 500Km - Average error for estimation of number of hops of the lightpath < 5%. 2. Reduction of running time, with a target value of > 2 orders of magnitude with respect to Split-Step Fourier Method (SSFM) simulation - The running time is over 3 orders of magnitude faster than SSFM. - The running time depends only on the number of hops and not on the distance (as in SSFM).

TABLE OF CONTENTS

TABLE OF CONTENTS 14

1 Introduction 1

2 High-level description of the control plane 4

2.1 B5G-OPEN control plane services 4

2.1.1 Point to Point Optical Connectivity 4

2.1.2 Point to Point DSR Connectivity 4

2.1.3 Point to Multipoint connectivity 4

2.1.4 IP link provisioning 5

2.1.5 Packet/IP Connectivity 5

2.1.6 P2MP Access Connectivity 6

2.1.7 B5G-OPEN Network Slice 6

2.1.8 Other services 6

2.1.9 Telemetry and Intent Based Networking 6

2.2 Optical Network control 7

2.2.1 TAPI-enabled Optical Network Orchestrator (TAPI NOrch) 7

2.2.2 Optical Controller 7

2.2.3 OLS Controller 8

2.2.4 Optical Path Computation Element 9

2.2.5 Multi-domain scenarios 9

2.3 Access Networks Control 10

2.3.1 The Framework of TDM-PON Configuration and Control 10

2.3.2 LIFI Control 11

2.4 Orchestration 12

2.4.1 IT and network resources orchestration 12

2.4.2 B5G-ONP modules 12

2.5 Packet/optical integration 13

2.5.2 Sonic generic architecture 14

2.5.3 Pluggable management and control 14

2.5.4 P2MP Pluggable Management and Control 15

2.6 Telemetry platform 16

2.6.1 B5G-OPEN Telemetry Architecture 16

2.6.2 OLS Node Agent and Telemetry Adaptor 18

2.7 Autonomous Networking and Quality Assurance 19

2.7.1 Autonomous Networking 19

2.7.2 Single-Domain and Multi-Domain Quality Assurance 20

3 B5G-OPEN Software Components 22

3.1 B5G-ONP (ELIG) 22

3.2 TAPI-enabled Optical Network Orchestrator with externalized Path Computation (CTTC) 26

3.3 Path Computation Elements – MB-PCE – (OLC-E) 32

3.4 Optical Controller (CNIT) 35

3.5 Access Controller / PON Controller (OLC-E) 38

3.6 LiFi controller (PLF) 41

3.7 LiFi agent (PLF) 43

3.8 OpenROADM Agent (TIM) 45

3.9 OpenConfig Agent (CTTC and CNIT) 48

3.10 SONIC-based Packet Optical Node (CNIT) 54

3.11 AI/ML models for PSD and Power Management (NOKIA) 58

3.12 Telemetry System (UPC) 61

3.13 FlexTelemetry Agent (Adtran) 65

3.14 Mesarthim – Failure management Using a SNR Digital Twin (UPC) 66

3.15 Ocata - Digital Twin for the Optical Time Domain (UPC) 68

References 72

List of figures

Figure 2-1 Macroscopic B5G-OPEN architecture and Service instantiation interfaces. 4
Figure 2-2 IP link provisioning between the whiteboxes. 5
Figure 2-3 multiple IP link provisioning between the whiteboxes using P2MP XR....................... 5
Figure 2-4 B5G-OPEN Control and Orchestration architecture 6
Figure 2-5 B5G-OPEN Intent Based Applications (IBN) and Knowledge-Sharing. 7
Figure 2-6 ADVA OLS Controller Northbound Interfaces 9
Figure 2-7 Control plane architecture for the multi-OLS scenario, showing a back to back add/drop-add/drop configuration. 10
Figure 2-8 B5G-OPEN Control of PON through the PON Controller 11
Figure 2-9: LiFi-XC AP 11
Figure 2-10: Initial assumption for LiFi control 12
Figure 2-11 Coordination of Kubernetes cluster from B5G-ONP 13
Figure 2-12 Traditional SDN architecture for transponder-based optical networks. 14
Figure 2-13 Pluggable management and control architecture 15
Figure 2-14 P2MP Control integration in B5G-OPEN 16
Figure 2-15 Overall network architecture 17
Figure 2-16 Measurements telemetry architecture and workflow 17
Figure 2-17 Events telemetry architecture and workflow 18
Figure 2-18 Intra domain Control loop architecture 20
Figure 2-19 Intent-based networking in the intra-domain 21
Figure 2-20 Intent-based networking in multi-domain scenarios.. 21

1 Introduction

This deliverable reports the high-level description of the software and sub-components implemented for B5G-OPEN. This is presented in two parts, a summary of the key elements of the architecture and tables for the sub-components, showing their status, availability and other information.

More formally, the document summarizes the **Infrastructure Control and Service Management platform** architecture implemented in the B5G-OPEN Control, Orchestration and Telemetry System (referred to as the control plane, for short). The key aspects of the architecture are:

- Multi-Band operation: Provision services using available bands out of the O-, E-, S-, C-, L-band in optical fibres.
- Optical continuum: Allow optical slicing based on service requirements and crossing network segments (i.e. access, metro, core, etc.)
- Integrated access: Operate and control service regardless of the access technology (Mobile, Fixed, WiFi, LiFi)
- E2E network orchestration: Operate service and network operations from the Access Point to the Cloud node, which may include monitoring and AI/ML
- Autonomous operation: Based on Intent-based and zero-touch networking paradigms, autonomous operation is built using closed-control loops at various levels, from device to network.

The major parts of the architecture include: service orchestration and planning, packet optical-integration systems, telemetry and intent-based networking. As presented in D4.1, the main control plane innovations are:

- [**multiband control**] Control of optical multi-band network, this means being able to exploit the multiband capabilities of optical devices such as transmission (transceivers) or switching (multi-band ROADMs).
- [**transparent multi-domain**, *domain-less*] The ability to set up connections in a transparent manner, across multiple domains and network segments. This is exemplified in the "*multi-OLS*" scenario, in which different optical line systems are interconnected without a O/E/O conversion. There is a systematic need to extend SDN principles to networks composed of multiple domains and technological layers, significantly more complex than single domain networks due to the lack of detailed and global topology visibility. The division into domains is driven by factors such as scalability limitations, confidentiality requirements, or interoperability issues, and the conception of scalable, efficient reliable, and trustable systems for the provisioning of end-to-end services.
- [**Packet/optical integration**] The evolution from discrete optics towards pluggable interfaces is also challenging the design of the control plane which, to a large extent, has considered the control plane of the IP/MPLS layer largely decoupled from the control plane of the optical layer. Current architectures for the SDN control plane of the transport network consider the scope of the control considering discrete transceivers and the tunability of the transceiver was directly under the control of the optical SDN controller and multi-layer networking was commonly accomplished

typically with a hierarchical arrangement of controllers (a packet controller and an optical controller under the orchestration of a parent controller). This is addressed in B5G-OPEN, considering multiple options including *exclusive* or *concurrent* control.

- [**physical layer impairments, PLI**] accounting for PLI is critical to efficiently plan and operate optical networks and high data rates, with increasing non-linear effects. When considering the extension to wide-band, such parameters can be specific to certain frequency bands and one can no longer assume uniform channel behaviour. Until recently, there has been a lack of common, standard, and open data models for physical impairments, a domain where it has been difficult to reach a wide consensus. Current systems need to interoperate with heterogeneous monitoring info sources and proprietary and costly simulation tools are difficult to interop or integrate. The new opportunities associated to the development of planning, validation, and path computation tools such as the Open-Source GNPy[Fer20] or Net2Plan[Pav15] has once again shown the importance and role of standard and open interfaces. The challenge is then two-fold: how to integrate such third-party, externalized tools and from a modelling perspective, how to extend current network and service models to account for PLI. This includes a finer characterization of transceivers operational modes, which characterize a given transceiver's different transmission modes including aspects such as bit/baud rate, FEC or modulation formats, as is being done in OpenConfig manifests, IETF operational mode characterization or TAPI transceiver profiles. Additionally, further work is required to model optical fibers – including the selection of a relevant sent of parameters --, amplifier functions e.g., in terms of parameters such as wavelength dependent gain, operation mode, noise figure as well as network elements such as ROADMs. Finding the right abstraction level, where a given model can be applied to a multiplicity of devices from different providers is challenging.
- [**telemetry**] The scope of the SDN no longer covers exclusively device / system control and configuration aspects but extends to optical monitoring and telemetry, a key enabled for advanced functions such as autonomous/autonomic networking via hierarchical and coordinated closed loops. Streaming Telemetry protocols and architectures such as gRPC/gNMI are increasingly being used to export telemetry data from devices, providing flexibility in the definition of streams, filtering, and use cases. Telemetry architecture is detailed in Section 2.6.
- [**external planning tools**] Planning tools, including QoT estimators or path computation and validation systems need efficient access (in terms of retrieval, storage and processing) to collected and managed data. Algorithm inputs need to be modelled in an efficient and scalable way, defining dynamic workflows with controlled and minimized impact on service provisioning latency. Algorithmically, functional elements dedicated to generalized Routing and Spectrum Assignment (RSA) or function placement are needed and are expected to operate in hybrid off-line/on-line modes, e.g., dynamically, used to compute/validate e.g., OTSi services over specific bands with satisfactory QoS/QoT. In this sense, further work is needed to have a *unified short-term provisioning and long-term network-planning* using a single software framework. Such systems need to scale in complexity. The fact that data is heterogenous and covers multiple application domains renders the development of placement algorithms of orchestrator schedulers that need to retrieve network information from multiple layers and domains extremely complex.

- [**network automation**] Aspects related to automation, zero touch networking and Intent Based Networking (IBN) are developed in the areas of service deployment, network planning and overall network operation. Outcomes related to automation in single domains and later cross-domain automation (across technology layers or network segments).

2 HIGH-LEVEL DESCRIPTION OF THE CONTROL PLANE

2.1 B5G-OPEN CONTROL PLANE SERVICES

The Figure 2-1 shows a simplified representation of the control plane architecture.

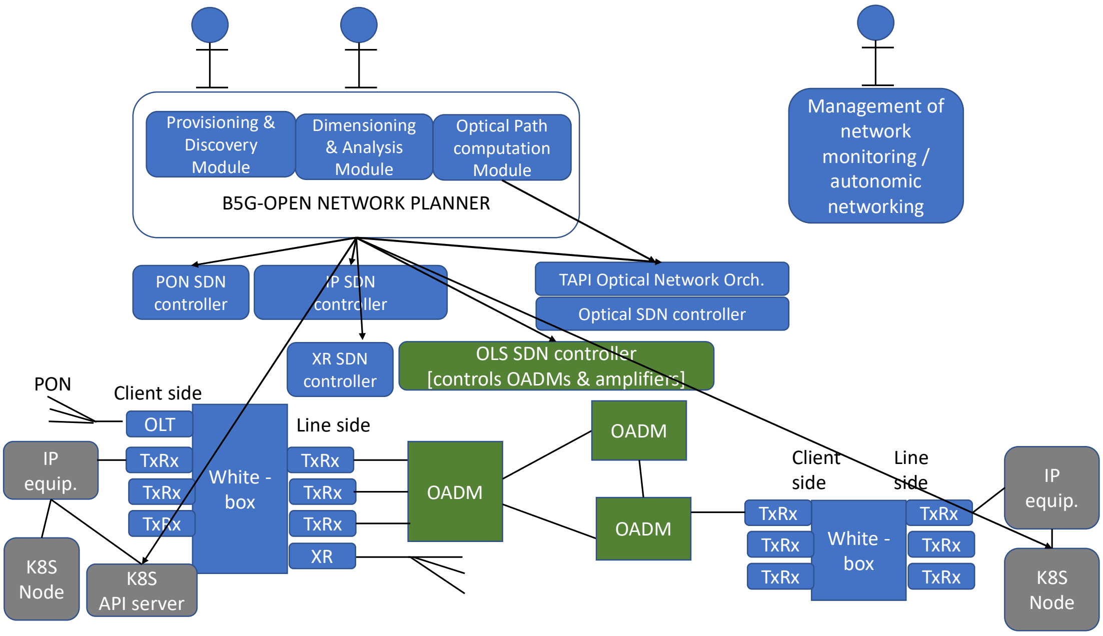


Figure 2-1 Macroscopic B5G-OPEN architecture and Service instantiation interfaces.

The following subsections summarize the targeted services, presenting a brief description and applicability statement.

2.1.1 Point to Point Optical Connectivity

The Point-to-Point Optical Connectivity service addresses point to point connection between optical ports, corresponding to, for example, the line ports of packet/optical devices or discrete transceivers (when the configuration remains at the OTSi layer) or corresponding to ROADM add/drop ports.

2.1.2 Point to Point DSR Connectivity

This service addresses Digital Signal Rate (DSR) provisioning between two stand-alone transceivers or whiteboxes (term that refers to a network element that uses a chassis and a node operating system, often provided by different vendors, and for which most of the components are open) with integrated transceivers. It is part of IP link provisioning between elements (packet/optical nodes) and relates to creating, dynamically and in real time, connectivity to support packet transmission between whiteboxes. Given end transceivers, rate and applicable constraints, the control plane configures and activates the "line part" of the transceiver (modulation, spectrum). Note that the creation of a DSR connectivity service typically triggers the interaction with the optical SDN controller and OLS controller, including, eventually, the creation of OLS point to point connectivity (see above).

2.1.3 Point to Multipoint connectivity

This service addresses the provisioning of a point to multipoint connection from a hub to several leaves. The service is realised by means of OpenXR configuration of the transceivers and relies on a dedicated sub-controller. This OpenXR controller is under the control of the

B5G-OPEN orchestrator, and logically provides multiple point-to-point links between routers attached to the hub (root) and leaves of the system.

2.1.4 IP link provisioning

Related to the previous service, and given an existing DSR service, the B5G-OPEN orchestrator interacts with IP SDN controller to configure the transceivers as IP interfaces in the whitebox. The newly created DSR connectivity becomes a logical interface (e.g., serialXX, ethXX), and The DSR connectivity is seen by the device as a physical port with an associated logical interface (...) which can be used to forward packets (of any kind, not only IP, for example LLDP, IS-IS, etc). This is shown in Figure 2-2, and the relevant list of operations to perform can cover e.g., interface activation, IP address configuration, etc.

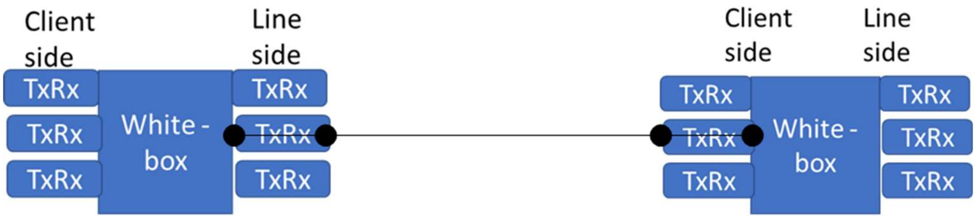


Figure 2-2 IP link provisioning between the whiteboxes.

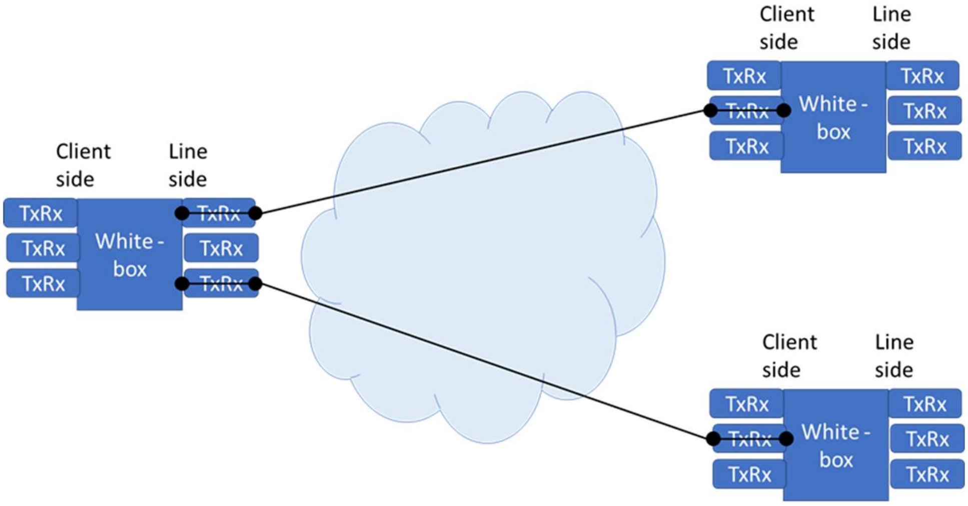


Figure 2-3 multiple IP link provisioning between the whiteboxes using P2MP XR.

2.1.5 Packet/IP Connectivity

Generally speaking, IP connectivity relies on the existence of IP links between whiteboxes. When we consider packet or IP connectivity, we refer to configuring packet switching at the Packet/Optical nodes. This configuration can rely, typically, on IP forwarding or in more advanced SDN-based solutions, such as those based on P4. In this context, an SDN controller may either i) configure IGP/routing protocols (such as OSPF or BGP) or ii) provide flow configuration for flow switching, based on e.g., addresses, ports.

For **non-connection-oriented IP**, (regular IP routing) given end IP routers (whiteboxes), rate, IP QoS, and constraints, it is responsibility of the B5G-OPEN Orchestration platform to check (via Dimensioning & analysis module) if there is enough IP capacity and take the decision of making the required IP link/DSR provisioning.

2.1.6 P2MP Access Connectivity

The orchestrator is also responsible to ensure P2MP connectivity with the access segment. This involves the configuration of the PON controller and is detailed in Section 2.3

2.1.7 B5G-OPEN Network Slice

In this context, a B5G-OPEN slice is defined as a set of interconnected computing and storage functions, deployed within the B5G-OPEN infrastructure, and which involves the orchestration of heterogeneous computing, storage, and networking resources.

2.1.8 Other services

2.1.8.1 Telemetry services

At any part of the control plane architecture, systems and devices may export telemetry services. Telemetry clients may connect and be updated with events, telemetry data etc.

2.1.8.2 Optical Topology Services

Clients MUST be able to retrieve the topology of the underlying optical network. This means being able to retrieve the set of links, nodes, and ports associated with the different layers and, notably, including additional information that may be useful for externalized path computation entities.

2.1.8.3 Optical Path Computation Services

Clients MUST be able to perform path computation on the underlying topology. This can be consumed internally or left for external clients.

2.1.9 Telemetry and Intent Based Networking

The domain telemetry collector architecture has also been defined. It involves a Telemetry Manager with its own repository as well as telemetry agents that sit on different elements, using the REDIS database. Intent Based Networking Applications implement Knowledge Sharing and rely on the services offered by the different functional elements.

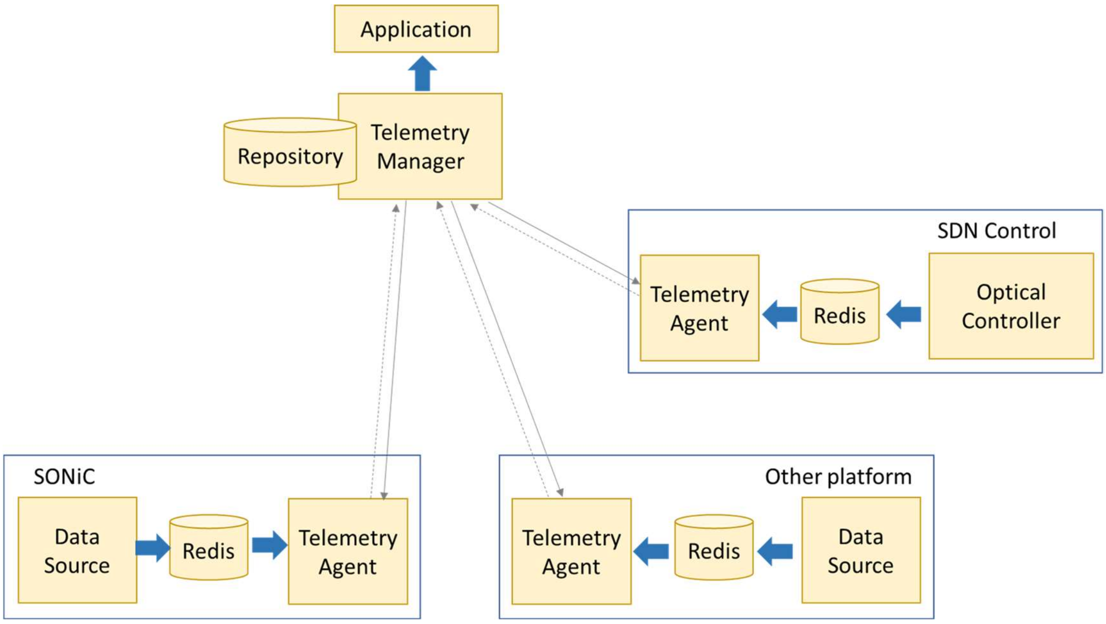


Figure 2-4 B5G-OPEN Control and Orchestration architecture

Finally, the B5G-OPEN architecture spans from the Access Point to the Cloud node, which might include monitoring and AI/ML. Based on Intent-based (IBN) and zero-touch networking paradigms, autonomous operation is built using closed-control loops at various levels, from device to network. Empowered by a distributed AI/ML-based engine providing data collection and intelligent aggregation, analysis, and acting on the network devices, autonomous operation enables coordinated decision-making across domains. This is shown in Figure 2-5.

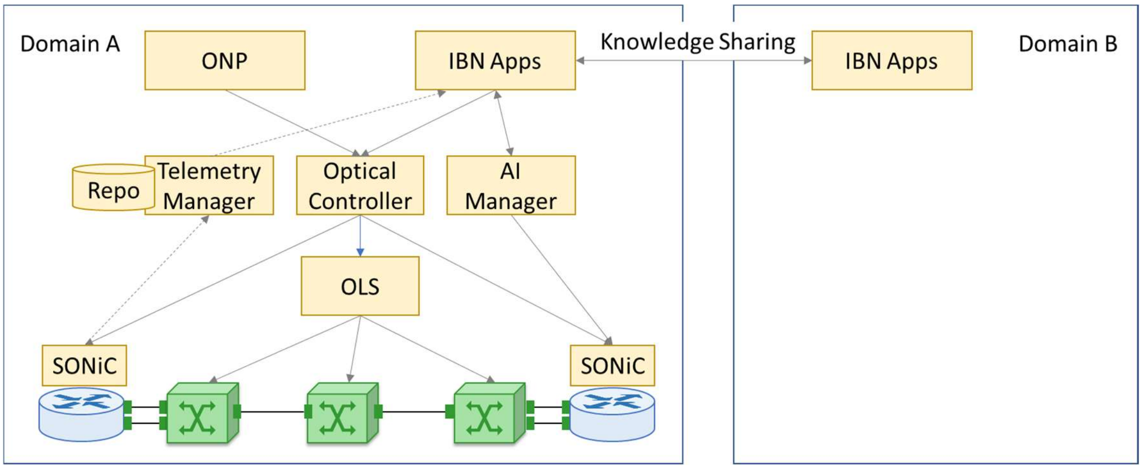


Figure 2-5 B5G-OPEN Intent Based Applications (IBN) and Knowledge-Sharing.

2.2 Optical Network control

2.2.1 TAPI-enabled Optical Network Orchestrator (TAPI NOrch)

The TAPI-enabled Optical Network Orchestrator is a functional element of the architecture that is responsible for the following functions:

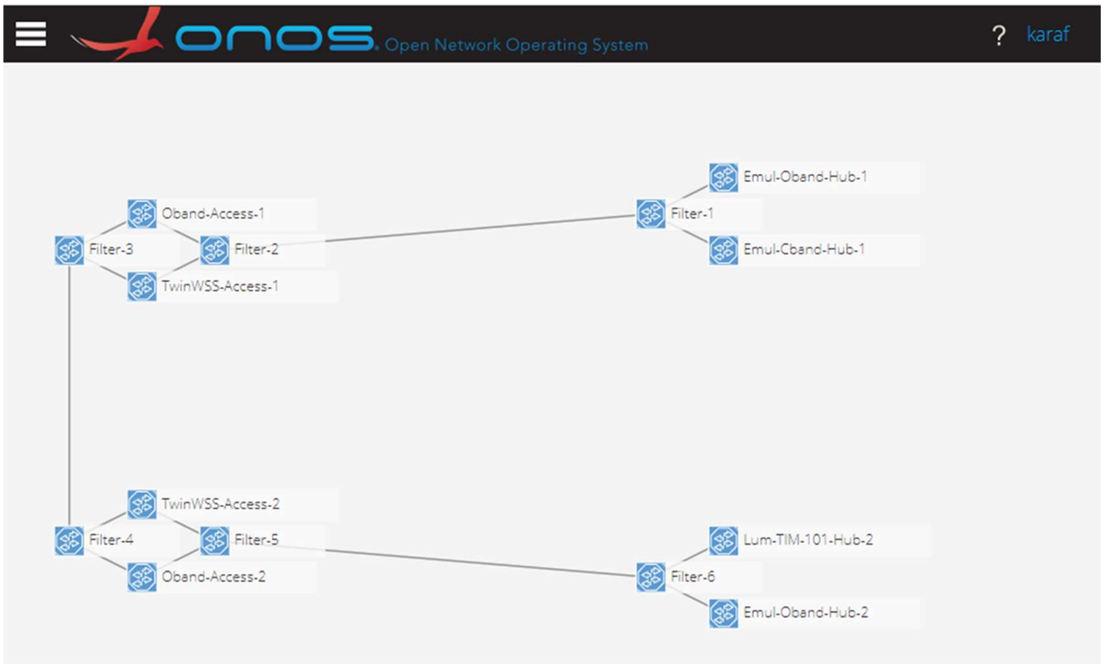
- Providing a uniform, open and standard view and interface to the higher levels and components of the B5G-OPEN control, orchestration, and telemetry system.
- Compose a complete Context to be consumed by B5G-OPEN network planner and additional consumers combining information retrieved from subsystems and sub-controllers (Optical Controller, external databases, monitoring systems, etc).
- Enable single entry point for provisioning DSR and Photonic Media services, including externalized path computation.
- Provide an event telemetry data source that reports events that happen asynchronously in the network.

2.2.2 Optical Controller

The optical controller is based on ONOS SDN controller that provides a wide environment that is used to control and configure optical devices and transceiver equipped within packet/optical white boxes. In particular, the main roles of the optical controller are: (i) retrieve devices description from data plane and abstract them toward the upper control layers; (ii) receive the service configuration requests by the upper control layers and translate such requests in a set of configuration messages to be forwarded to each involved device.

The 3.0 version of ONOS have been forked at the beginning of the project and augmented with several project specifics features published in a public repository. Some selected features have been also exported and merged into the main ONOS distribution. Work done has been mostly oriented to enable the integration with other components of the B5G-OPEN control plane such as the T-API orchestrator (NBI), devices OpenConfig and OpenROADM agents, and to introduce the support of flexi-grid and multi-band in the controller’s core.

The figure below illustrates the ONOS GUI deployed at the TIM premises for the flex-grid, multi-band experimental testbed, where both control plane and data plane B5G-OPEN components have been integrated and validated. In particular, the ONOS controller was used to control two network domains. The one illustrated in the figure includes O-BAND switches (implemented using TUE devices), C-BAND switches (implemented using commercial devices), and emulated multi-band filters. The testbed also included SONiC-based white-boxes with coherent optical transceivers that were directly controlled by the network orchestrator. The two ONOS controllers (one per network domain) export the network topology to the T-API orchestrator and have demonstrated the ability to process connectivity requests (creation and deletion) from the T-API orchestrator, consistently configuring all involved data plane devices. This work has been published at ECOC 2024 [Mor24].



2.2.3 OLS Controller

The ADVA OLS controller is based on the Ensemble Network Controller software solution and is offering a northbound ONF Transport-API (TAPI) towards the Optical Controller.

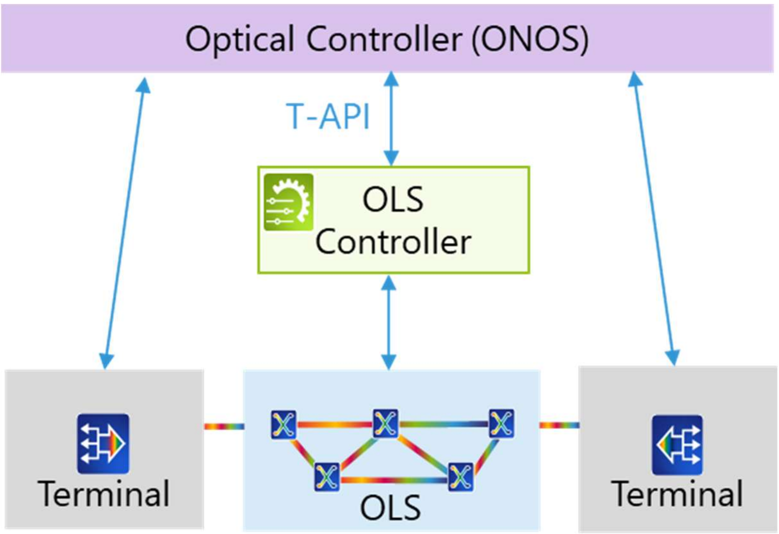


Figure 2-6 ADVA OLS Controller Northbound Interfaces

The OLS controller is exposing the topology. The topology model provides the explicit multilayer topology that the Layer 2 to Layer 0 represents. This topology includes the OTS, OMS, and OCH. Based on ONF TAPI 2.1 models, the OLS controller 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 OCH, and Photonic Media, which include media channels, OMS, OTS and so on. This topology is modelled as a tapi- topology:topology object within the tapi-topology:topology-context/topology list.

2.2.4 Optical Path Computation Element

In B5G OPEN, TAPI has been chosen as the NBI for the optical network controllers (TAPI Optical Network Orchestrator), handling the provisioning and control of optical connections. The optical SDN controller may optionally use an external Path Computation Element, for assisting it in the path computation of the connections.

In TAPI, the Optical Path Computation Element (OPCE) determines an end-to-end path between Service Interface Points (SIPs) and is developed as a TAPI-enabled component. The orchestrator sends to the OPCE a TAPI path-request. This module requests an abstract topology from the context manager, calculates the path and responses with TAPI path-reply after finding a path within that internal context. The interactions between the OPCE and the TAPI- Optical Network Orchestrator element is governed by the standardized Path-Computation-Service interface and APIs, as defined in [Man21], and when needed, standard extensions may be proposed along the project.

2.2.5 Multi-domain scenarios

Of special interest for B5G-OPEN is the “multi-OLS scenario”, (see Figure 2-7) which is to be considered for use cases related to the provisioning of services across a muti-segment network in a transparent way. In the multi-OLS scenario, several domains are interconnected transparently (e.g. via optical links), connecting, for example a degree of a ROADM to a degree of a ROADM or add/drop to add/drop, as shown in the figure). Such scenarios shall be addressed with an arrangement of controllers and the key issue to research is how to retrieve the abstracted topological information to perform efficient path computation.

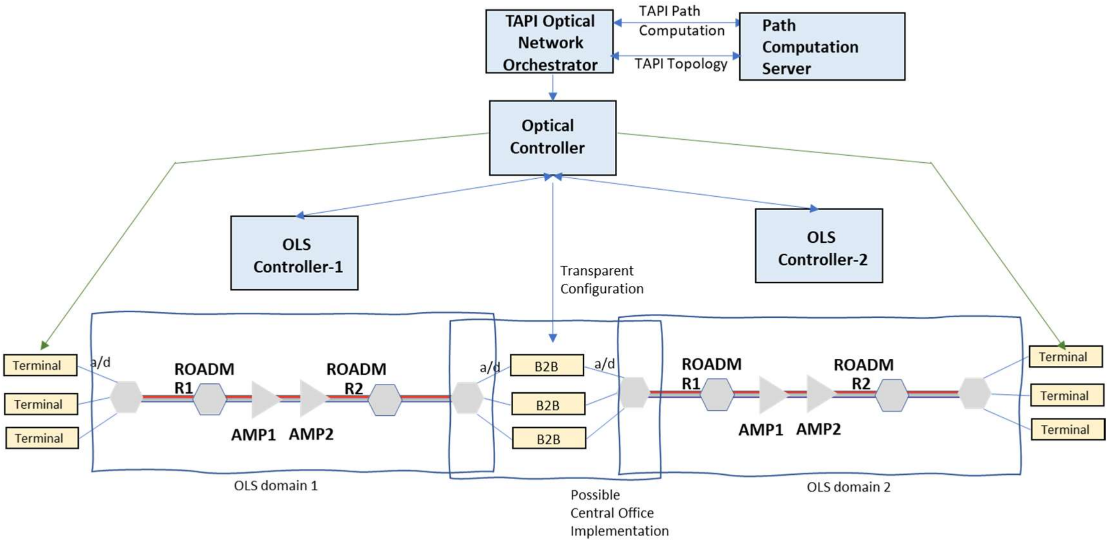


Figure 2-7 Control plane architecture for the multi-OLS scenario, showing a back to back add/drop-add/drop configuration.

2.3 Access Networks Control

The B5G-OPEN control and orchestration software system will also support the control of *access network segments* in addition to the control and orchestration of packet and optical network segments. In this direction, B5G-OPEN will have the capability to control access networks including Passive Optical Networks (PONs) and LiFi networks.

2.3.1 The Framework of TDM-PON Configuration and Control

The B5G-OPEN TDM-PON infrastructure is realised using an XGS-PON OLT pluggable transceiver (e.g., TiBit pluggable) and a couple of pluggable ONUs (e.g., Tibit ONUs). The OLT is interfaced directly to a whitebox switch while the OLT is interconnected to the ONUs by means of splitters, forming up an ODN branch. The integration of these pluggables with the B5G-OPEN software platform is made feasible at three different levels (from higher to lower layer). These options lead to different alternatives for the implementation of TDM-PON's control-plane, presented in the next subsections.

The selected alternative, the PON vendor provides the pluggable software and the PON controller software. The TDM-PON control-plane architecture and its integration to the B5G OPEN platform are illustrated in Figure 2-8. Since the PON Controller is provided by the PON vendor, a Higher-Layer PON Controller is developed as part of the B5G-OPEN software platform, providing a slightly different functionality:

- The information exchange is again based on the BBF/ITU YANG models. However, the SBI that communicates with the PON Controller is a software client that is developed based on OLT PON SDK.
- A NETCONF/REST server at the Northbound Interface (NBI) which exposes a set of APIs that allow the B5G-ONP app to provision and configure the PONs. This API is using a simplified (subset) BBF/ITU YANG model which depend on the abstraction and transformation realised in the lower layer.

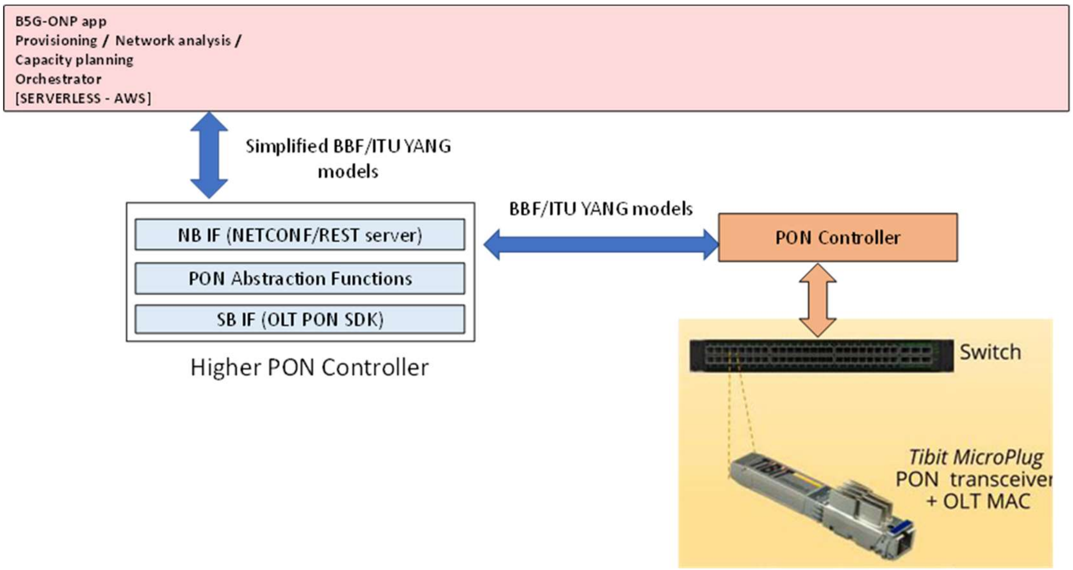


Figure 2-8 B5G-OPEN Control of PON through the PON Controller

2.3.2 LIFI Control

The LiFi access networks is provided by Access Points (APs), named LiFi-XC, provided by pureLiFi.

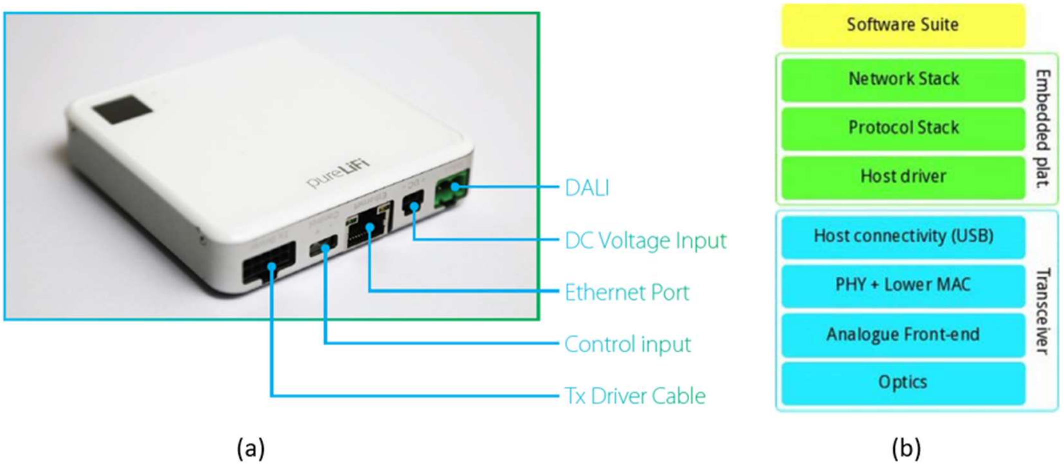


Figure 2-9: LiFi-XC AP

1) The LiFi control for B5G-OPEN supports a NETCONF interface, with a LiFi specific YANG model to configure a LiFi AP. The motivation behind NETCONF and YANG is that instead of having individual devices with functionalities, there is a need to have a network management system that manages the network at the service level. To integrate the LiFi access technology in the overall B5G-OPEN architecture, NETCONF and YANG add more functionalities in the network management.

2) There is a telemetry adaptor within the LiFi AP for LiFi telemetry data collection and transmission.

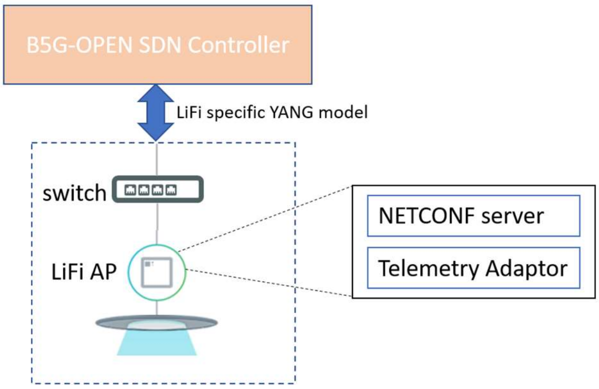


Figure 2-10: Initial assumption for LiFi control

2.4 Orchestration

2.4.1 IT and network resources orchestration

The orchestrations process consists of the coordination of both **IT** and **network** resources of the infrastructure, in an efficient and harmonized form, pursuing a global optimization of the infrastructure usage.

The so-called *slice* is the key service requiring such a joint IT and network allocations. In B5G-OPEN, we generalize the concept of slice as a set of IT requirements to be allocated in the IT infrastructure, together with a set of network requirements connecting them, to be allocated in the network infrastructure. In B5G-OPEN, the orchestration process is implemented in a collaborative form among three key groups of components:

1. The IT resources, potentially distributed in one or more clusters, at different locations across the operators' infrastructure, are handled by one or more IT orchestrator systems.
2. The network resource, involving IP/MPLS and optical layers, are controllable via one or more SDN controllers.

The coordination of IT and network resource allocations is handled by the B5G-ONP (Open Network Planner). The key functions of the ONP are providing tools for the design, optimization, and planning of services.

2.4.2 B5G-ONP modules

B5G-ONP consists of three main modules (see Figure 2-11):

- **Provisioning and discovery module.** This module is intended to manage the provisioning and termination of different operator-level services, as the ones discussed in Section 3, that may involve It and/or network resources. Such functions are accessed via an open API designed along the project. However, a Graphical User Interface is prototyped to ease the interactions.
- **Dimensioning and analysis module.** This module hosts different algorithmic resources, that realize the resource allocation decisions, in different use cases, covering both offline network dimensioning, and online resource allocations. These modules are designed to be accessed via an open API defined along the project, and also a prototyped GUI.

- **Optical Path Computation Element.** This module is specifically developed to be able to interact with the TAPI Optical Network Orchestrator, in order to act as an Optical Path Computation Element node, to which the TAPI Optical Network Orchestrator can delegate the optical path computations.

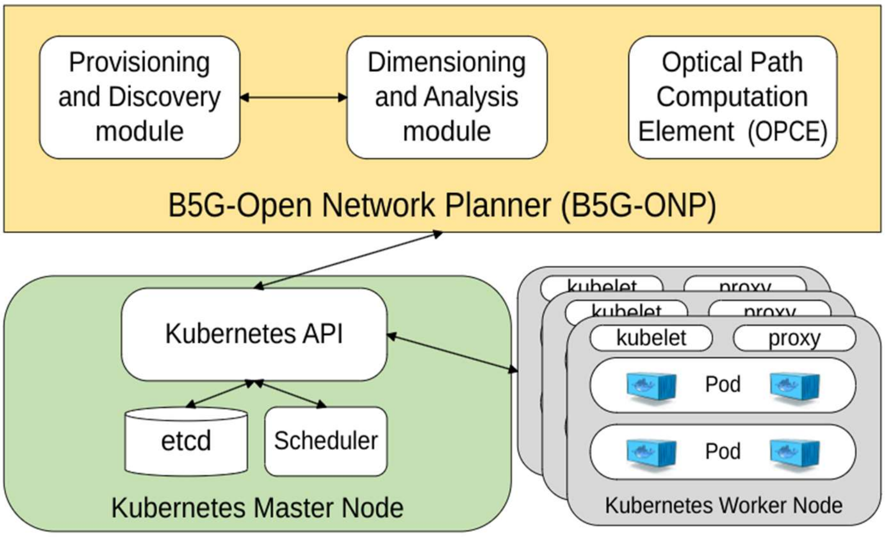

Provisioning and Discovery module
Dimensioning and Analysis module
Optical Path Computation Element (OPCE)
B5G-Open Network Planner (B5G-ONP)
Kubernetes API
etcd
Scheduler
Kubernetes Master Node
kubelet
proxy
Pod
Pod
Kubernetes Worker Node

Figure 2-11 Coordination of Kubernetes cluster from B5G-ONP

2.5 PACKET/OPTICAL INTEGRATION

Two alternative SDN-based hierarchical solutions are in phase of discussion in the community enabling control of coherent pluggable transceivers in a multi-layer network exploiting hybrid packet-optical nodes.

2.5.1.1 Reference scenario and proposed solutions

Figure 2-12 shows a traditional metro network using packet switching nodes (i.e., routers) and stand-alone transponders interconnected through optical line systems (OLSs), where the OLS is typically composed by a number of ROADMs and optical amplifiers. In this scenario, the SDN architecture is implemented with a clear domain separation. Three controllers are typically considered: a Hierarchical Controller (HrC) coordinating the end-to-end connectivity; an Optical Controller (OptC) in charge of transponders and OLS; and a Packet Controller (PckC) in charge of packet switching devices. However, since the two domains are practically independent of each other, the role of the HrC is almost limited to forwarding the received requests to one of the child controllers which, traditionally, has full and exclusive visibility on all underlying network elements. For example, OptC is the unique entity accessing the transponders while PckC is the unique entity configuring the packet nodes.

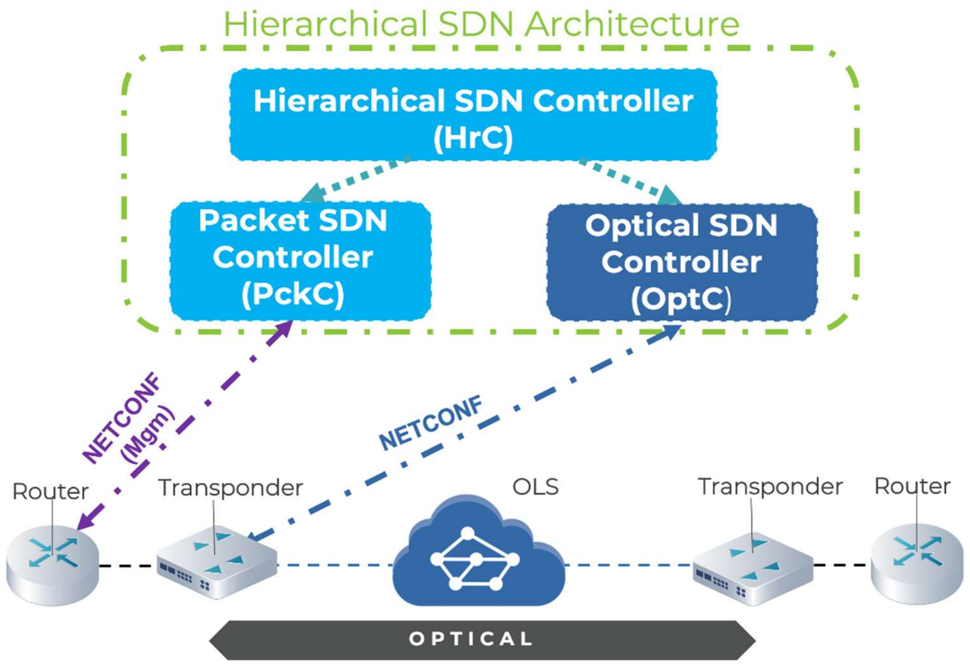

Hierarchical SDN Architecture
Hierarchical SDN Controller (HrC)
Packet SDN Controller (PckC)
Optical SDN Controller (OptC)
NETCONF (Mgm)
NETCONF
Router
Transponder
OLS
Transponder
Router
OPTICAL

Figure 2-12 Traditional SDN architecture for transponder-based optical networks.

The introduction of packet-optical nodes imposes the redesign of the overall SDN control architecture. Indeed, transponders are replaced by packet-optical nodes equipped with pluggable modules and the traditional control mechanisms provided by the PckC only are not sufficient to configure optical parameters. Since, in large metro networks, a single controller with visibility of both layers is not feasible due to scalability issues, a proper workflow needs to be defined to enable coordinated operations among controllers, where the HrC assumes a fundamental coordination role.

2.5.2 Sonic generic architecture

SONiC system's architecture is composed of various modules implemented as Docker containers that interact among each other through a centralized and scalable infrastructure. At the center of this infrastructure resides a redis-database engine, a key-value database that provides a language independent interface to all SONiC subsystems. Thanks to the publisher/subscriber messaging paradigm offered by the redis-engine infrastructure, applications can subscribe only to the data-views that they require. The docker containers run within the SONiC operating system, based on the Linux kernel, at user space level. Linux allows access to the hardware of the machine only in kernel mode, i.e., elevating the privileges of the running process in controlled mode. For this reason, the interface to the underlying hardware takes place by means of appropriate drivers. SONiC exploits the possibility of extending the Linux kernel thanks to the so-called Loadable Kernel Modules (LKM), which avoid the need to prepare a kernel version containing the drivers needed by the specific hardware, considerably simplifying the support of switches with different features.

2.5.3 Pluggable management and control

Whitin the BG5-OPEN project, the SONiC network operating system (NOS) running on the packet-optical node is extended with a new docker container that enables SDN on SONiC. A NETCONF Agent, developed in the BG5-OPEN project, is deployed in a docker container that runs within the NOS and, as depicted in Figure 2-13, communicates with the other containers in the system for retrieving and writing information related to coherent pluggable modules. More in detail, in the SONiC version 202205 the *pmon* container runs an updated version of *xcvrd* daemon, capable to retrieve and write the coherent optical parameters from/to the registers of pluggable modules. The interfaces used by the demon are compliant with the CMIS v5.0 and C-CMIS v1.1 standards. The daemon periodically stores the optical transmission parameters in the Redis database.

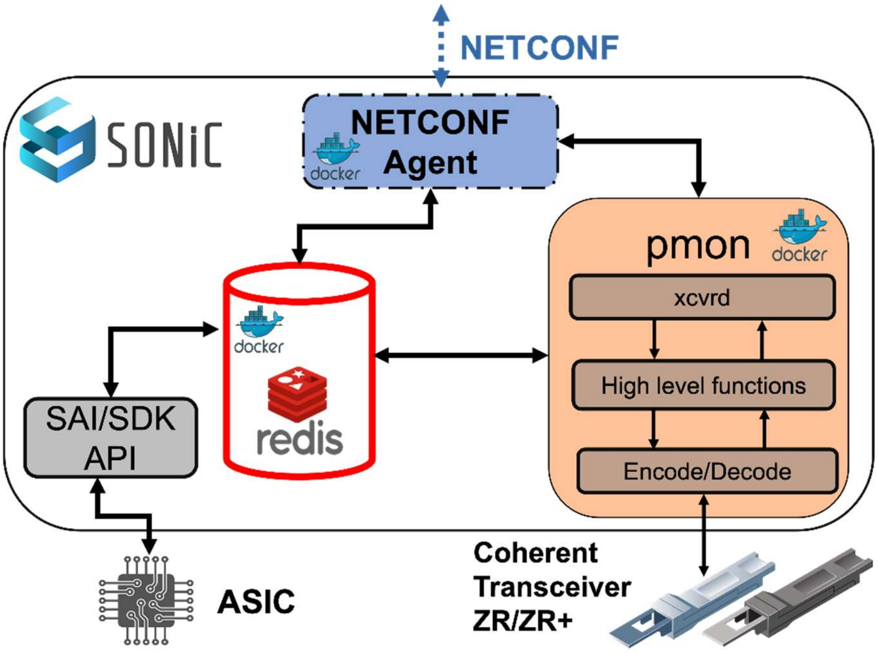


Figure 2-13 Pluggable management and control architecture

SONiC utilizes custom YANG models that do not take into account optical pluggable modules. To address this limitation, in B5G-OPEN, the standard OpenConfig YANG model *openconfig-platform-transceiver.yang* is used within the NETCONF agent to model the optical pluggable modules. More in details, the parameters in the model can be filled in two ways: in the first one, the agent reads the optical parameter stored in the Redis database by the *xcvrd* daemon. In the second one, the agent reads or writes the optical parameters of the pluggable module leveraging the API used by *xcvrd*. Indeed, as depicted in Figure 2-13 two bidirectional arrows reach the agent, they represent the communication interfaces (e.g., a socket or/and REST API), developed in B5G-OPEN to allow the exchange of information between the agent and/or Redis/Pmon containers. In B5G-OPEN the optical SDN controller communicates with the NETCONF agent to monitor and control the pluggable modules placed in the packet optical nodes.

2.5.4 P2MP Pluggable Management and Control

The management of P2MP pluggable modules proposed by Open XR [Swe22] considers a dual management structure. The first path, as shown in Figure 2-14 left side, provides the “traditional” functionality via the register-based information model defined in Multi-source agreements such as OIF CMIS.

However, the latest version of CMIS lack the capabilities of setting up multiple subcarriers or dynamically assigning traffic to the different subcarriers. Hence, a second path, as shown in Figure 2-14 right, is proposed to be able to communicate directly with the P2MP pluggable. When the messages are destined for the Open XR module are received by the router, they are handled by the Communication Agent Service running on the router. These messages are forwarded to the data path entering the Open XR module via the module host electrical lanes where they are recognized as management/control messages and handled appropriately.

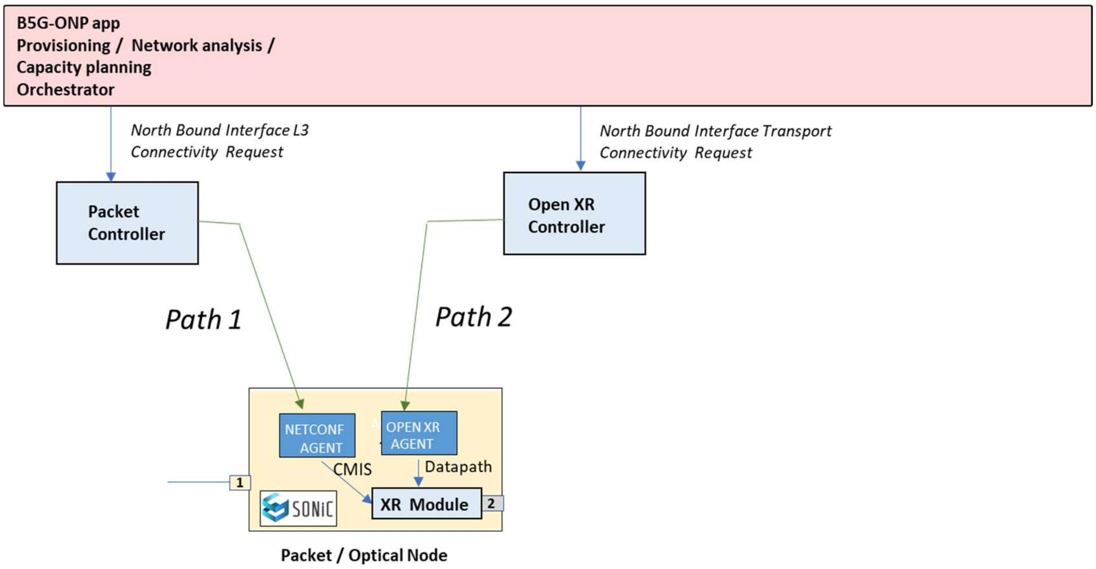


Figure 2-14 P2MP Control integration in B5G-OPEN

2.6 TELEMETRY PLATFORM

Telemetry data is collected from observation points in the devices (*measurements*), as well as *events* from applications/platforms (e.g., Software Defined Networking (SDN) controllers and orchestrator) which are then sent and collected by a central system. A telemetry "collector" or "mediator" agent may overcome this challenge and provide mechanisms to obtain a stable stream of telemetry from legacy devices.

In B5G-OPEN, we have designed a telemetry architecture that supports both measurements and events telemetry. For the former, intelligent data aggregation is placed nearby data collection to reduce data volumes, whereas for event telemetry, data is transported transparently.

2.6.1 B5G-OPEN Telemetry Architecture

Figure 2-15 presents the network scenario, where the B5G-OPEN Control system is in charge of several optical nodes: optical transponders (TP) and reconfigurable optical add-drop multiplexers (ROADM). A centralized telemetry manager is in charge of receiving, processing, and storing telemetry data, including measurements and events. The telemetry database (DB) includes two repositories: i) the measurements DB is a time-series DB stores measurements, whereas the ii) the event DB is a free-text search engine. In addition, telemetry data can be exported to other external systems.

Some data exchange between the SDN control and the telemetry manager is needed, e.g., the telemetry manager needs to access the topology DB describing the optical network topology, as well as the label switched path (LSP) DB describing the optical connections (theses DBs are not shown in the figure). Every node in the data plane is locally managed by a node agent, which translates the control messages received from the related SDN controller into operations in the local node and exports telemetry data collected from observation points (labelled M) enabled at the optical nodes. In addition, events can be collected from applications and controllers (labelled E).

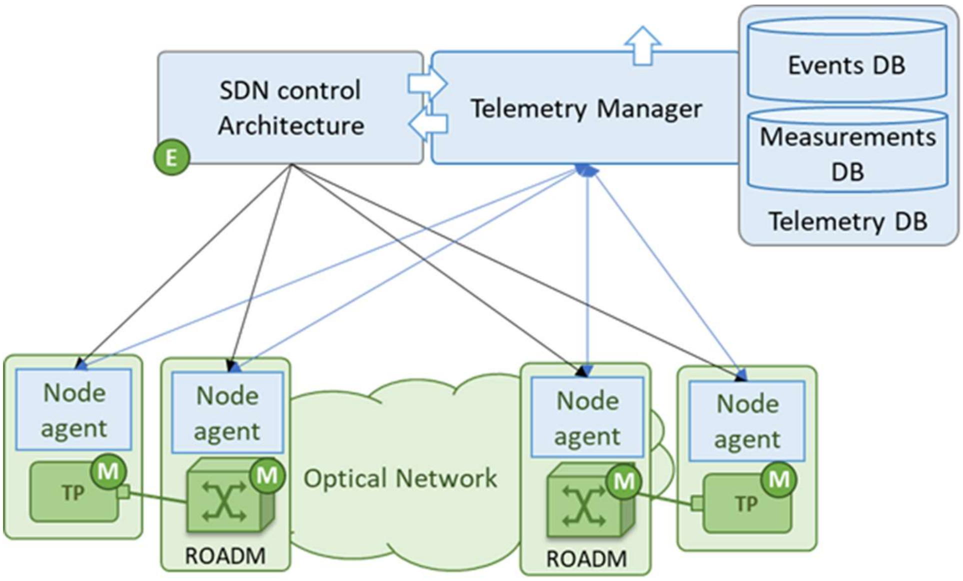


Figure 2-15 Overall network architecture

A detailed architecture of the proposed telemetry system is presented in Figure 2-16 for the case of measurements telemetry. The internal architecture of telemetry agents inside node agents and the telemetry manager is shown. Internally, both, the telemetry agent and manager are based on three main components: i) a manager module configuring and supervising the operation of the rest of the modules; ii) a number of modules that include algorithms (e.g., data processing, aggregation, etc.) and interfaces (e.g., gRPC); and iii) a Redis DB that is used in publish-subscribe mode to communicate the different modules among them. This solution provides an agile and reliable environment that simplifies communication, as well as the integration of new modules. A gRPC interface is used by the telemetry agents to export data to the telemetry manager, and by the telemetry manager to tune the behaviour of the algorithms in the agents.

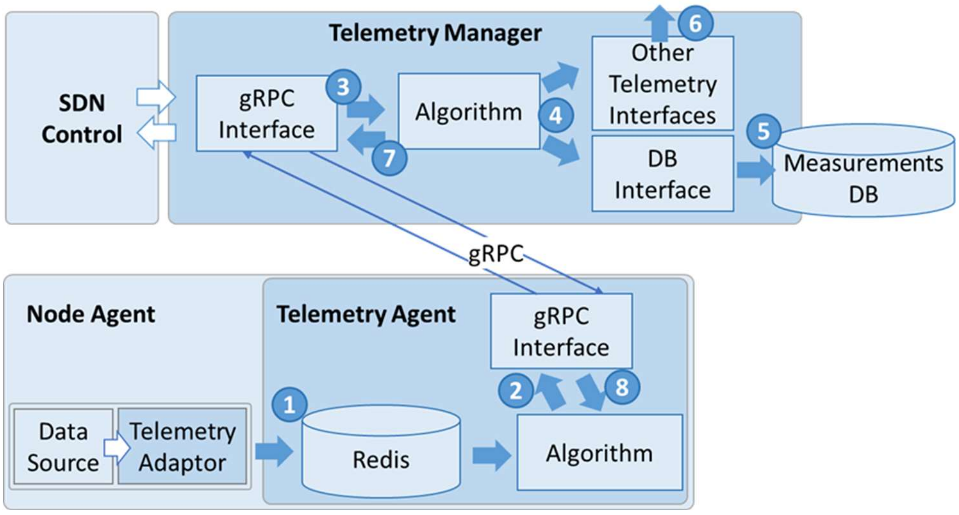


Figure 2-16 Measurements telemetry architecture and workflow

Events generated in a SDN controller (or other system), are injected in the telemetry agent, and transported transparently to the telemetry manager, which stores them in the Events DB and exports to external systems. Note that Null Algorithms are used here just to propagate events, which results in the same workflow as in the case of measurements, but no processing is performed.

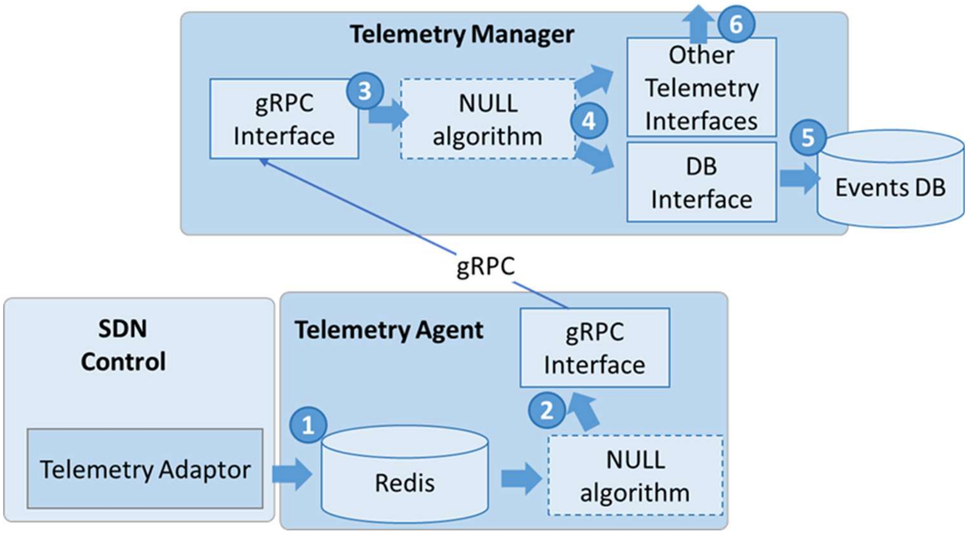


Figure 2-17 Events telemetry architecture and workflow

2.6.2 OLS Node Agent and Telemetry Adaptor

The OLS Node Agent is a Python-based application designed to stream telemetry data from multiple devices simultaneously. In the southbound direction, it collects data using NETCONF (for general telemetry) and SNMP (specifically for amplifiers). The agent can then push the collected metrics to various northbound plugins, including Redis, Apache Kafka, MQTT (Mosquitto), and InfluxDB. The agent handles telemetry data from a range of devices, including optical transponders, Carrier Ethernet switches, and optical amplifiers. When data is sent to message brokers like Kafka or MQTT, the Python script automatically initiates a Telegraf instance to collect and push these metrics into InfluxDB. This setup enables real-time data handling for applications that require immediate performance data, such as machine learning models.

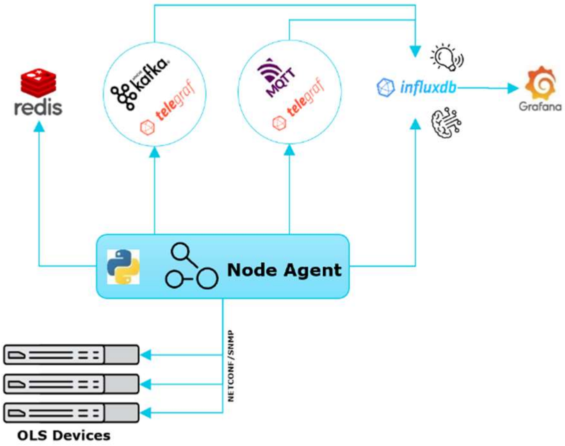


Figure 2-18: Overview – OLS Node Agent

Additionally, the system stores historical data in a time-series database, which is advantageous for retrospective analysis and for training machine learning models. This dual approach supports both real-time analytics and long-term data retention, allowing flexible data handling to meet varied application needs.

2.7 Autonomous Networking and Quality Assurance

The monitoring and performance telemetry system developed in this consortium will enable to close a control loop and envisage autonomous network operations.

2.7.1 Autonomous Networking

Optical Autonomous networks are based on several building blocks addressed in this project: physical impairment modelling and performance monitoring, telemetry systems and a control and orchestration. From these building blocks, we envisage three main architectures to define the control loop:

- a local control loop: This scenario is leveraging some limited intelligence at the node level. The main objective is the live optimization of a reduced set of parameters on a lightpath. One can cite the work already achieve by the members of the consortium on frequency optimization to mitigate the filtering penalties [Del19a], power optimization to mitigate transient loss [Gou21] hitless baudrate switching [Dut22].
- A domain control loop: This scenario is the most common and is leveraging intelligence in a centralized architecture. A wide-ranging set of applications for closed loop reconfigurations can be deployed and are triggered in response to events identified in the central Telemetry Manager. Such an architecture, while not giving the best performance in term of reaction speed, will certainly provide the best overall decision [Del19b].
- A multi-domain control loop: This scenario is probably the most challenging as the parameters from one domain are not opened to the other domain and there is a need to rely on the previously explained knowledge sharing. It is also a centralized architecture empowered by AI/ML to have autonomous networking coordinated across domains without exchanging internal domain details.

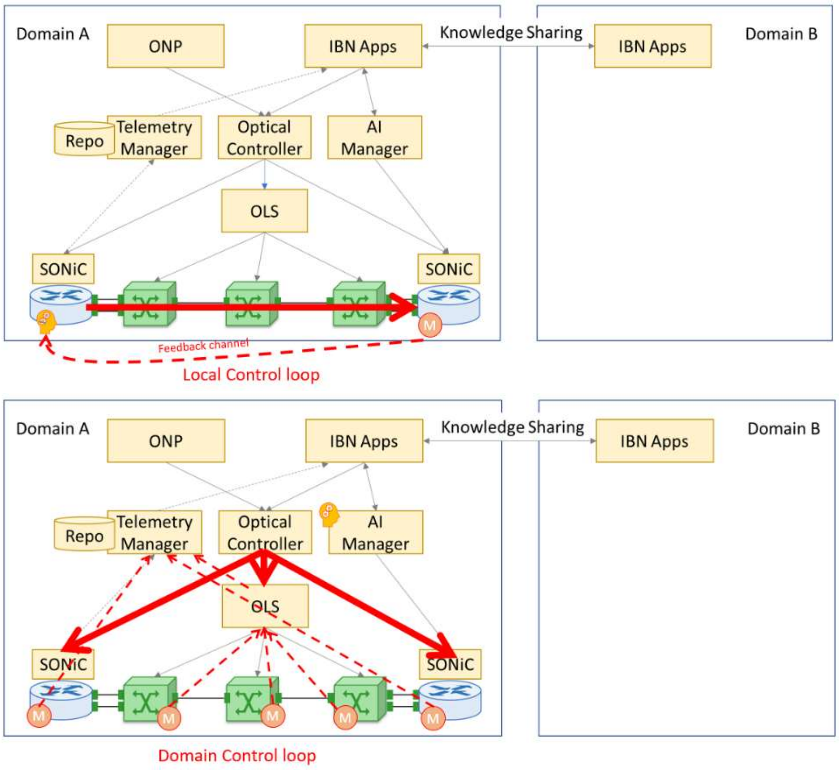


Figure 2-19 Intra domain Control loop architecture

2.7.2 Single-Domain and Multi-Domain Quality Assurance

Quality assurance is based on Intent Based Networking (IBN) [IBN] applications to represent the optical transport network (Figure 2-20). In this section, we rely on a deep learning-based IBN application for the optical time domain, named OCATA [Rui22], which initial concept has been developed in B5G-OPEN. OCATA is based on the concatenation of deep neural networks (DNN) modelling optical links and nodes, which facilitates representing lightpaths. The DNNs can model linear and nonlinear noise, as well as optical filtering. Additional DNN-based models are proposed to extract useful lightpath metrics, such as lightpath length, number of optical links and nonlinear fibre parameters.

OCATA includes a sandbox domain to pre-train DNN models, based on the measurements available through telemetry. Such models are made available to IBN applications, which use them to generate expected signals that can be compared with those obtained from the network. In that way, deviations between the observed and the expected signals can be detected and used for, e.g., soft-failure detection, identification, and localization.

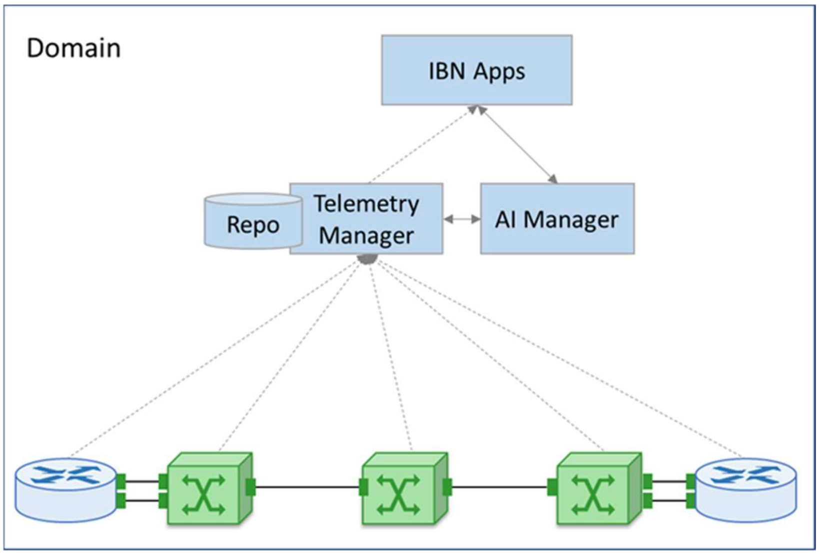


Figure 2-20 Intent-based networking in the intra-domain

Because telemetry and DNN models are domain internal, knowledge sharing is proposed for the IBN applications to solve the problem of inter-domain scenarios (Figure 2-21). IBN applications exchange their internal models for the segment of the optical lightpath in their domain. By working on DNNs' internal architecture to ensure not disclosing internal domain details, such models can be shared among different domains to create end-to-end lightpaths' models. Armed with such end-to-end lightpaths' models, domain IBN applications can carry out diagnosis and collaborate to localize failures.

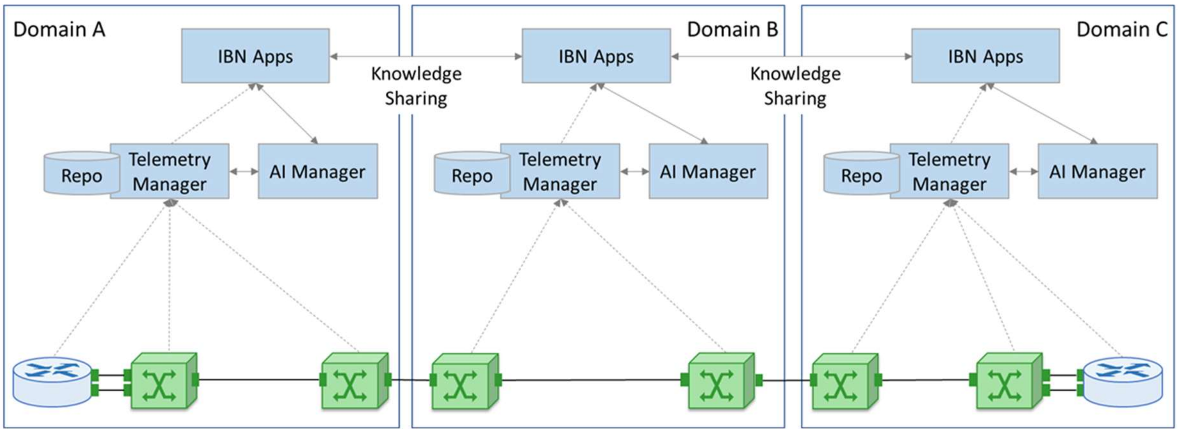


Figure 2-21 Intent-based networking in multi-domain scenarios

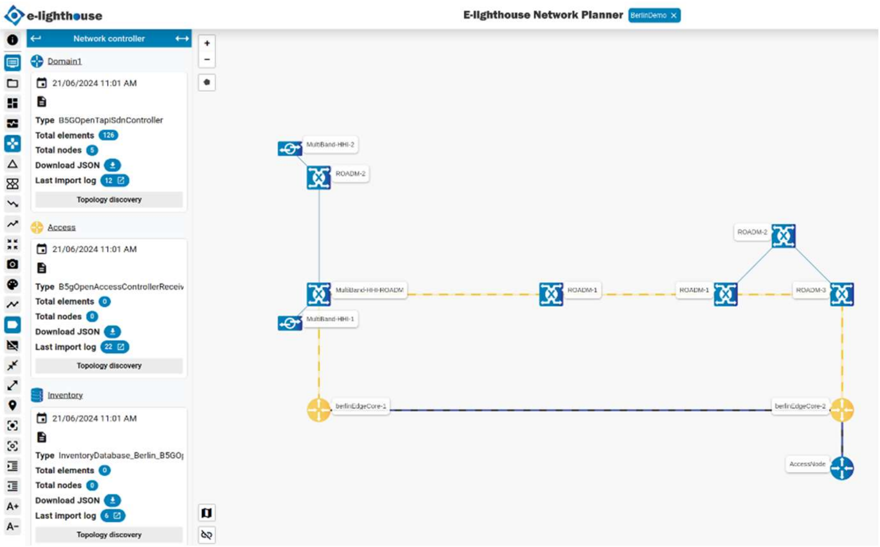
3 B5G-OPEN Software Components

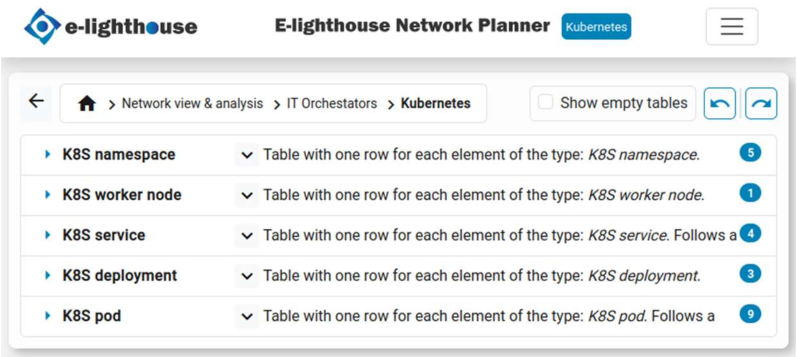
This section lists and summarizes the main software components that have been designed, implemented and used in B5G-OPEN WP4 and WP5, including previously existing components that have been extended to address B5G-OPEN objectives and innovation aspects.

3.1 B5G-ONP (ELIG)

Component Name: B5G-ONP	
Summary	This component is part of the control plane and orchestrates the IT and network resources. B5G-ONP provides design, optimization and planning tools to deploy, manage and configure services and resources, easing the integration with external components. It includes three main modules: i) Provisioning and Discovery module; ii) Dimensioning and analysis module; and iii) Optical Path Computation Element. The B5G-ONP communicates with the application/service layer (Operators) via its Northbound Interface (NBI) and interacts with the rest of the control plane elements (PON SDN Controller, IP SDN Controller, XR SDN Controller, TAPI Optical Network Orchestrator, and Kubernetes) by using the Southbound Interface (SBI).
Description and Internal architecture of the component	B5G-ONP is a control plane module which allows the coordination and orchestration of IT and network resources and provides the design, optimization and planning tools to deploy, manage and configure services and resources. Additionally, the B5G-ONP will prototype a user-friendly Graphical User Interface (GUI) aiming to improve the Quality of user Experience (QoE) and ease the interaction with the underlying components present in the network. Concerning process automation and Zero-Touch management, the B5G-ONP will expose a Northbound Interface (NBI) REST API to be used by network operators and higher-level components or services. On the other hand, B5G-ONP uses the Southbound Interface (SBI) to integrate with external components such as PON SDN Controller, IP SDN Controller, XR SDN Controller, TAPI Optical Network Orchestrator, and Kubernetes. The B5G-ONP component is a self-contained unit that integrates different modules to add the required functionalities and improve the network performance by having a clear understanding of the global requirements. The integrated modules are: - The Provisioning and Discovery module is responsible for orchestrating the allocation of network resources (physical and virtual) automating the deployment and configuration based on user requests, policies, and predefined templates. This unit scans and imports the network segment topologies and identifies the available services and dependencies as a centralized repository of this information and a comprehensive platform for managing the complex network environment for network administrators. - The Dimensioning and analysis module is in charge of predicting and optimising the performance of the network infrastructure by taking decisions on capacity planning and network dimensioning. This unit

	uses data analytics and ML techniques to examine the network traffic patterns and resource utilization to predict the performance requirements (e.g., capacity, latency, jitter, etc.) of the network according to the topology and application requirements. - The Optical Path Computation Element considers technical factors to determine the best path for the optical connections based on available resources in the B5G-ONP reducing congestion and improving efficiency. This module requires the previous discovery work, estimating the performance of the different admissible paths. B5G-ONP will analyse the performance of the new path-candidates and establish the optimal configuration needed on the underlying resources (e.g., Optical SDN controller) to ensure the Quality of Transmission (QoT). This module is integrated in B5G-ONP as part of the network orchestration easing the integration with external components through a unique entity.
Interface Specification	The Northbound Interface (NBI) is the top interface between the B5G-ONP and the application and services that use the network resources based on REST API. The GUI uses the exposed information on this interface to define a pretty-printed schema of the network. The network operators define the network requirements via this interface (e.g., real-time network tasks such as the bandwidth allocation to a specific application, traffic priority, etc.), and underlying controllers configure the network resources and policies on network devices. The Southbound Interface (SBI) of the B5G-ONP, on the other hand, is the interconnection point with external components that B5G-ONP manages, translating the high-level network requests into concrete actions in layers below to drive the network configuration accordingly via NBI exposed by the controllers. This interface may include the RESTCONF protocol and different standards to enable communication with other modules such as Transport API (TAPI) or IETF RFC8345. B5G-ONP SBI enables network automation tasks (e.g., provisioning new network equipment, in real-time) improving the efficiency, availability and reliability of management tasks. The Optical Path Computation Element exposes an autonomous API, accessed by the TAPI Optical Network Orchestrator for planning and provisioning the B5G-OPEN networks. Exemplary Workflow: Once the network operator introduces the network requirements, the B5G-ONP starts discovering the network topologies in the different segments and continues with the analysis based on the requirements and the available resources. Planning and dimensioning tasks, using the results of the analysis operations, will consider the required changes in the network to improve the performance and to meet the expected KPIs. B5G-ONP translates the adopted solution into low-level commands/calls into concrete actions to the corresponding control-plane entities via SBI. Once actions are applied, the orchestrator validates and tests the network performance under different conditions. Conversely, when external components report metrics back to the B5G-ONP the NBI must be able to

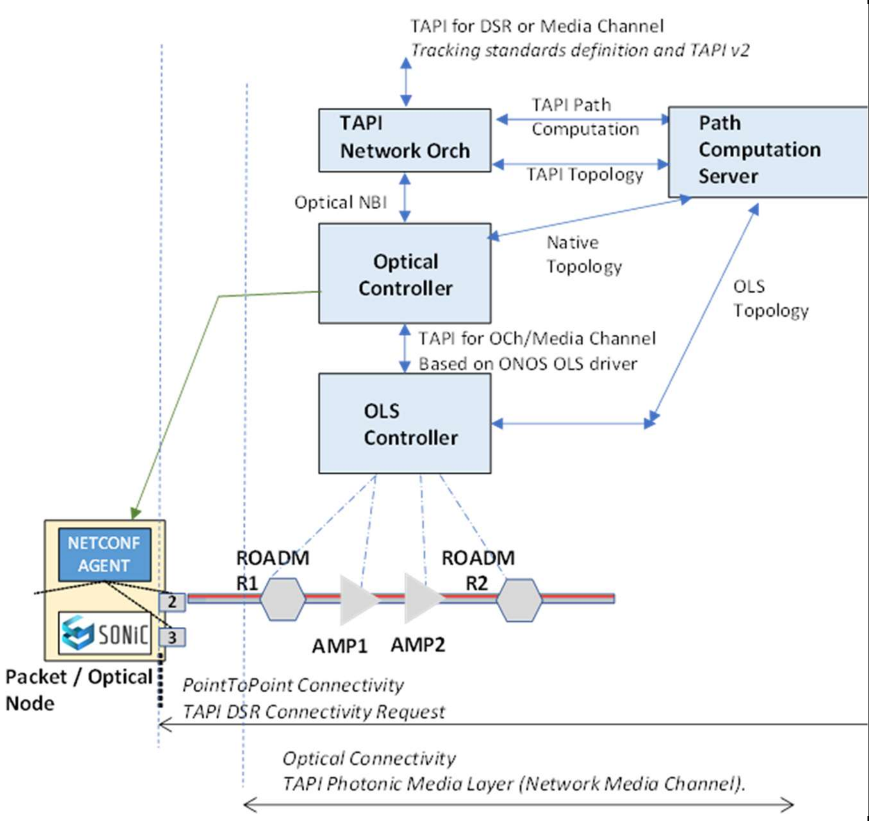
	interpret those metrics and present them to the network administrator in a meaningful way.
Functional Validation	Tests done to validate the component: - The B5G-ONP component was tested by handling the expected user loads with minimal response times. The system effectively scaled up or down based on demand, performed concurrency operations, and maintained high efficiency under stress conditions. - Discovery validations from IP and optical segments were completed, ensuring accurate and coherent responses. The GUI correctly imported the complete topology and displayed it in the layout, providing a clear view of the system's status. - Provisions were executed to the required controller for creating new links within the topology. The GUI displayed these new links in the layout, accurately reflecting the requested changes (see figure below).  e-lighthouse E-lighthouse Network Planner BerlinDemo Network controller Domain1 21/06/2024 11:01 AM Type B5GOpenTapiSdnController Total elements 126 Total nodes 5 Download JSON Last import log 12 Topology discovery Access 21/06/2024 11:01 AM Type B5gOpenAccessControllerReceiv Total elements 0 Total nodes 0 Download JSON Last import log 22 Topology discovery Inventory 21/06/2024 11:01 AM Type InventoryDatabase_Berlin_B5GO Total elements 0 Total nodes 0 Download JSON Last import log 6 Topology discovery MultiBand-HHI-2 ROADM-2 ROADM-2 MultiBand-HHI-ROADM ROADM-1 ROADM-1 ROADM-3 MultiBand-HHI-1 berlinEdgeCore-1 berlinEdgeCore-2 AccessNode - The analysis module's proposals were validated against existing network conditions, confirming that it offered optimal solutions based on current data. - In a Kubernetes context, tests were conducted to read the cluster's status, deploy new services or deployments, and verify the correct implementation and status of these services (see figure below). All operations were executed successfully and validated.

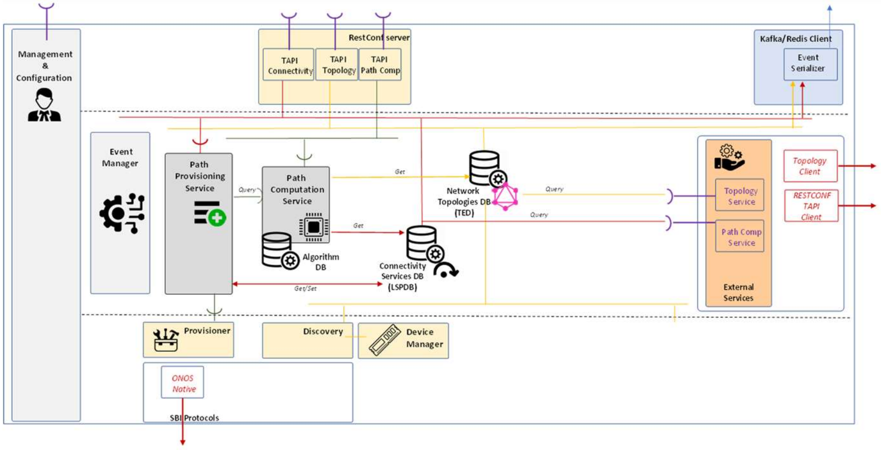
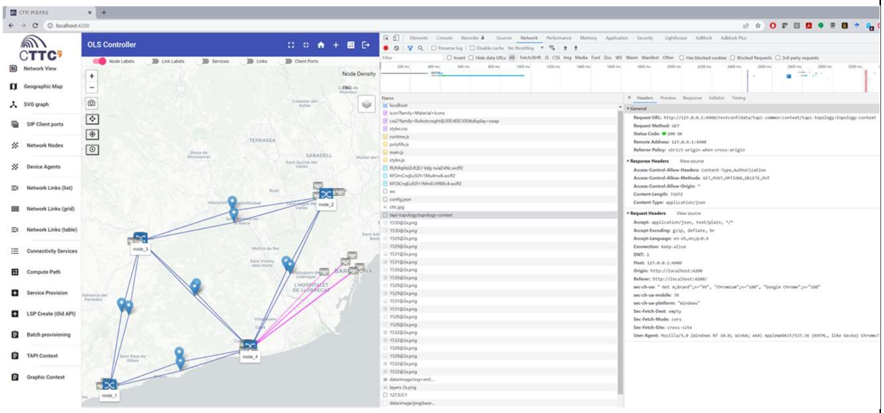
	 e-lighthouse E-lighthouse Network Planner Kubernetes Network view & analysis > IT Orchestators > Kubernetes Show empty tables K8S namespace — Table with one row for each element of the type: *K8S namespace.* — 5 K8S worker node — Table with one row for each element of the type: *K8S worker node.* — 1 K8S service — Table with one row for each element of the type: *K8S service.* Follows a — 4 K8S deployment — Table with one row for each element of the type: *K8S deployment.* — 3 K8S pod — Table with one row for each element of the type: *K8S pod.* Follows a — 9
Component Integrations	B5G-ONP using the exposed SBI allows the task automation with the following modules: - PON SDN Controller allows the network management and automation of this domain using the available tools and platforms via the NBI of the PON SDN Controller - IP SDN Controller supplies all the IP-based information and presents an API to read the state of the resources and provision new IP-related services. - XR SDN Controller provides a centralized point of control for XR network resources in point-to-point or point-to-multipoint connections via the NBI exposed by the unit. - TAPI Optical Network Orchestrator offers a uniform, open and standardised way to obtain information from subsystem and sub-controllers to B5G-ONP according to TAPI 2.1, and the exposed API by the module. - Kubernetes cluster (K8s) connected with B5G-ONP to automate the deployment, scaling, and management of applications to build more efficient and scalable IT infrastructures. The orchestrator, via kube-apiserver, addresses their tasks related to i) service discovery and load balancing, ii) configuration management, iii) security and access control, and iv) K8s monitoring and analytics.
Component KPIs	The KPIs measured are: - Discovery time: The time elapsed from when a network operator sends the discovery command until all layers of the topology are imported into B5G-ONP. This period varies from seconds to several minutes depending on factors such as network size, complexity, and the efficiency of the orchestrator. o Several measurements were taken, and discovery times differed based on the scenario and the controller being integrated. In a simple experiment, the average metrics were as follows: ▪ TAPI Optical Network Orchestrator: 300 ms ▪ IP SDN Controller: 4 seconds ▪ Kubernetes Cluster: 2 seconds

	- Provisioning time: The time from when a change occurs to a network resource or service until it becomes fully operational again. This period starts when B5G-ONP receives an instruction from the network operator or after the analysis task, when the orchestrator performs actions based on the analysis results. The time depends on the complexity of the network and services. o Multiple evaluations were performed. The time required for provisioning varied depending on the controller and the network size. From the project experiments, the following mean baseline values were recorded: ▪ DSR provisioning: 237 ms ▪ OLS path provisioning: 110 ms ▪ IP BGP adjacency provisioning: 4 seconds ▪ Reconfiguring OLS path provisioning: 200-300 ms ▪ Kubernetes Deployment: 150 ms ▪ Kubernetes Service: 100 ms - Optical Path Computation Element (PCE) latency: The time required to compute a path, which is directly proportional to the number of network elements involved and the traffic load on the network. o Multiple evaluations were conducted during the project experiments. The computation time varied significantly based on the number of nodes and connections present in the network. ▪ PCE: average of 200 ms ▪ Multiband-PCE: average of 600 ms, depending directly on the time consumed by the MB-PCE when requesting the optical context.
Status, availability, repository	E-Lighthouse Network Solutions SL proprietary software.
Additional Remarks	List of applicable publications: - Integration with B5G-ONP for multi-domain networks: [Cas24a] - Final demonstrations: [Mor24]

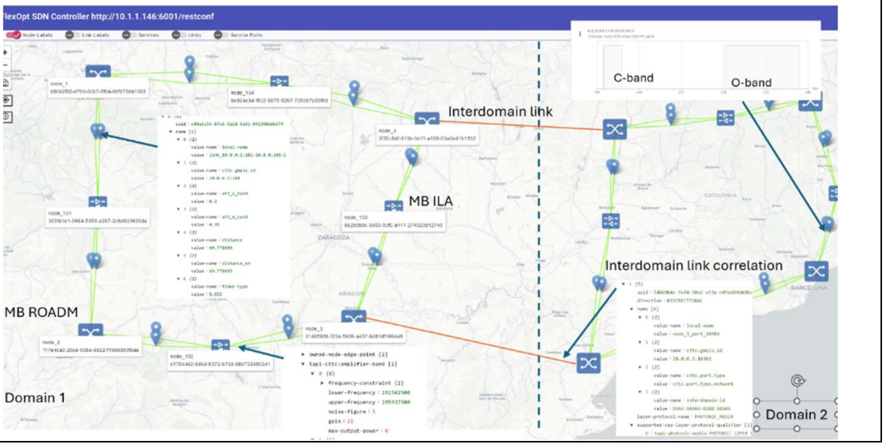
3.2 TAPI-enabled Optical Network Orchestrator with externalized Path Computation (CTTC)

Component Name: TAPI-enabled Optical Network Orchestrator with externalized Path Computation	
Summary	The TAPI-enabled Optical Network Orchestrator is a functional element of the architecture that is responsible for the following functions: i) providing a uniform, open and standard view and interface to the higher levels and components of the B5G-OPEN control, orchestration, and telemetry system; ii) Compose a complete Context to be consumed by B5G-OPEN network planner and additional consumers combining information retrieved from subsystems and sub-controllers (Optical Controller, external databases, monitoring systems, etc), iii) Enable single

	entry point for provisioning DSR and Photonic Media services, including externalized path computation and iv) provide an event telemetry data source that reports events that happen asynchronously in the network.  TAPI for DSR or Media Channel Tracking standards definition and TAPI v2 TAPI Network Orch TAPI Path Computation TAPI Topology Path Computation Server Optical NBI Optical Controller Native Topology OLS Topology TAPI for OCh/Media Channel Based on ONOS OLS driver OLS Controller NETCONF AGENT SONiC 2 3 ROADM R1 ROADM R2 AMP1 AMP2 Packet / Optical Node PointToPoint Connectivity TAPI DSR Connectivity Request Optical Connectivity TAPI Photonic Media Layer (Network Media Channel).
Description and Internal architecture of the component	The core of the TAPI Optical Network Orchestrator controller is an asynchronous event loop. On the one hand, it exports multiple services via its multiple North Bound Interfaces (NBI) to users or clients, using RESTCONF/YANG. The most relevant services are Topology Management, Connectivity Service Management and Path Computation. The RESTCONF server is responsible for processing requests using the RESTCONF protocol. The planned Yang models are a subset of the ONF TAPI v2.1 Requests are mapped to internal structures and processed by functions in the event manager. The controller is a multi-threaded application, written in C++ (C++20). It targets GNU/Linux systems (e.g., Ubuntu 20.04 and later) and can be executed as docker containers. The design is highly modular, so additional functionality can be implemented as shared link libraries that can be configures and loaded on demand.

	 Management & Configuration RestConf server TAPI Connectivity TAPI Topology TAPI Path Comp Kafka/Redis Client Event Serializer Event Manager Path Provisioning Service Query Path Computation Service Get Network Topologies DB (TED) Query Query Topology Client Topology Service RESTCONF TAPI Client Path Comp Service External Services Algorithm DB Get Connectivity Services DB (LSPDB) Get/Set Provisioner Discovery Device Manager ONOS Native SBI Protocols  CTTC OLS Controller Network View Geographic Map SVG graph SIP Client ports Network Nodes Device Agents Network Links (list) Network Links (grid) Network Links (table) Connectivity Services Compute Path Service Provision LSP Create (Old API) Batch provisioning TAPI Context Graphic Context Node Density node_1 node_2 node_3 node_4
Interface Specification	The Interfaces that have been developed and tested are the following • The TAPI Optical Network Orchestrator receives a new connection request with some requirements (source and destination, bandwidth provision, latency constraints, QoT conditions, etc.). The client of this interface is the B5G-ONP and uses TAPI 2.1 for this purpose. • The interface from the TAPI Optical Network Orchestrator to the optical controller is based on the ONOS native interface, extending the existing implementation to support additional requirements and use cases. • The interface from the TAPI Optical Network Orchestrator to path Computation Engine is 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 relies on acting as a REDIS client sending telemetry information following the TAPI

	Reference Implementation Agreement (RIA) for streaming TR-548
Functional Validation	Tests done to validate the component: • Launch the TAPI orchestrator and retrieve the topology in terms of nodes and links and display this information. This test shall be carried out: i) loading the information from a set of JSON files that have previously been retrieved and ii) performing dynamic loading of links and related data from the ONOS instance. • Retrieve the TAPI context from the TAPI orchestrator and validate the topology in terms of nodes and links. Validate that the TAPI context is correct and consistent and can be consumed by: i) B5G-ONP clients as well as ii) Path Computation Elements • Load a topology and validate that the TAPI orchestrator is able to report Telemetry data to the REDIS database that is part of the Telemetry System • Perform an externalized path computation and validate the function using an external PCE with TAPI enabled interface • Perform the provisioning of services See the list of publications showing the integrations and evaluation of the component.
Component Integrations	The TAPI-enabled optical network orchestrator integrates with the following elements: • The B5G-ONP. This functional entity is the main client of the controller. The B5G-ONP performs requests related to service provisioning in the optical network, using TAPI and requesting DSR connectivity services • The Optical Path Computation Element (PCE). The network orchestrator relies on a dedicated system for externalized path computation. For this, it uses extended TAPI interfaces for the purposes of topology discovery and path computation functions. • The Telemetry system to act as a data source for the reporting of events. This means that the TAPI orchestrator sends JSON encoded telemetry data to clients, such as OSS or data visualizers. • The ONOS SDN controller that takes care of provisioning connectivity by means of optical connectivity intents, using a dedicated interface. The interface shall be augmented to support the specification of a computed path (in terms of links as well as frequency ranges for the optical media channel to be used).

Component KPIs	The KPIs to be measured are: - Service Provisioning Latency (< 10 min with hardware and < 30 seconds with emulated hardware). This is the time it takes to provision an optical service. - Service Provisioning Overhead – In terms of messages, message size, encoding, etc. This includes a characterization of the protocol overhead (e.g., HTTP, RESTCONF, etc). - Path Computation Latency – measured as the time it takes to perform a path computation with a dedicated PCE. This is to be evaluated. **Integration with Path Computation** We developed the necessary extensions to the current ONF Transport API v2.1.3 photonic media layer models to support the dynamic provisioning of services of MB-WRN networks exploiting a MB-PCE. The development of these extensions is a challenging task as it requires to provide extensions a larger number of system parameters. The emulated network is BT's optical mesh that consists of 22 ROADM nodes, 56 amplifiers, 28 terminal devices (106 network elements in total) and 238 unidirectional links. For the scope of this experiment, it is assumed that each link may support E, S, C and L bands. The MB-PCE latency for scenario C was measured to be in the range between 1.8 – 2.2 seconds. This parameter value depends on whether the MB-PCE is also tasked to retrieve the network topology as explained above (first request in the series). On the contrary, this latency is two orders of magnitude lower, ranging between 17 ms and 36 ms, when the network status was already up to date (assuming state synchronization). In contrast, for scenarios A and B, the deliberate degradation of the OSNIR rendered a large number of the listed frequency slots as 'unavailable/void', so the RMSA algorithm had to execute the PHY layer validation process thousands of times resulting to considerable delays for the MB-PCE to return results. For these two scenarios A and B, the latency is measured to be between 2.5 and 3.2 seconds. **Integration with B5G-OPEN in multi-domain transparent networks:**  lexOpt SDN Controller http://10.1.1.146:6001/restconf C-band O-band Interdomain link MB ILA Interdomain link correlation MB ROADM Domain 1 Domain 2

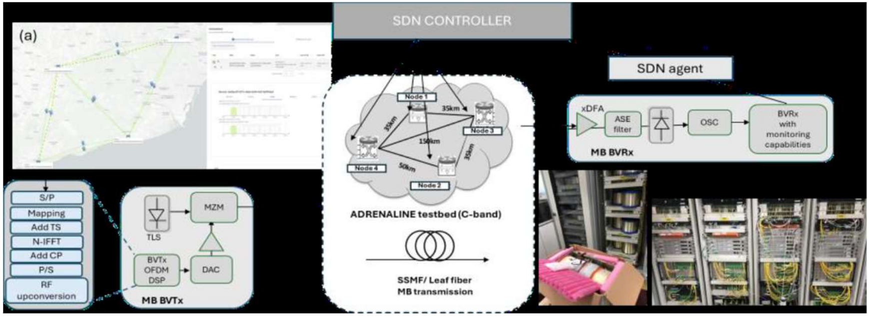
In this scenario, we provision first a service between 2 transceivers in the same domain. The allocated path uses the O-band since the a-posteriori QoT validation (in terms of OSNR, PMD, CD and power levels of the signal) is within the receptor tolerances and a second test is between 2 transceivers that are not in the same domain, and, in this case, the selected band is the C-band. For assessing the control plane latency values coming from the topology discovery, algorithm computation and provisioning phases, we provide in Table 1 averaged values coming from 10 repetitions of the tests. Note that: (i) discovery and provisioning phases are relatively fast, since they operate on an emulated hardware, (ii) path and spectrum computation benefit from an optimized implementation of the algorithms, and a simplified impairments calculation that does not consider in this setup the Raman scattering effects. Note that by having a parallelized behaviour of the requests, overall latency is minimized with regards to the serialized setup and values from ENP are affected by Internet latency.

	Measured at VPN / OLS (milliseconds)	Mean value ENP (milliseconds)
Discovery context	280 (empty) 492 (services)	931
Single-domain provision	2.5 (O-band)	309
Multi-domain provision	56 (C-band) parallel	515

B5G-OPEN Demo

In each domain, the optical path set-up provisioning time is less than 1 second. Moreover, PCE latency is measured to be in the range between 0.5s and 0.6s and it is due to: a) the time needed to retrieve network topology and; b) the time needed for the PLI-aware RSA algorithm to return the selected path, band, channel frequency assignment, and optimal launch powers. The IP/BGP and pluggable configuration requires less than 4 seconds

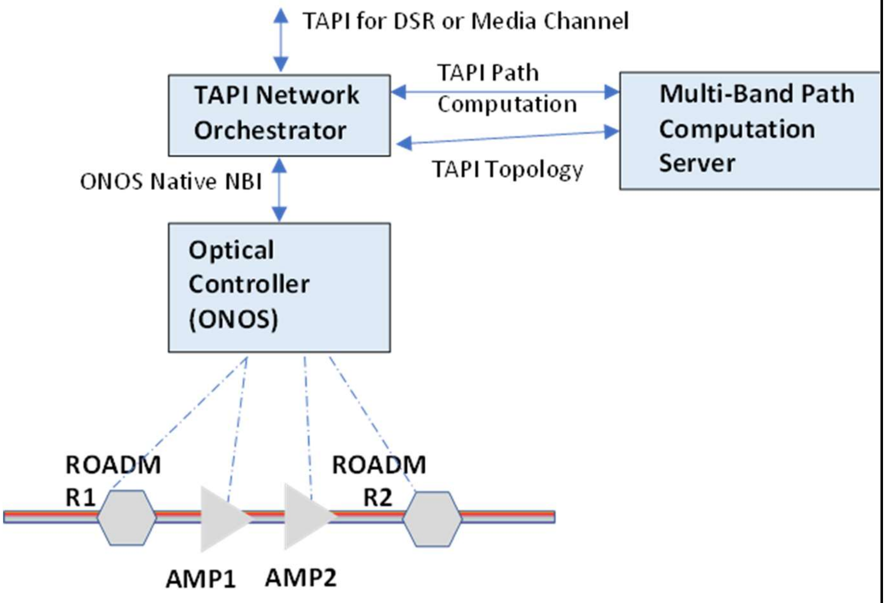
Integration with Nokia Chromatic Dispersion based algorithms



Status, availability, repository	CTTC Proprietary software
Additional Remarks	List of applicable publications: - Integration with Path Computation Elements: [Kos23] - Integration with S-BVT Open Config agents: [Cas24b]

	- Integration with B5G-ONP for multi-domain networks: [Cas24a] - Final demonstrations: [Mor24] - Final demonstrations: [Boi24]

3.3 Path Computation Elements – MB-PCE – (OLC-E)

Component Name: Multi-Band Path Computation Engine (MB-PCE)	
Summary	The Multi-Band Path Computation Engine (MB-PCE) is based on a multi-band routing engine which ensures that: i) routing is implemented by means of an efficient spectrum and modulation-format assignment; and ii) the impact of physical layer effects over the selected optical paths is estimated and the results are benchmarked against QoT target values (BER, OSNIR, OSNR, etc). In this way, the planning tool ascertains the conditions that maximize the total capacity of the network while it minimizes the global blocking probability and prevents network misconfiguration.  TAPI for DSR or Media Channel TAPI Network Orchestrator TAPI Path Computation Multi-Band Path Computation Server TAPI Topology ONOS Native NBI Optical Controller (ONOS) ROADM R1 ROADM R2 AMP1 AMP2
Description and Internal architecture of the component	The MB-PCE functionality is realised in three stages as follows: STAGE-I: Network Topology Implementation: the network topology is defined by setting the connectivity pattern between the nodes and the traffic matrix. Next, the k-shortest paths for all network node pairs are derived. More specifically, in this step, the following quantities are defined: the network topology including nodes, edges and amplifiers, the available optical bands, the capacity per band, the traffic matrix, the average time duration of the demands and the average inter-arrival time between two consecutive demands, as well as the available line-rates and their distribution on the demands.

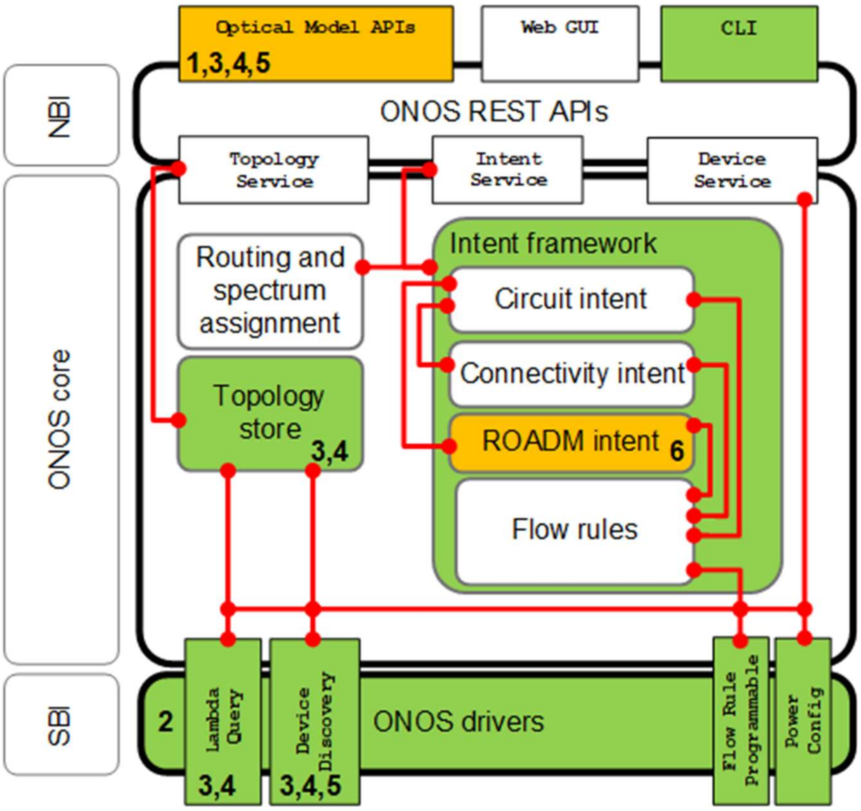
	STAGE–II: Spectral and Modulation Assignment (SMA) and PL entanglement: the operation is completed in two steps: In the first step, i) a preliminary spectrum and modulation format assignment (SMA) is made for a number of the k-shortest paths, and ii) the Optical Signal to Noise plus Interference Ratio (OSNIR) for these shorter paths is estimated taking into account the impact of the physical layer effects by means of closed-form expressions. In the second step, the Optical Multi-band Physical Layer Aware Routing Modulation and Spectral Assignment (OMB-PLA-RMSA) algorithm either selects or rejects a lightpath. A path is rejected if a) no contiguous spectral slots are available in any optical band to support the end-to-end connection, b) either the OSNIR of the candidate lightpath falls short of the QoT estimator threshold or the OSNIR of at least one of the already established lightpaths would perform below the QoT threshold due to the presence of this candidate lightpath. In either (a), (b) cases, the rejected lightpath is assigned the next available path from the sorted list of k-shortest paths and it is then re-iterated. If these paths are all rejected, the first step is repeated using a lower cardinality SMA values. If no path is retained, the engine registers a blocking condition. STAGE–III: Path Allocation: This is the stage where the lightpaths are established in the network. The final assessment on network's throughput is completed and a lightpath is successfully set if contiguous spectral slots are available over the end-to-end transparent path with acceptable physical layer performance (above the QoT estimator threshold). The successful establishment of a lightpath triggers the update of the corresponding arrays for each link of the path, e. g., arrays of power, modulation format, consumed frequency slots.
Interface Specification	Interfaces developed and tested: - The MB-PCE uses two interfaces to communicate with the TAPI Optical Network Orchestrator - The first interface is based on TAPI v2.1 and it is exposed by the TAPI Optical Network Orchestrator. MB-PCE uses this interface in order to retrieve the current optical network topology and status. - The second interface is exposed by the MB-PCE. It is again based on TAPI v2.1 and it is used by the TAPI Orchestrator. The TAPI Orchestrator request a path computation from the MB-PCE. The MB-PCE analyses the request and computes the optimum path and send this information back to the TAPI Orchestrator.
Functional Validation	Tests that can be done to validate the component: - Launch the MB-PCE and retrieve the topology from TAPI Optical Network Orchestrator in terms of nodes and links and display this information. This test shall be carried out: i) loading the information from a set of JSON files that have previously been

	retrieved and ii) performing dynamic loading of links and related data from the TAPI Optical Network Orchestrator. - Retrieve a new service request from TAPI Optical Network Orchestrator and validate that: a) the path is properly estimated; b) the correct band is selected; c) the number of frequency allocations units are correctly assigned; d) the correct frequencies are assigned to the service; e) the correct message is generated and send back to TAPI Optical Network Orchestrator.
Component Integrations	The MB-PCE integrates with the following elements: - The MB-PCE communicates with the TAPI Optical Network Orchestrator to realise two functionalities: a) Retrieve optical network topology; b) Path computation for a new service request. - Initially, MB-PCE communicates with the TAPI Optical Network Orchestrator using the TAPI v2.1 interface to retrieve the current optical network topology context and status. - Then, MB-PCE receives a new service request from the TAPI Optical Network Orchestrator using the TAPI v2.1 interface. The MB-PCE compute the optimum path and send this information back to the TAPI Optical Network Orchestrator,
Component KPIs	The KPIs measured are: **1. Path Computation Latency** This is the time it takes to MB-PCE to compute the path of a new service request, measured from the timestamps between request and response with a target value of < 40s. • Several scenarios were executed. The MB-PCE Path Computation Latency was measured to be in the range between 1.8 – 3.2 seconds **2. TAPI Topology retrieval and parsing Latency** This is the time it takes to MB-PCE to retrieve and parse the optical network topology context (described using the TAPI format), measured from the timestamps between request and response, with a target value of < 20s. • Several scenarios were executed. The MB-PCE Topology retrieval and parsing Latency was measured to be in the range between 0.9 – 2.5 seconds
Status, availability, repository	OLC-E Proprietary software. The pseudocode of the algorithms together with a thorough description has been published in [Kos23] and D4.2.

Additional Remarks	Some results were published in [Kos24:] Measurements were made of the optical SDN controller and the MB-PCE, connected via tunnels over the public Internet. The emulated network consisted of 22 ROADM nodes, 56 amplifiers, 28 terminal devices (106 network elements in total) and 238 unidirectional links assumed to support the E, S, C and L bands. Three scenarios were presented, with the routing engine of the MB-PCE instructed to check the potential of the available bands to support service connection requests, searching over multiple transmission bands in the case of misconfigurations that could be either due to an erroneous data exchange between the MB-PCE elements or even the result of deliberate/malicious acts. The first scenario investigated the impact of a deliberate or accidental physical layer misconfiguration instruction on the completion (or not) of service requests in the case of fully transparent paths. In the second scenario was similar but the deliberate misconfiguration of the PHY parameters was less dramatic, while in the third scenario the parameters were within acceptable but suboptimal ranges. In the more realistic third scenario, the MB-PCE latency was measured to be in the range between 1.8 – 2.2 seconds. In contrast, for the other two scenarios with deliberate misconfigurations, the RMSA algorithm had to execute the PHY layer validation process thousands of times and the latency was higher, measured to be between 2.5 and 3.2 seconds.

3.4 OPTICAL CONTROLLER (CNIT)

Component Name: ONOS optical controller	
Summary	The optical controller is based on ONOS SDN controller that provides a wide environment that is used to control and configure optical devices and transceiver equipped within packet/optical white boxes. In particular, the main roles of the optical controller are: (i) retrieve devices description from data plane and abstract them toward the upper control layers; (ii) receive the service configuration requests by the upper control layers and translate such requests in a set of configuration messages to be forwarded to each involved device.
Description and Internal architecture of the component	The 3.0 version of ONOS that we considered at the beginning of the project already provides a rich NBI based on REST APIs and, on its SBI it is already able to connect to a variety of packet-based and optical devices (e.g., exploiting NETCONF protocol). The ONOS core already implements the basic connectivity services using the concept of intent that simplify and automate the service management (e.g., in case of network failure). During the project, several developments within the ONOS controller have been implemented at different levels of the ONOS architecture (i.e., in the NBI, in the SBI and in the Core) for introducing B5G-OPEN specific features: 1. Enable integration with T-API orchestrator (NBI)

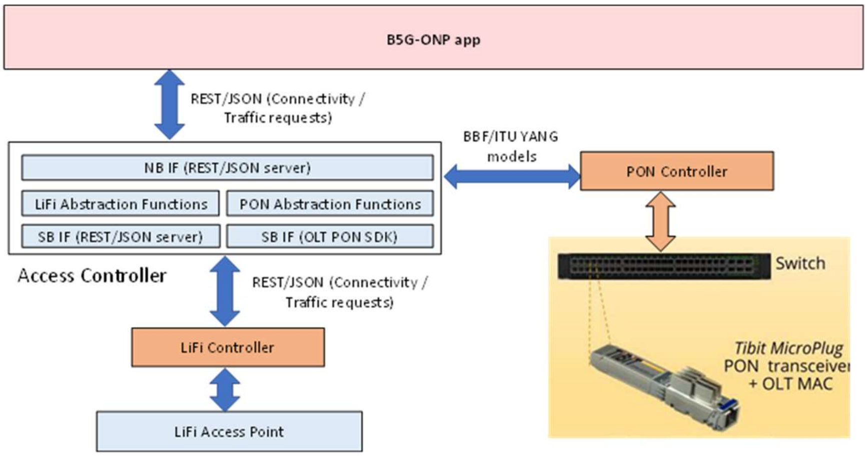
	2. Develop drivers toward new devices and update existing drivers against most recent versions of standard models (SBI) 3. Introduce the support of flexible grid (NBI, core, SBI) 4. Introduce the support of multi-band (NBI, core, SBI) 5. Import/Export physical impairment device manifest (SBI, NBI) 6. Activate intents using as end-points the ROADM's ports (Core) In the figure below the aforementioned development (from 1 to 6) targets are mapped within the ONOS architecture, where each number is reported in the affected blocks. Blocks reported in white are not modified during the project, block reported in green have been upgraded during the project, while blocks reported in orange have been created from scratch during the project.  Optical Model APIs 1,3,4,5 Web GUI CLI NBI ONOS REST APIs Topology Service Intent Service Device Service ONOS core Routing and spectrum assignment Intent framework Circuit intent Connectivity intent Topology store 3,4 ROADM intent 6 Flow rules SBI 2 Lambda Query 3,4 Device Discovery 3,4,5 ONOS drivers Flow Rule Programmable Power Config
Interface Specification	The ONOS optical controller integrates with the following elements: - The TAPI-enabled optical network orchestrator. This entity will use REST APIs. 1) It will use POST (and DELETE) calls to perform requests related to service provisioning (and deletion) in the optical network. 2) it will use GET calls to retrieve information regarding network topology and details regarding links and devices. - The optical devices (i.e., packet-optical nodes, transponders, ROADM and OLS). Such devices expose to the controller a YANG models and provide a NETCONF server. The ONOS controller

	uses a NETCONF client to retrieve information and configure the devices.
Functional Validation	The component has been utilized in several of the experimental demonstrations conducted during the project (e.g., in TIM and HHI) moreover it has been experimentally demonstrated in international conferences (e.g., [Gio23]). In these contexts, the following functional validations have been conducted multiple times and testified the reliability of the developed software. • Network discovery (mainly involve SBI), emulated environment. Launch the ONOS controller, including required applications and drivers. Post an emulated network topology including devices and links. Verify that all devices are correctly discovered, including interfaces and augmented details (e.g., related to physical impairments). • Network abstraction (mainly involve NBI), emulated environment. After network initialization verify that all the acquired information regarding devices is correctly exported in the REST APIs toward upper layers. • Service provisioning and releasement (involve NBI, core and SBI), emulated environment. Receive an intent request from the TAPI orchestrator (for several types of intent). Verify that the intent is correctly installed and that all involved devices are correctly configured. Cancel the configurations when an intent deletion request is received. • Device tests (mainly involve SBI), real devices. Push a specific device, test connectivity, device discovery and the ability to properly discover all the device details.
Component Integrations	Other components this component is integrated with: • The TAPI-enabled optical network orchestrator. • OpenROADM agent (ROADM by TIM) • OpenConfig agent (Transponder by CTTC) • OpenConfig agent (Pluggables in SONiC devices by CNIT)
Component KPIs	Set of component KPIs that have be measured during experimental demonstrations: • Time required for network discovery: o Measured in the order of tens of seconds, always lower than one minute. • Physical activation delay for an optical intent: o Measured in the order of one-two minutes mostly depending on the utilized transceivers. • Time elapsed in the controller: o Measured in the order of one second.

	• Time elapsed for configuration of devices: ○ Devices typically confirm the reception of a configuration message in the order of few hundreds of milliseconds.
Status, availability, repository	ONOS version 3.0 has been forked at the beginning of the project. A version with all the software contributions developed during B5G-OPEN project (and all scripts utilized during experimental demonstrations) is currently available in the following public repository: https://github.com/Network-And-Services/onos-b5g-open
Additional Remarks	Selected components the developed software have been contributed to the open-source ONOS community and are now part of the official ONOS distribution: https://gerrit.onosproject.org/c/onos/+/25681 https://gerrit.onosproject.org/c/onos/+/25616 https://gerrit.onosproject.org/c/onos/+/25596 https://gerrit.onosproject.org/c/onos/+/25594 https://gerrit.onosproject.org/c/onos/+/25593 https://gerrit.onosproject.org/c/onos/+/25168

3.5 Access Controller / PON Controller (OLC-E)

Component Name: Access Controller / PON Controller	
Summary	The B5G-OPEN TDM-PON infrastructure is realised using an XGS-PON OLT pluggable transceiver (e.g., TiBit pluggable) and a couple of pluggable ONUs (e.g., Tibit ONUs). The OLT is interfaced directly to a whitebox switch while the OLT is interconnected to the ONUs by means of splitters, forming up an ODN branch. The PON vendor (Tibit) will provide the pluggable software and the PON controller software. The integration of Tibit PON Controller with the B5G OPEN platform is realised with the development of an Access Controller as illustrated in the below figure. The Access Controller is responsible to: a) monitor the PON network and receive any requests for PON reconfiguration; b) translate these requests into high level traffic requests that is reported to the B5G-ONP App; c) execute the appropriate actions in the PON Controller in order to support the new requests. In addition, the Access Controller will communicate with the LiFi Controller for retrieving any connection/traffic requests.

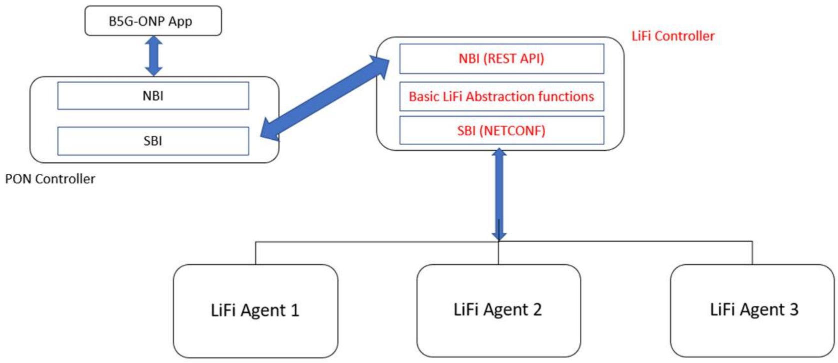
	 B5G-ONP app REST/JSON (Connectivity / Traffic requests) BBF/ITU YANG models NB IF (REST/JSON server) LiFi Abstraction Functions PON Abstraction Functions SB IF (REST/JSON server) SB IF (OLT PON SDK) Access Controller PON Controller Switch REST/JSON (Connectivity / Traffic requests) LiFi Controller Tibit MicroPlug PON transceiver + OLT MAC LiFi Access Point
Description and Internal architecture of the component	The Access Controller is developed as part of the B5G-OPEN software platform, and it will provide the below functionalities: • On the South Bound Interface (SBI), the Access Controller will communicate with the vendor specific PON Controller using a subset of the BBF/ITU YANG models. The SBI that communicates with the PON Controller is a software client that is developed based on OLT PON SDK. • On the South Bound Interface (SBI), the Access Controller will communicate with the LiFi Controller using REST/JSON for receiving any connectivity/traffic requests generated in the LiFi network. • The Access Controller implements a set of: a) PON abstraction functions which are responsible to extract the PON parameters and their values; b) LiFi abstraction functions for extracting LiFi traffic parameters. In addition, the PON and LiFi abstraction functions will expose to the higher layers only the valuable for the B5G-OPEN software platform set of parameters. • On the Northbound Interface (NBI), the Access Controller communicates with the B5G-ONP app. The NBI implements a REST/JSON server which will support the exchange of traffic related information adopting a data structure defined in B5G-OPEN project.
Interface Specification	Interfaces developed and tested: - The Access Controller communicates with the PON Controller (Tibit) using the OLT PON SDK. - The Access Controller communicates with the LiFi Controller using REST/JSON - The Access Controller exposed one interface toward the B5G-ONP app component. The interface is realised using REST/JSON.

Functional Validation	Tests that can be done to validate the component: - Deploy the PON network including the XGS-PON OLT pluggable transceiver and a couple of pluggable ONUs. Launch the PON Controller. Then launch the Access Controller and test that: a) the Access Controller communicates successfully with the PON Controller; b) the Access Controller retrieves the PON configuration information. - Deploy the PON network and launch both the PON Controller and LiFi Controller. Then launch the Access Controller. Then LiFi Controller will generate a new traffic request and send to Access Controller. Then test that: a) the Access Controller receives successfully the request and parse all its data; b) translate the new request into a high level traffic request; c) deliver the high level traffic request to the B5G-ONP app using the NBI; d) execute the appropriate reconfiguration actions in the PON Controller; e) observe that the new PON configuration is realised in the testbed.
Component Integrations	The Access Controller integrates with the following elements: - The vendor specific (Tibit) PON Controller using the SBI. Communication realized using OLT PON SDK. - The LiFi Controller using the SBI. Communication using REST/JSON. - The B5G-ONP app using the NBI. Communication using REST/JSON.
Component KPIs	The KPIs measured are: **1. PON Reconfiguration Latency** This is the time it takes for the actual reconfiguration of the PON network, Measured as time difference between the request timestamp and response timestamp, with a target value of < 20s. Initially, we measured latencies for retrieving the configuration and status information from different elements in the PON network. In detail: • Login: 160 – 200 ms • Retrieving Controller configuration: 206 - 290 ms • Retrieving OLTs configuration: 160 – 220 ms • Retrieving ONUs configuration: 130 - 220 ms • Retrieving configuration of selected OLT: 160 - 190 ms • Retrieving configuration of selected OLT: 123 - 180 ms • Retrieving SLAs: 130 - 310 ms • Retrieving configuration of selected SLA: 150 - 204 ms In addition, the PON Reconfiguration Latency is measured in terms of reconfiguring the SLAs that defines the OLT upstream and downlink bandwidth profiles:

	• PON Reconfiguration Latency (SLA update and application): 450 - 600 ms All the measured values are below 1s, therefore far lower than the target of 20s. **2. Access Controller Latencies** This is the time it takes to Access Controller to execute different functionalities including: a) to receive a new request and parse all its data; b) to translate the new request into a high level traffic request; c) to deliver the high level traffic request to the B5G-ONP app; d) to execute the appropriate actions in the PON Controller. The latency is measured from the timestamps between request and response or timestamps between starting and completing a specific task. Here are the values reported in the Berlin Demo (WP5): • Authentication: 234 ms – 790 ms • PON SLA creation: 187 ms – 781 ms • ONU SLA configuration: 316 ms – 529 ms • Logout: 71 ms – 514 ms • Send access traffic descriptor to B5G-ONP app: 51 ms – 79 ms • Response from B5G-ONP app: 53 ms – 7260 ms
Status, availability, repository	OLC-E Proprietary software
Additional Remarks	Details on the access controller latencies will be presented in D5.2 "Final experimental B5G-OPEN validation."

3.6 LiFi controller (PLF)

Component Name: LiFi Controller	
Summary	The LiFi controller for managing LiFi APs with LiFi agents is a simple controller that allows for device discovery and configurations such as accessing SSIDs, IP addresses, and enabling/disabling APs. It is positioned between the PON controller and the LiFi agents, and it is designed to be lightweight and efficient with minimal processing requirements.
Description and Internal architecture of the component	The LiFi controller communicates with the LiFi agents on the LiFi APs using NETCONF protocol, allowing for centralized management and control of the network. The controller includes a REST API for programmatic control and integration with the PON controller. The LiFi controller would be capable of the following functions: • Network Topology Discovery: The LiFi controller is able to discover the topology of the network, including all devices and links between them. • Network Configuration Management: the LiFi controller is able to configure the LiFi APs via the LiFi agents, by sending commands

	through the SBI. The configurations supported include accessing/modifying SSIDs, IP addresses, enabling/disabling APs. - Network Monitoring: the LiFi controller is able to monitor the throughput and latency of the LiFi APs. Network Automation: The LiFi controller is able to automate network operations, such as provisioning or configuration, to reduce manual effort and improve efficiency.  B5G-ONP App NBI SBI PON Controller LiFi Controller NBI (REST API) Basic LiFi Abstraction functions SBI (NETCONF) LiFi Agent 1 LiFi Agent 2 LiFi Agent 3
Interface Specification	The LiFi controller uses two main interfaces for communication with the PON controller and the LiFi agents: - REST API (NBI): The REST API provides a simple and standardized way for external applications and systems to interact with the LiFi controller. The REST API uses HTTP/HTTPS as its transport protocol and supports a range of operations, including GET, PUT, POST, and DELETE. Using the REST API, external applications can retrieve information from the controller, configure the LiFi APs, and monitor the status of the LiFi network. - NETCONF (SBI): The NETCONF protocol provides a standardized way for the LiFi controller to communicate with the LiFi agents. NETCONF uses XML-based messages over SSH or TLS to perform operations such as configuration, monitoring, and software management. The use of NETCONF as the SBI ensures that the SDN controller can communicate with the SDN agents in a secure and reliable manner. Currently the configurations to be supported including the access and modifications on the SSIDs, IP addresses, as well as enabling/disabling the APs. By using both REST API and NETCONF, the LiFi controller can provide a flexible and scalable solution for managing the LiFi APs with LiFi agents. The REST API enables easy integration with external systems and applications, while NETCONF provides a robust and standardized interface for communication with the LiFi agents.
Functional Validation	Several tests can be performed to validate the functionality and performance, which include: Basic functionality test: this test ensures the LiFi controller is able to performance the basic functions such as discovering all connected LiFi agents, accessing/modifying SSIDs, IP addresses, and enabling/disabling APs, etc. Integration test: This test ensures the LiFi controller is able to integrate with PON controller via the REST API.

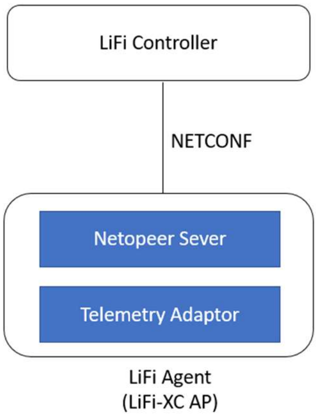
	Scalability test: This test evaluates the ability of the controller to handle multiple LiFi APs with LiFi agents. By performing these tests, the LiFi controller can be validated for its functionality, performance, and reliability, ensuring that it meets the requirements for managing LiFi APs with SDN agents
Component Integrations	The LiFi controller has been installed in the VM provided by HHI. It has been tested with several LiFi APs with LiFi agent implemented. On the other side, the LiFi controller communicates with the Access controller.
Component KPIs	Some key indicators showing in the functional validation: Device Management Accuracy: Upon LiFi switched on, it should be discovered and recognised by the controller. And the controller should be able to manage devices correctly. This KPI assesses how accurately the controller works and manages device states. No failure was observed in normal working conditions. Scalability: Evaluate the controller's ability to scale by managing an increasing number of LiFi devices without performance degradation. It has been tested for up to three devices in lab condition. Latency: the latencies that the controller responds to user's request. This is measured between 10~70 ms, excluding any NETCONF sessions accessing the LiFi agent.
Status, availability, repository	Status: The controller has been implemented, applied and tested in the project. Availability: The LiFi controller is based on ONOS which is available to public. This controller has not been made publicly available yet as it is designed for managing the LiFi agent which is product specific.
Additional Remarks	No additional remarks

3.7 LiFi agent (PLF)

Component Name: LiFi Agent	
Summary	The LiFi agent for managing LiFi APs is a software component that runs on each AP and communicates with the LiFi controller via the Netconf protocol. The agent is responsible for managing the network configuration of the AP, including SSIDs, IP addresses, and other basic parameters.
Description and Internal architecture of the component	The LiFi agent for LiFi APs provides a standardized and programmable interface for managing network configurations, allowing network operators to automate the configuration and management of the LiFi networks. The LiFi AP consists embedded Linux system and it has been implemented for network configuration management based on the

<table>
<tr><td></td><td>

NETCONF protocol by using *sysrepo*, *netopeer2-server*, *netopeer-cli* and *sysrepo-plugin*.



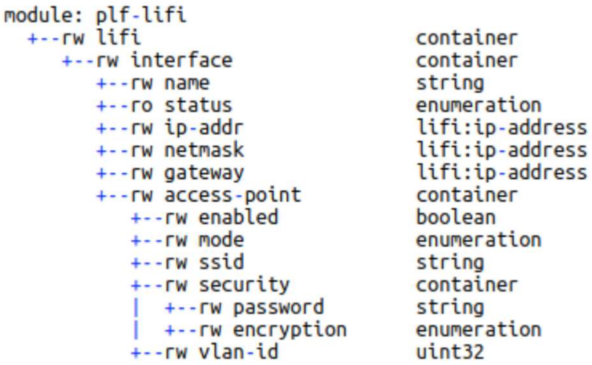
</td></tr>
<tr><td>

Interface Specification

</td><td>

The NETCONF protocol was used which provides a standardized way for the LiFi agents to communicate with the LiFi controller. NETCONF uses XML-based messages over SSH or TLS to perform operations such as configuration, monitoring, and software management.

A LiFi specific YANG model has been implemented to manage the LiFi APs. The LiFi YANG model is called plf-lifi and is illustrated in figure below.



```
module: plf-lifi
  +--rw lifi                        container
     +--rw interface                container
        +--rw name                  string
        +--ro status                enumeration
        +--rw ip-addr               lifi:ip-address
        +--rw netmask               lifi:ip-address
        +--rw gateway               lifi:ip-address
        +--rw access-point          container
           +--rw enabled            boolean
           +--rw mode               enumeration
           +--rw ssid               string
           +--rw security           container
           |  +--rw password        string
           |  +--rw encryption      enumeration
           +--rw vlan-id            uint32
```

</td></tr>
<tr><td>

Functional Validation

</td><td>

The LiFi agent has been validated on completing the designed functions together with the implemented LiFi controller, including:

- Retrieve device information
- Modify device information
- Remove device information
- Collect and prepare device telemetry data for transferring to LiFi controller and the Access controller.
- Scalability Tests: These tests validate that multiple LiFi APs with multiple LiFi agents are able to work with one LiFi controller, without experiencing performance issues or errors.

</td></tr>
<tr><td>

Component Integrations

</td><td>

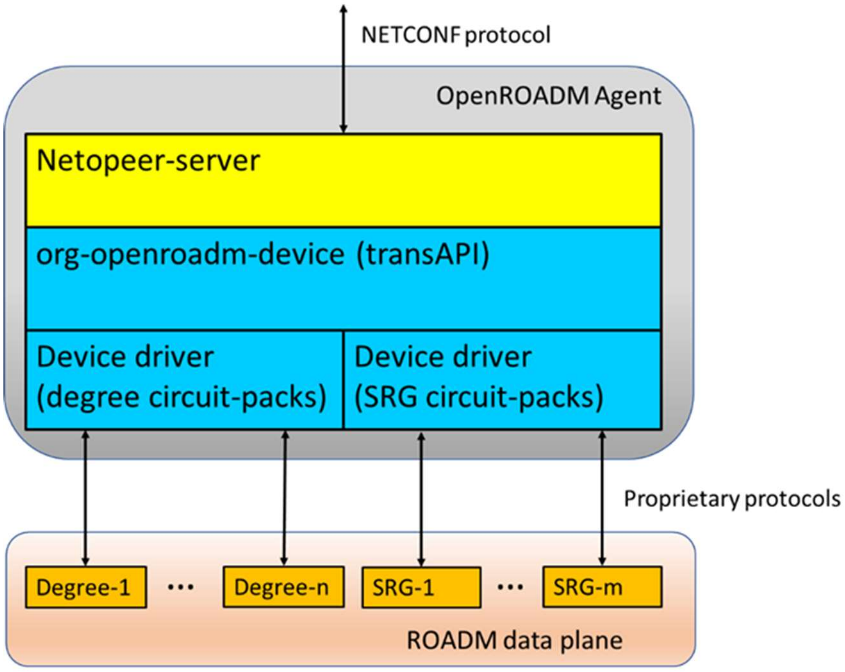
The LiFi Agent has been applied to multiple LiFi APs and communicates with the LiFi controller via NETCONF interface.

</td></tr>
</table>

Component KPIs	Several KPIs for validating the developed agent: • Device Recognition Success Rate: Measure how often the agent successfully recognizes and interfaces with devices. No failure was observed on recognising the LiFi APs. • Configuration Accuracy: Evaluate how accurately the agent applies the configuration changes to the devices and no failure observed. • Stability: while applying the agent in multiple LiFi AP devices, evaluate how reliable and stable the agent is. It has been tested with up to three LiFi APs in lab condition. • Latency: The latencies for accessing the LiFi AP is measured between 5~20 ms, while the latency for having a NETCONF session is typically 30~71 ms. In worst condition that the light communication path is partially blocked, 150 ms has been observed.
Status, availability, repository	Status: The agent has been implemented, applied and tested in the project. Availability: The agent will not be public available. It is an enhancement to current LiFi product and will be applied to the APs for NETCONF support.
Additional Remarks	No additional remarks

3.8 OpenROADM Agent (TIM)

Component Name: TIM OpenROADM Agent	
Summary	The TIM OpenROADM agent is an implementation of a NETCONF server controlling optical network elements using OpenROADM device models. It's basically an evolution of the agent developed for the H2020 Metro-Haul project enhanced to cover MultiBand technology exploiting the latest OpenROADM models.
Description and Internal architecture of the component	The OpenROADM agent exploits the transAPI framework available for Netopeer, an open source implementation of the NETCONF protocol. The transAPI allows invoking call-back functions whenever an edit-config rpc operation performs changes on a specific branch of the configuration. Starting from this feature, the agent implements call-back functions that manage the controller requests for the creation of the interfaces that, according to the OpenROADM device model, are required for connection and optical channel setup. To decouple the OpenROADM model processing from the action required by the underlying hardware, the agent architecture leverages on the Linux dynamic libraries subsystem to load specific drivers at runtime. The drivers are associated to circuit-packs, an OpenROADM entity used to model atomic elements inside a device that, according to the model, must have a type attribute. For the agent, every `circuit-pack-type` can have its specific driver that is loaded by the main module

	when an `edit-config rpc` creates the first circuit-pack of that type. The following figure shows the agent architecture applied to a ROADM architecture.  NETCONF protocol OpenROADM Agent Netopeer-server org-openroadm-device (transAPI) Device driver (degree circuit-packs) Device driver (SRG circuit-packs) Proprietary protocols Degree-1 … Degree-n SRG-1 … SRG-m ROADM data plane

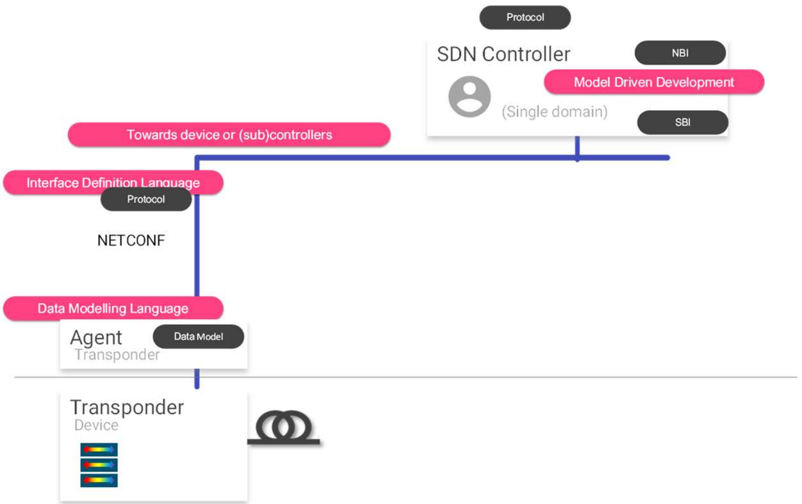
Interface Specification	The OpenROADM agent has two different interfaces: a NorthBound interface towards an SDN controller and some SouthBound interfaces towards the data plane devices.
	
	The NBI is a NETCONF/YANG interface implementing the OpenROADM device models. For the B5G-OPEN project, the agent has been upgraded to the 12.1 device model in order to incorporate all the latest enhancements dedicated to the multiband technology.
	The YANG models that are involved (directly or because imported by other models) are:
	`org-openroadm-device.yang`
	`org-openroadm-network-media-channel-interfaces.yang`
	`org-openroadm-media-channel-interfaces.yang`
	`org-openroadm-prot-otn-linear-aps.yang`
	`org-openroadm-port-capability.yang`
	`org-openroadm-rstp.yang`
	`org-openroadm-otn-odu-interfaces.yang`
	`org-openroadm-otn-otu-interfaces.yang`
	`org-openroadm-optical-transport-interfaces.yang`
	`org-openroadm-lldp.yang`
	`org-openroadm-ethernet-interfaces.yang`
	
	The SBI is a proprietary interface based on function calls. A driver module implements functions to perform actions on the circuit-packs composing the device. For example, a ROADM degree can be composed of WSSes and amplifiers modelled as programmable circuit-packs and a dedicated driver can be written for configuring them. The functions that a driver can implement are the:

	• **init**: called during agent start-up to set up the internal communication session between the agent and the circuit-pack and to perform initial circuit-pack setup. • **close**: called at agent closing to free all the allocated resources. • **get_inventory**: called when the agent needs circuit-pack inventory information. • **get_port_operational_state**: called to get the operational state of a port. • **get_port_mc_capabilities**: called by the agent to retrieve the optical capabilities (in terms of supported bands) of a device's port. • **make_connection**: cross-connection (spectral window) creation between circuit-pack ports. • **delete_connection**: cross-connection removal. To cope with circuit packs that are not programmable (such as mux-demux), all functions are optional, but the followings constraints apply: • If *init* function is defined the *close* function must be present. • If *make_connection* is defined the *init* function must be present. • If *make_connection* is defined the *delete_connection* function must be present. It's worth specifying that the agent software architecture is flexible enough to manage devices composed of circuit-packs from different vendors and allows easy implementations of emulators, since it is possible to create "dummy" drivers that perform no actions. This feature has been exploited to control TU/e hardware prototypes.
Functional Validation	The tests that can be done to validate the OpenROADM agent are the following: • Startup of the netopeer-server daemon and loading of the transAPI module implementing the agent, of the OpenROADM device models and of the initial configuration data. After loading the configuration data, the agent must load all the configured device drivers (Loadable Linux libraries) and identify the implemented API functions. • Using a NETCONF client (e.g. netopeer-cli), retrieval of configuration and state data, focusing on the optical parameters, in terms of supported spectral windows (media channel capabilities). • Edit-config operation to create cross-connections on the different bands of a multi-band ROADM • Edit-config operation to setup the transmission wavelength on a ZR/ZR+ pluggable module
Component Integrations	The OpenROADM agent has been integrated with the ONOS SDN Controller for the control of multiband ROADMs.

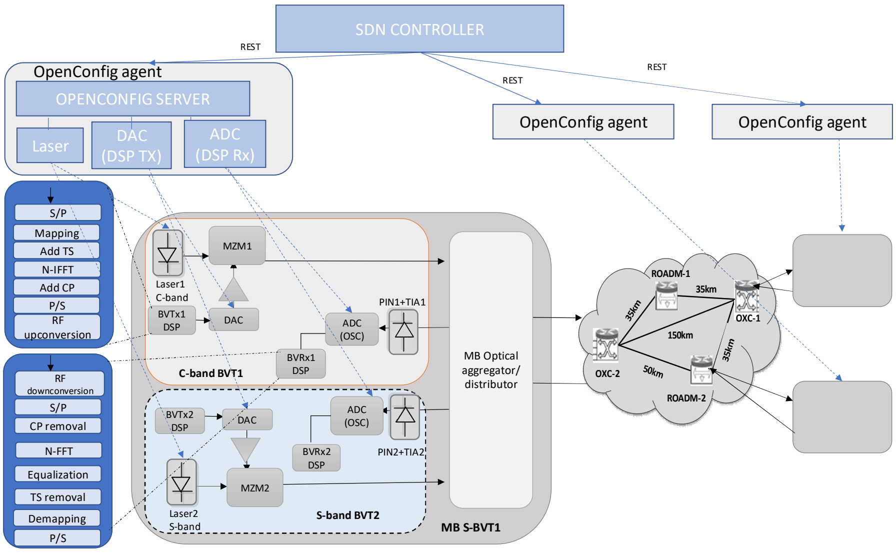
Component KPIs	The kind of KPI that can be applied to the agent are related to delays and latencies. The agent architecture allows it to be employed with different data-plane hardware to implement basic OpenROADM based ROADMs and transponders. This makes it difficult to define a set of measurements that can characterize all kind of devices that can be implemented. Moreover, most of the time required to perform different actions is spent by the specific data plane hardware and it results to be several orders of magnitude slower than the time taken by the agent for NETCONF messages processing (tenth of seconds wrt tenth of milliseconds). Therefore, it's clear that it would be more meaningful to characterize a device employing the agent. That said and keeping in mind that delays involving a software module are negligible wrt delays of optical resources, it can be meaningful to characterize the agent as a standalone component (i.e. without underlying data-plane hardware) from a scalability point of view. The following KPIs are defined: • Changes of loading time of the startup datastore (at agent initialization) wrt the number of modelled circuit-packs, i.e. wrt to the datastore dimensions. For example, in the ROADM case how startup times changes increasing the degrees number (2, 4, 9 to cover the most typical ROADMs sizes). • Time required to create roadm-connections (express or add-drop) as a function of the number of degrees (2, 4, 9), i.e. of the datastore dimensions. Such tests have been documented in deliverable D4.2.
Status, availability, repository	The agent is publicly available as a docker container. Access is granted on-demand, aiming at providing further collaboration opportunities.
Additional Remarks	No remarks.

3.9 OpenConfig Agent (CTTC and CNIT)

Component Name: CTTC OpenConfig agent, CNIT OpenConfig agent	
Summary	B5G-OPEN has produced two different implementations of the OpenConfig agent. One implemented at CTTC with the main aim of integrating it with the CTTC developed multi-band transceiver and one implemented at CNIT with the main aim of integrating it into the packet-optical SONiC based node. Since the software architecture, interfaces, and proposed KPIs are the same, both implementations are described in this section. OpenConfig agent is an implementation of an SDN agent using NETCONF/YANG with the OpenConfig data models. It implements a subset of the data models, namely the OpenConfig platform and optical transport as well as some extensions devised in the context of B5G-OPEN to report details about the transceiver operational modes.

	The software relies on ConfD free, a Tail-f/Cisco management agent software framework for network elements. It enables the industry adoption of NETCONF and YANG, and provides a simple mechanism to develop SDN agents focusing on the business logic and on the actual data models and semantics.  Protocol SDN Controller NBI Model Driven Development (Single domain) SBI Towards device or (sub)controllers Interface Definition Language Protocol NETCONF Data Modelling Language Agent Data Model Transponder Transponder Device
Description and Internal architecture of the component	OpenConfig agent relies on a ConfD process running, which implements the basic NETCONF/Yang framework. The software per-compiles data models and keeps a Configuration Database (CDB). Open Optical Terminals in general covers transponders, switchponders, muxponders, etc. with the ability to switch and multiplex multiple client signals into optical signals. The agent deals with uniform components hierarchy, multiplexing stages and Cross-connection logic discovery and Optical channel configuration (Frequency, power and operational mode). The actual logic is implemented as a second process that connects to the ConfD daemon via dedicated sockets. This process is written in C++ and implements different classes for interacting with the ConfD engine. MAAPI is C API which provides full access to the ConfD internal transaction engine. CDB API can read (committed) configuration from the CDB and has functions like cdb_set_value for operational (state) data only. With MAAPI it is possible to create or attach to existing transaction and access configuration data in the CDB. The modifications will be then propagated at commit time of the transaction. Notably, the agent takes care of the following aspects: • Notification of changes in the configuration database: in this sense, the actual SDN agent may react and configure the hardware accordingly. In particular, it registers appropriate call-backs for changes in the configuration of Optical Channels, including the actual frequency, transmit optical power and operational mode • It relies on CDB API and MAAPI from ConfD to write on the operational data store, in such a way that operational data can be

written to the ConfD database based on the status of the hardware. In particular, the agent MUST report the composition of the actual device in terms of components and subcomponents and reflect configuration changes (e.g., config/frequency) into state values (e.g., state/frequency).



Interface Specification

The interfaces of the component are:

- NETCONF/YANG as the basic framework.

The supported data models are, for and OpenConfig release:

openconfig/optical-transport/openconfig-transport-types@2017-08-16.yang

ietf/ietf-interfaces@2014-05-08.yang

openconfig/interfaces/openconfig-interfaces@2017-07-14.yang

```
openconfig/types/*.yang
openconfig/platform/*.yang
openconfig/optical-transport/*.yang
```

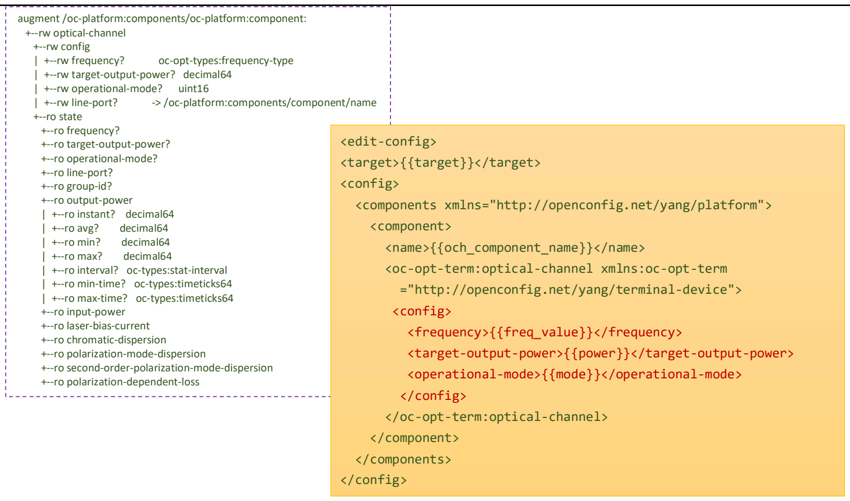
In particular, the agent implements the following extensions:

```
b5gopen/openconfig-terminal-device-property-types.yang

b5gopen/openconfig-terminal-device-properties.yang
```

The workflows that are in scope of B5G-OPEN are:

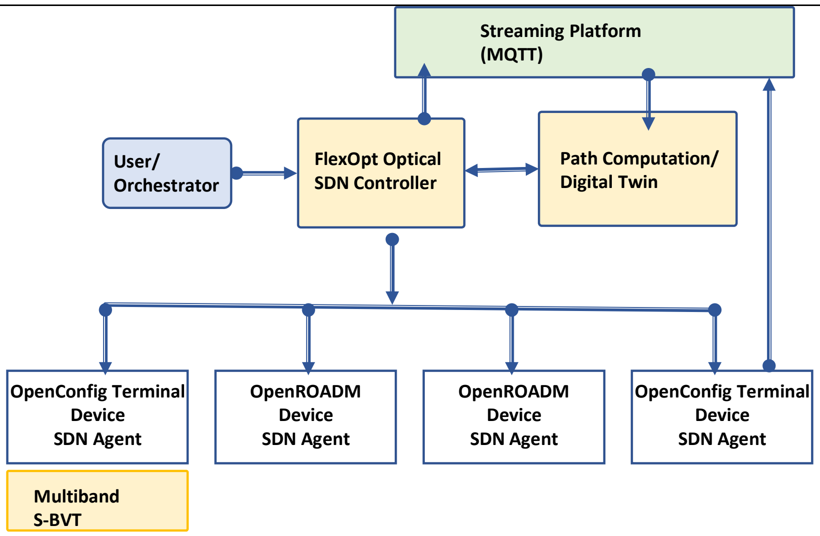
- Device discovery (NETCONF GET)
- OpenConfig Component discovery (NETCONF GET)



```
augment /oc-platform:components/oc-platform:component:
  +--rw optical-channel
    +--rw config
    |  +--rw frequency?           oc-opt-types:frequency-type
    |  +--rw target-output-power?   decimal64
    |  +--rw operational-mode?      uint16
    |  +--rw line-port?           -> /oc-platform:components/component/name
    +--ro state
      +--ro frequency?
      +--ro target-output-power?
      +--ro operational-mode?
      +--ro line-port?
      +--ro group-id?
      +--ro output-power
      |  +--ro instant?    decimal64
      |  +--ro avg?        decimal64
      |  +--ro min?        decimal64
      |  +--ro max?        decimal64
      |  +--ro interval?   oc-types:stat-interval
      |  +--ro min-time?   oc-types:timeticks64
      |  +--ro max-time?   oc-types:timeticks64
      +--ro input-power
      +--ro laser-bias-current
      +--ro chromatic-dispersion
      +--ro polarization-mode-dispersion
      +--ro second-order-polarization-mode-dispersion
      +--ro polarization-dependent-loss
```

```
<edit-config>
<target>{{target}}</target>
<config>
  <components xmlns="http://openconfig.net/yang/platform">
    <component>
      <name>{{och_component_name}}</name>
      <oc-opt-term:optical-channel xmlns:oc-opt-term
        ="http://openconfig.net/yang/terminal-device">
       <config>
         <frequency>{{freq_value}}</frequency>
         <target-output-power>{{power}}</target-output-power>
         <operational-mode>{{mode}}</operational-mode>
        </config>
      </oc-opt-term:optical-channel>
    </component>
  </components>
</config>
```

Functional Validation	The tests that can be done to validate the component are the following: • Startup of the ConfD daemon and loading of initial operational and configuration data • Retrieval of the datastore of the NETCONF agent. • Retrieval of components of the OpenConfig terminal device, including focusing on the optical channel augment. • Retrieval of the characteristics and current state of an Optical Channel components • Dynamic configuration of an optical channel attributes. This can be done as an emulated device or integrated with CTTC S-BVT and SONiC-based packet-optical nodes (i.e., pluggable configuration). • Characterization of a given Operational Mode (e.g., mode-id 100) in terms of B5G-OPEN physical impairment validation.
Component Integrations	The OpenConfig agent is integrated with the ONOS SDN Controller, for the control of sliceable BVTs as well as CTTC optical SDN controller

	 Streaming Platform (MQTT) User/ Orchestrator FlexOpt Optical SDN Controller Path Computation/ Digital Twin OpenConfig Terminal Device SDN Agent OpenROADM Device SDN Agent OpenROADM Device SDN Agent OpenConfig Terminal Device SDN Agent Multiband S-BVT The validation covers the management of S-BVT operational modes across the whole provisioning process workflow. We demonstrate the retrieval of the operational modes, how they are mapped to TAPI transceiver profiles, and later used for path computation/validation and the subsequent configuration of the BVT, including closed-loop adaptive transmission use cases. The CNIT version of the OpenConfig agent has been integrated with the SONiC-based switche and with the HHI transponders utilized in the experimental demonstration performed at HHI premises.

Component KPIs	The considered set of component KPIs that can be measured independently:
	• *Instantiation delay and footprint*: when the agent is running as a containerized application, characterized aspects related to instantiation of the agent, as well as aspects related to memory usage.
	• *Discovery latency*: measure the time and the control plane overhead (in terms of bytes, and throughput) it takes for an SDN controller to discover the components of the transceiver upon a NETCONF get operation.
	• *Operational Mode characterization*: measure the time and the control plane overhead (in terms of bytes, and throughput) it takes for an SDN controller to discover the details of a given operational mode, as defined within
	`b5gopen/openconfig-terminal-device-property-types.yang`
	`b5gopen/openconfig-terminal-device-properties.yang`
	• *Transaction delay*: the time it takes to send a configuration change, and this is reflected in the datastore. The focus shall be to change an Optical channel frequency, power and operational model. This KPI will be evaluated with and without hardware

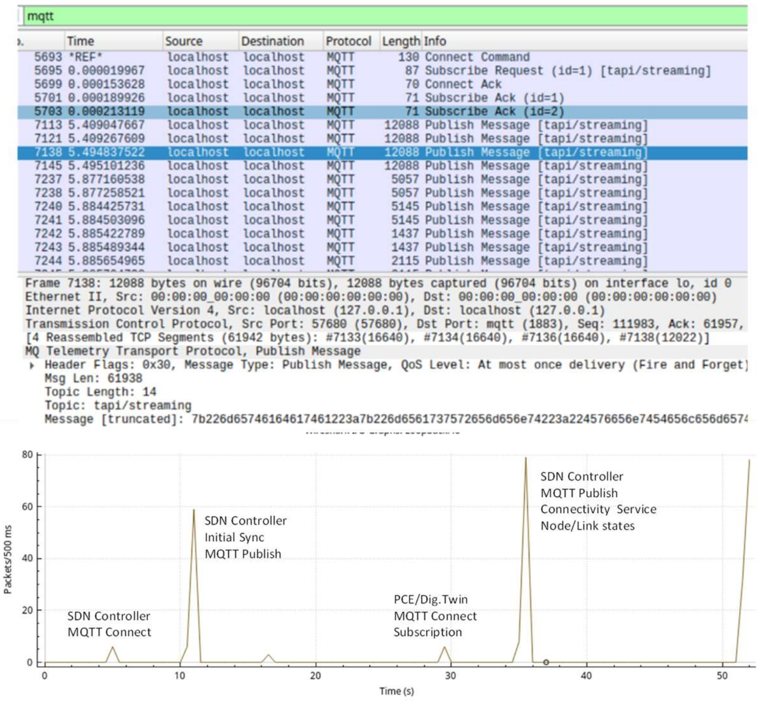
- The **Instantiation Delay** characterizes aspects related to the instantiation of the agent's containers, and/or memory usage. Launching the ConfD framework, which includes the loading of the operational and initial configuration data, can range from ~17 – 20 ms (lower bound when operational data and config data are pre-stored in xml files) to several seconds (1.230 s in a sample execution). This is due to the latency to retrieve operational data from the devices. The HAL startup time is ~8 ms, including subscription to events. Consequently, including the container orchestration latency, the initial startup of the SDN agent is characterized by O(10s).

- The **Discovery latency** is defined as the time of the SDN controller to discover the components of the transceiver upon a NETCONF get operation. This latency comprises the time and the control plane overhead (in terms of bytes, and throughput). With a back-to-back setting between the controller and the agent, the retrieval of components is done in ~475 ms (for an equivalent of ~400 xml lines). Similarly, the Operational Mode Characterization latency is the time to obtain the parameters of a given operational mode given its mode-id (see Fig. 10). The retrieval of the operational mode took ~300ms, associated to a NETCONF reply with 84 XML lines (4837 characters).

- At the transmitter side, the central frequency and power of the Tunable Laser Source (TLS) and the Digital/Analog Converter (DAC) channels are modified. At the receiver side, the OSC can be reconfigured. For example, two transceiver slices of the S-BVT1, working within C- and S-bands and corresponding to two different clients (c1 and c2), and two receiver slices of the S-BVT2 are configured. This operation determines different parameters, such as frequency (e.g., 193.4 THz, 200.26 THz), operational-mode (e.g., 111), name (e.g., OCH-A-Out-1, OCH-A-Out-2), power (e.g., 6.5 dBm, 4 dBm), status (e.g., enabled, disabled), type (e.g., optical-channel) and direction (e.g., TX, RX). Note that in this case, the power at S band is lower being constrained by laser stability.
- A total setup time of ~300s is needed to perform all the required OpenConfig operations to set up the connection. This time is mainly caused by the programmable elements of the MB S-BVT1, which include the TLS and the DAC, and are eventually configured within ~60s. On the other hand, the configuration of the MB S-BVT2 requires a higher setup time around 254s. The reason behind this is the time needed to configure the oscilloscope, which acts as ADC and includes both the signal acquisition and SNR/BER calculation (offline DSP).

We use MQTT as a streaming platform where a MQTT intermediate broker forwards publications in topics to subscribers. This enables synchronizing the FlexOpt SDN controller (publisher) with the PCE/DT functional element (subscriber). The latter connects to the Mosquitto MQTT broker and

<table>
<tr><td></td><td>subscribes to [tapi/streaming] topic to receive asynchronous notifications. The initial synchronization happens at startup, after the network has been discovered which takes O(6s), namely: 5 seconds for network discovery (with the corresponding OpenConfig and OpenROADM messages), and around 900 ms involving the sending of ~40 MQTT publish messages. The initial synchronization forwards to the PCE/DT relevant information, such as network topology and supported operational modes as reflected in the figure

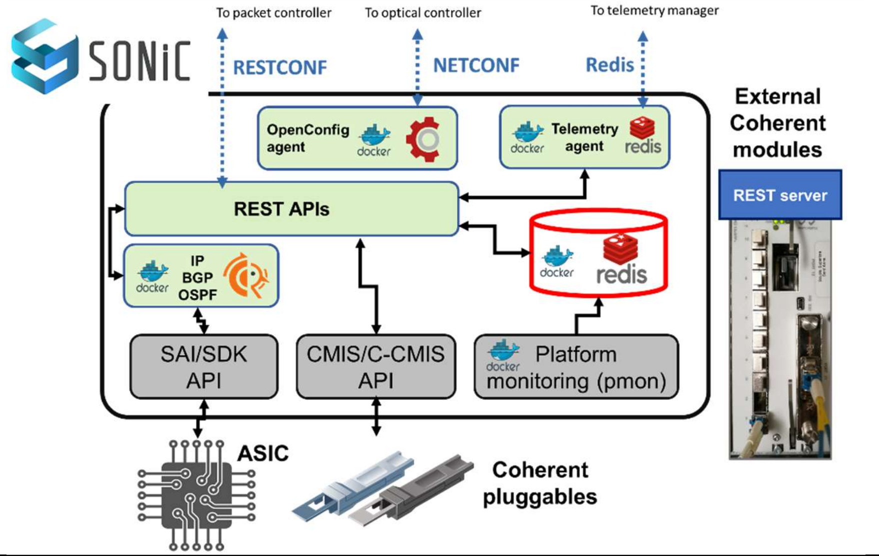


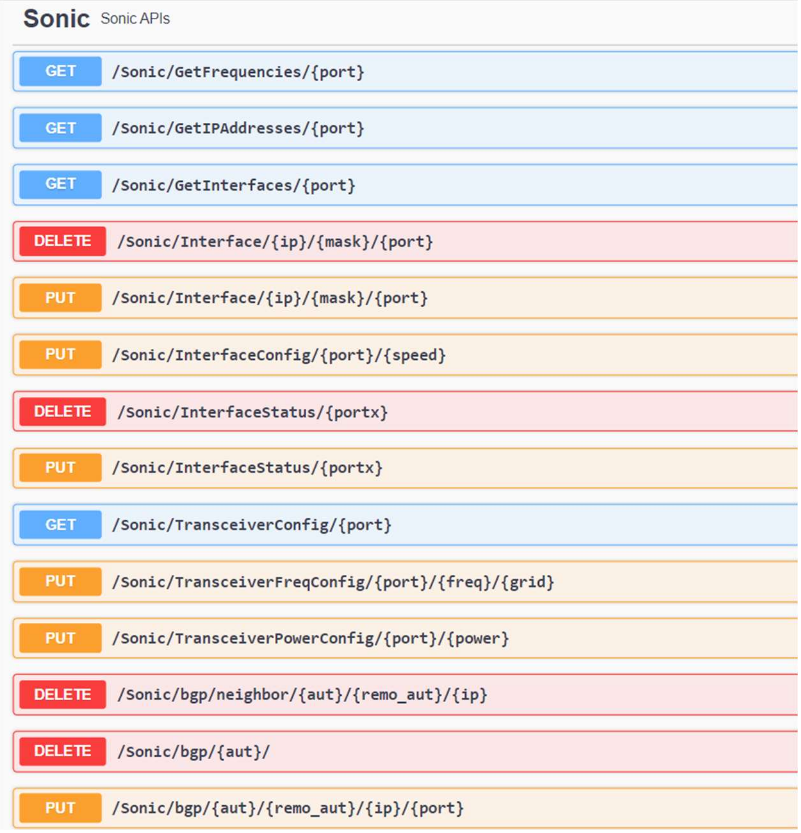
<pre>
mqtt
).   Time   Source   Destination Protocol Length Info
  5693 *REF*   localhost localhost   MQTT   130 Connect Command
  5695 0.000019967   localhost localhost   MQTT   87 Subscribe Request (id=1) [tapi/streaming]
  5699 0.000153628   localhost localhost   MQTT   70 Connect Ack
  5701 0.000189926   localhost localhost   MQTT   71 Subscribe Ack (id=1)
  5703 0.000213119   localhost localhost   MQTT   71 Subscribe Ack (id=2)
  7113 5.409047667   localhost localhost   MQTT   12088 Publish Message [tapi/streaming]
  7121 5.409267609   localhost localhost   MQTT   12088 Publish Message [tapi/streaming]
  7138 5.494837522   localhost localhost   MQTT   12088 Publish Message [tapi/streaming]
  7145 5.495101236   localhost localhost   MQTT   12088 Publish Message [tapi/streaming]
  7237 5.877160538   localhost localhost   MQTT   5057 Publish Message [tapi/streaming]
  7238 5.877258521   localhost localhost   MQTT   5057 Publish Message [tapi/streaming]
  7240 5.884425731   localhost localhost   MQTT   5145 Publish Message [tapi/streaming]
  7241 5.884503096   localhost localhost   MQTT   5145 Publish Message [tapi/streaming]
  7242 5.885422789   localhost localhost   MQTT   1437 Publish Message [tapi/streaming]
  7243 5.885489344   localhost localhost   MQTT   1437 Publish Message [tapi/streaming]
  7244 5.885654965   localhost localhost   MQTT   2115 Publish Message [tapi/streaming]
Frame 7138: 12088 bytes on wire (96704 bits), 12088 bytes captured (96704 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
Transmission Control Protocol, Src Port: 57680 (57680), Dst Port: mqtt (1883), Seq: 111983, Ack: 61957,
[4 Reassembled TCP Segments (61942 bytes): #7133(16640), #7134(16640), #7136(16640), #7138(12022)]
MQ Telemetry Transport Protocol, Publish Message
 ▸ Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
   Msg Len: 61938
   Topic Length: 14
   Topic: tapi/streaming
   Message [truncated]: 7b226d6574616461617461223a7b226d6561737572656d656e74223a224576656e7454656c656d657
</pre>

Fig. Wireshark capture of the initial MQTT synchronization</td></tr>
<tr><td><b>Status, availability, repository</b></td><td>CTTC and CNIT software with proprietary license.
Relies on Cisco / Tail-F ConfD framework.</td></tr>
<tr><td><b>Additional Remarks</b></td><td>Full paper with performance details: [Cas24b]</td></tr>
</table>

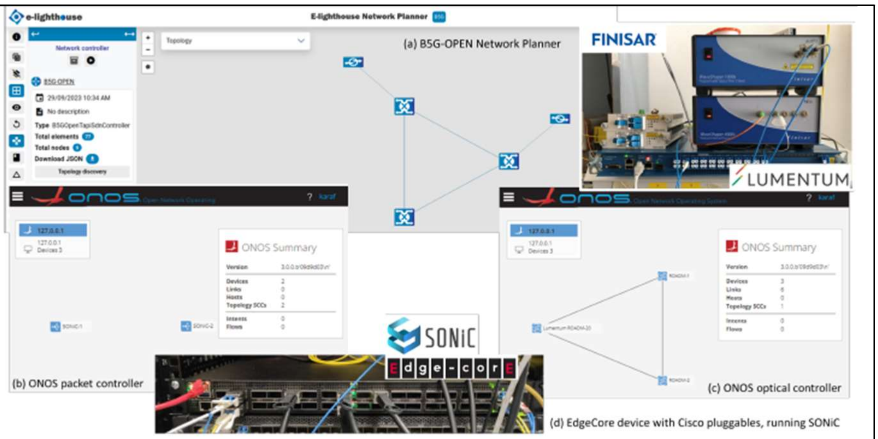
3.10 SONIC-BASED PACKET OPTICAL NODE (CNIT)

Component Name: SONiC-based packet optical node	
Summary	SONiC has been used as network operating system (NOS) of packet-optical white-box nodes and extended with some components (illustrated in green in the architectural figure) to enable the integration with other B5G-OPEN components: (1) docker container running the OpenConfig agent (CNIT version) to control external transponders, local coherent transceivers, and IP interfaces; (2) REST-based APIs enabling the control of coherent transceivers, Ethernet/IP interfaces and routing processes; (3) exporter/adapter of monitoring information from SONiC system and transceivers to the telemetry server.

Description and Internal architecture of the component	 To packet controller To optical controller To telemetry manager SONiC RESTCONF NETCONF Redis OpenConfig agent docker Telemetry agent redis External Coherent modules REST server REST APIs IP BGP OSPF SAI/SDK API CMIS/C-CMIS API Platform monitoring (pmon) ASIC Coherent pluggables
Interface Specification	The SONic-based packet-optical node integrates with the following elements: • To SDN optical controller: the OpenConfig agent exposes a YANG-based model to the SDN optical controller that will be consumed through NETCONF. • To SDN packet controller: the OpenConfig agent exposes a YANG-based model to the SDN packet controller to be consumed through NETCONF, that can be used to configure IP interfaces. In addition, the SDN packet controller can directly access the REST-based interface using RESTCONF. • To telemetry manager: the telemetry agent communicates with the telemetry manager adopting a client Redis adapter, exploiting a Pub/Sub mechanism, injecting the data with the proper format to the remote Redis DB. • From the agent to REST-based APIs: (1) the OpenConfig agent interacts with the REST APIs to read and write the pluggable coherent transceivers configuration, to configure Ethernet/IP interfaces and setup BGP adjacency; (2) the OpenConfig agent is also enabled to interact with external coherent modules to enable the control of experimental transceivers; (3) the telemetry agent interacts with the REST APIs to retrieve monitoring information via local Redis DB. Following a capture of the REST API swagger is shown with the configuration registered paths.

	
	Sonic Sonic APIs
	GET /Sonic/GetFrequencies/{port}
	GET /Sonic/GetIPAddresses/{port}
	GET /Sonic/GetInterfaces/{port}
	DELETE /Sonic/Interface/{ip}/{mask}/{port}
	PUT /Sonic/Interface/{ip}/{mask}/{port}
	PUT /Sonic/InterfaceConfig/{port}/{speed}
	DELETE /Sonic/InterfaceStatus/{portx}
	PUT /Sonic/InterfaceStatus/{portx}
	GET /Sonic/TransceiverConfig/{port}
	PUT /Sonic/TransceiverFreqConfig/{port}/{freq}/{grid}
	PUT /Sonic/TransceiverPowerConfig/{port}/{power}
	DELETE /Sonic/bgp/neighbor/{aut}/{remo_aut}/{ip}
	DELETE /Sonic/bgp/{aut}/
	PUT /Sonic/bgp/{aut}/{remo_aut}/{ip}/{port}
	• The REST APIs interacts with the SAI/SDK and CMIS/C-CMIS to perform the configuration of pluggable coherent transceivers

Functional Validation	The tests performed to validate the component are the following: • Startup of the SONiC NOS and loading of initial data correctly performed. Validate basic functionalities such as: the Performance Monitoring (pmon), the REDIS database, the CMIS/C-CMIS APIs, the SAI/SDK APIs. • Deployment of each considered containers and perform functional validation inside the SONiC environment. Executed without significant degradation with respect to the test already executed for the validation of each component. • For such containers using the SONiC APIs, configuration requests received from the upper layers (e.g., the SDN optical controller) are correctly translated in a node configuration (e.g., the tuning of a coherent pluggable on a specified wavelength). • Multi-layer traffic switching: packet traffic incoming on packet interfaces is correctly routed through coherent modules.


e-lighthouse
E-lighthouse Network Planner
(a) B5G-OPEN Network Planner
FINISAR
LUMENTUM
(b) ONOS packet controller
ONOS Summary
SONiC
Edge-corE
(c) ONOS optical controller
(d) EdgeCore device with Cisco pluggables, running SONiC

The figure illustrates an experimental deployment including two SONiC switches, and other devices. Specifically, the packet controller based on ONOS (on the left) is connected to the SONiC switches using the OpenConfig agent. During the experiment the optical coherent transceivers have been configured several times using different configuration parameters, successfully validating the components deployed on the SONiC switches.

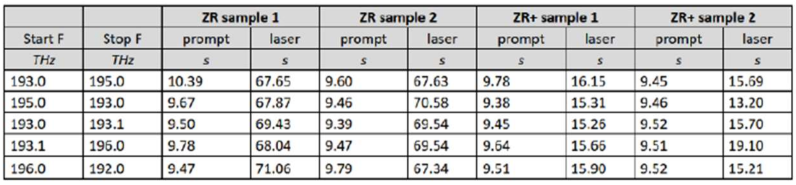
Component Integrations

Other components this component is integrated with:

- SDN optical controller
- SDN packet controller
- Network orchestrator
- Telemetry manager.

Component KPIs

- Time required at the data layer for enforcing a modification of the central frequency of the optical coherent transceivers, has been measured using ZR and ZR+ transceivers:



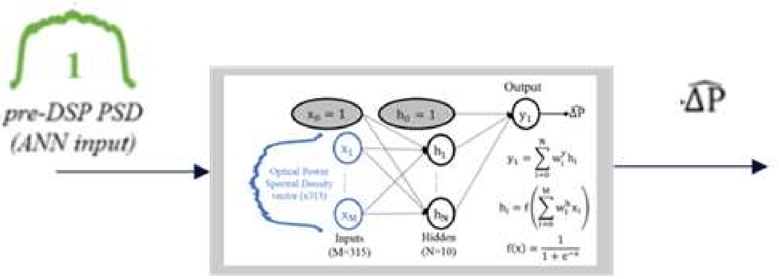
		ZR sample 1		ZR sample 2		ZR+ sample 1		ZR+ sample 2	
Start F	Stop F	prompt	laser	prompt	laser	prompt	laser	prompt	laser
THz	THz	s	s	s	s	s	s	s	s
193.0	195.0	10.39	67.65	9.60	67.63	9.78	16.15	9.45	15.69
195.0	193.0	9.67	67.87	9.46	70.58	9.38	15.31	9.46	13.20
193.0	193.1	9.50	69.43	9.39	69.54	9.45	15.26	9.52	15.70
193.1	196.0	9.78	68.04	9.47	69.54	9.64	15.66	9.51	19.10
196.0	192.0	9.47	71.06	9.79	67.34	9.51	15.90	9.52	15.21

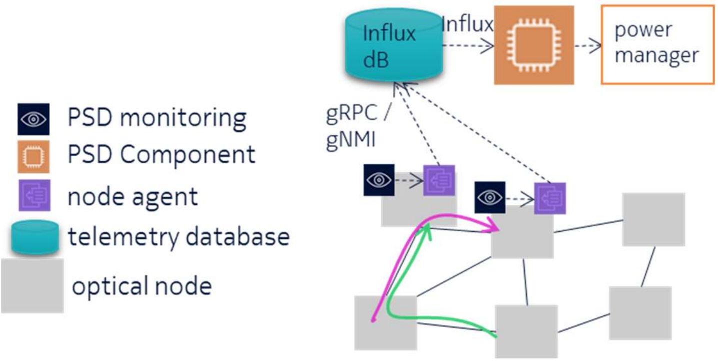
- The spectrum width required to preserve the quality of transmission of a 400 Gbps channel with 16-QAM modulation format has been evaluated in experimental measures:

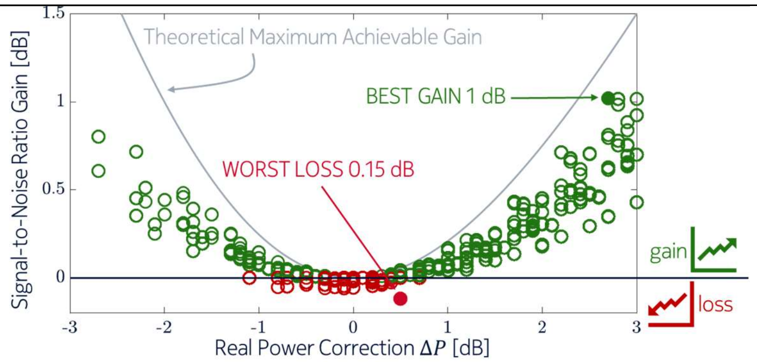
Central Freq [THz]	Bandwidth [GHz]	Received power [dBm]	BER	OSNR [dB]	eSNR [dB]	Status
195.5	100	-16.31	0.00253	29.9	16.6	UP
195.5	90	-16.31	0.00256	29.8	16.6	UP
195.5	80	-16.36	0.00257	29.7	16.6	UP
195.5	70	-16.42	0.00276	29.5	16.5	UP
195.5	64	-16.54	0.00319	28.9	16.3	UP
195.5	62	-16.66	0.00359	28.5	16.2	UP
195.5	60	-16.72	0.00417	28.0	16.0	UP
195.5	58	-16.88	0.00728	26.6	15.3	DOWN
195.5	56	-17.03	0.01297	25.0	14.4	DOWN

Status, availability, repository	CNIT software with proprietary license.
Additional Remarks	Detailed functional and KPI evaluation can be found in [Sga24]

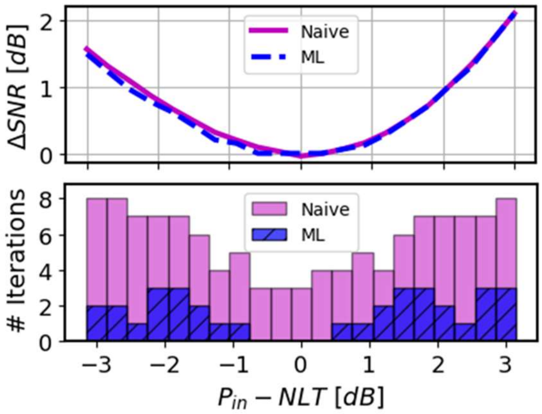
3.11 AI/ML MODELS FOR PSD AND POWER MANAGEMENT (NOKIA)

Component Name: Automatic power correction	
Summary	The desired performance is defined as a compromise between linear and nonlinear contributions. It is the so-called nonlinear threshold which is operating in the weakly nonlinear regime. Monitoring ASE and NL signal to noise ratios is quite complex in a live network and has some limitations which make this problem attractive for artificial neural networks (ANN). The optical spectrum contains knowledge regarding linear and nonlinear impairments specifically the shape is different if operating in linear or nonlinear regime. In addition, the SNR-induced fluctuations from the polarisation dependent loss (PDL) also show a pdf with a different shape in linear and nonlinear regimes. Therefore, we can use either of these 2 shapes at the input of the ANN to get an optimal power correction to be applied.
Description and Internal architecture of the component	**ANN for power monitoring architecture** For a given lightpath, the neural network takes as input normalized power spectral density (PSD) or PDL-induced SNR fluctuation PDF vector and returns as output an estimated power correction ($\widehat{\Delta P}$). It works with AI/ML and uses an ANN with 1 hidden layer with 10 neurons and 1 output layer with 1 neuron. A neuron is a sequence of two operations: a weighted sum followed by a nonlinear manipulation. It is a fully connected ANN meaning that all neurons of the following layer take as input all neurons from the preceding layer. The hidden layer employs sigmoid function and the output layer uses the identity function. We use pre-processing to normalize the PSD or PDF in the range [0, 1].  1 pre-DSP PSD (ANN input) $x_0 = 1$ $h_0 = 1$ Output y_1 ΔP x_1 x_M Optical Power Spectral Density vector (x315) h_1 h_N Inputs (M=315) Hidden (N=10) $y_1 = \sum_{i=0}^{N} w_i^y h_i$ $h_i = f\left(\sum_{i=0}^{M} w_i^h x_i\right)$ $f(x) = \frac{1}{1+e^{-x}}$ $\widehat{\Delta P}$

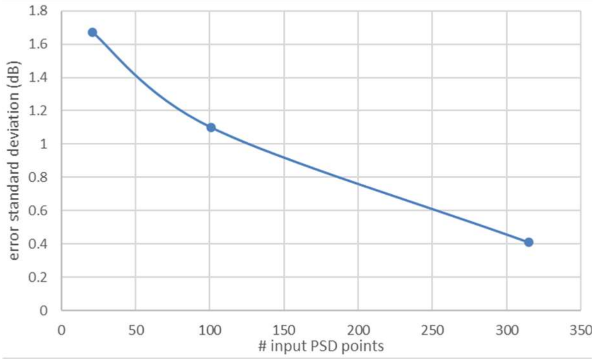
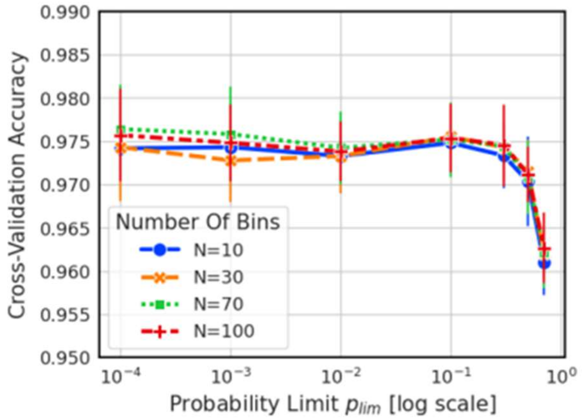
Interface Specification	The PSD component will be used as a standalone component. It is integrated with an optical mesh network and telemetry database, as shown in the figure below.  Influx dB Influx power manager gRPC / gNMI PSD monitoring PSD Component node agent telemetry database optical node In the figure, the green and pink lightpaths are leveraging the PSD component to optimize their performance. Each of them has the following workflow. First the optical PSD is measured in the optical node and sent to the node agent, responsible for communicating with the control & management plane to (re)-configure optical nodes (here transponder launch powers) and feed the telemetry database via a gRPC / gNMI interface. Then the PSD component pre-processes the PSD to normalize it and optionally compress it to a given format, feeds the ANN with the resulting PSD and obtains estimated power correction for each ligthpath. These estimated power corrections are stored in a power manager and can be used as recommendations to reconfigure the optical nodes.
Functional Validation	Tests done to validate the component - Generate a set of measurements (different powers, lengths, …) to feed the database for training, validation and testing of the neural network. - Evaluate the error between known optimal powers and new estimated powers - Assess the SNR gain obtained after applying power corrections
Component Integrations	This component is integrated with a physical layer testbed to be able to collect measurements. This is needed to build a database used for training, validation and test of the PSD component.
Component KPIs	• *Gain/loss of performance:* We measure the SNR gain/loss after power corrections for both PSD and PDF shapes as input feature. We show in the figure below the results for the PSD shape. The SNR gain after correction is 0.5 dB for 1dB power correction. On one had we can slightly lose close to the optimal power a 0.15dB, but in the other hand, we can win 1dB for 2.5dB power correction.



In a later work, we proposed to use the PDF shape as input features. We show similar trends in the measurement of SNR gain/loss after power corrections as with the PSD shown in the top figure below. We can observe a 0.3dB gain for 1dB power corrections. Note that the two figures cannot be compared as the transmission parameters are not the same, notably in the second case, PDL impairments is accounted for.

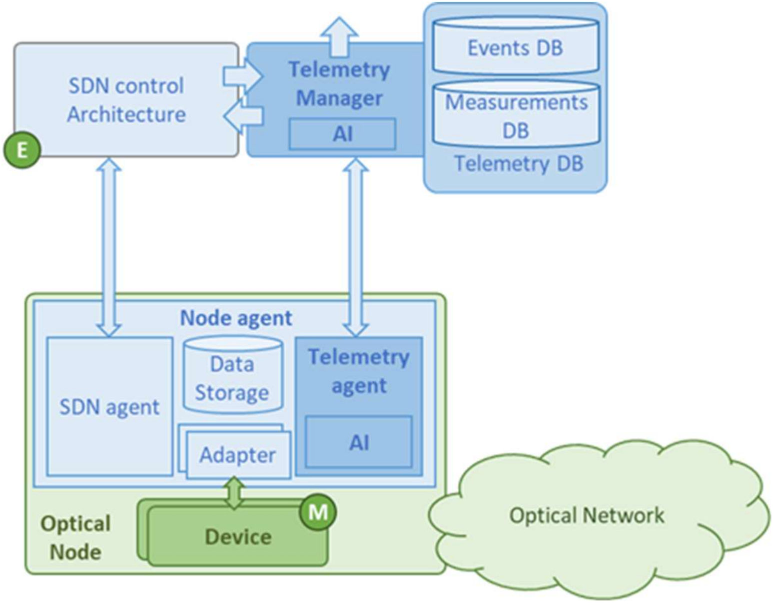


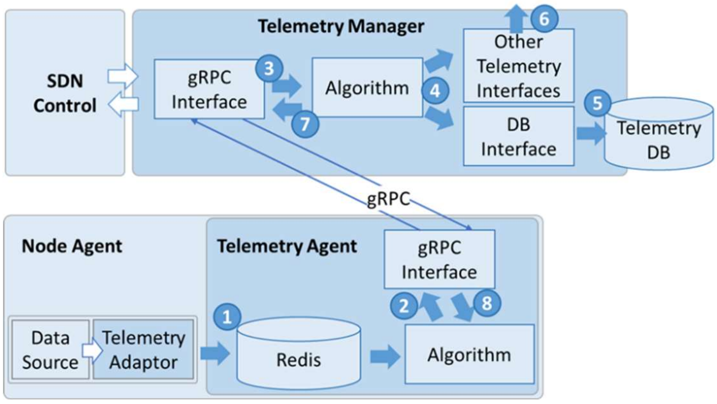
- *Compression rate*: evaluate the minimum number of points required in optical PSD / PDF to get a given accuracy. We investigated the scalability of the component. First, we compress the input PSD by taking fewer points equally spaced. We investigated 3 different configurations with either 20, 100 and 315 input PSD points. As a figure of merit, we defined the error as the different between the true power and the predicted output power. We plot in the figure below the standard deviation of the error as a function of the number of input PSD points. As expected, when the number of PSD points is low, the standard deviation increases. For 20 PSD points, the error standard deviation is 1.67dB while for 315 PSD points it can decrease down to 0.41 dB. To maintain a reasonable performance, we can go down to 100 PSD points which gives 1dB error standard deviation for the investigated line configurations.

	 Second, we compress the input of the PDF shape by either taking fewer points or by cutting the PDF tails. We notice that the accuracy is almost constant when we decrease the number of points (#bins). However, when we cut the tails (when P_{lim} → 1) we start to degrade the performance. Both compressing techniques are shown in the plot below. 
Status, availability, repository	We developed two components leveraging AI/ML models for power optimisation with either power spectrum density (see D4.2) or PDL-induced SNR fluctuations (see D4.3). We proposed low complexity implementation for both by evaluating the minimum number of inputs required. And we tested their performance accuracy. It is completed and available as a Nokia proprietary Matlab toolbox.
Additional Remarks	No contribution to standard

3.12 Telemetry System (UPC)

Component Name:	
Summary	Several telemetry architectures are available. In general, telemetry data is collected from observation points in the devices and send to a central system running besides the SDN controller. Although protocols specifically devised for telemetry, like gRPC, compress data, the amount of data that can be collected and the frequency of collection make those architecture not practical.

	Our distributed telemetry system integrates measurements and event data collection and supports intelligent data aggregation nearby data collection, so agents receive and analyse measurements before sending to a centralized manager.
Description and Internal architecture of the component	The figure presents the reference network architecture with distributed telemetry. An SDN architecture controls a number of optical nodes, e.g., optical transponders and reconfigurable optical add-drop multiplexers, in the data plane. A centralized telemetry manager is in charge of receiving, processing and storing telemetry data in a telemetry database (DB). Typically, the telemetry manager runs inside a Monitoring and Data Analytics (MDA) system. 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 DB describing the lightpaths.  SDN control Architecture E Telemetry Manager AI Events DB Measurements DB Telemetry DB Node agent SDN agent Data Storage Adapter Telemetry agent AI Optical Node Device M Optical Network Every node in the data plane is locally managed by a node agent, which translates the control messages received from the related SDN controller into operations in the local node and exports telemetry data collected from observation points (labelled M) enabled in the optical nodes. In addition, events can be collected from applications and controllers (labelled E). Telemetry agents run inside node agents and provide the needed services for intelligent algorithms based on Artificial Intelligence (AI) techniques to process collected telemetry measurements. Internally, both, the telemetry agent and manager are based on three main components: *i*) a manager module configuring and supervising the operation of the rest of the modules; *ii*) a number of modules that include algorithms, e.g., data processing, aggregation, etc. and interfaces, e.g., gRPC; and *iii*) a Redis DB that is used in publish-subscribe mode to communicate the different modules among them. This solution provides an agile and reliable environment that simplifies communication, as well as integration of new modules. 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.

Interface Specification	A detailed architecture of the proposed telemetry system is presented in the next figure, where the internal architecture of telemetry agents inside node agents and the telemetry manager is shown.  Telemetry Manager SDN Control gRPC Interface 3 Algorithm 4 6 Other Telemetry Interfaces DB Interface 5 Telemetry DB 7 gRPC Node Agent Telemetry Agent gRPC Interface 2 8 1 Data Source Telemetry Adaptor Redis Algorithm Let us describe a typical telemetry workflow valid for a wide range of use cases. The node agent includes modules (denoted data sources) that gather telemetry data from observation points in the optical nodes. Examples include optical spectrum analysers (OSA) in the ROADMs and data from digital signal processing, e.g., optical constellations, in the TPs. A telemetry adaptor has been developed, so data sources can export collected data to the telemetry system; specifically, the adaptor receives raw data from the data source and generates a structured json object, which is then published in the local Redis DB (labelled 1 in the figure). The periodicity for data collection can be configured within a defined range of values. A number of algorithms can be subscribed to the collected measurements. In this example, let us assume that only one algorithm is subscribed, which processes the measurements locally. Such processing might include doing: i) no transformation on the data (null algorithm); ii) some sort of data aggregation, feature extraction or data compression; or iii) some inference (e.g., for degradation detection). The output data (transformed or not) are sent to a gRPC interface module through the Redis DB (not shown in the figure) (2), which conveys the data to the telemetry manager. Because gRPC requires a previous definition of the data to be conveyed, our implementation encodes the received data in base64, which allows generalization of the telemetry data to be conveyed. Note that, although such encoding could largely increase the volume of data to be transported, intelligent data aggregation performed by telemetry agents could reduce such volume to a minimum. In the telemetry manager, the data are received by a gRPC interface module that publishes them in the local Redis DB, so subscribed algorithms can receive them. The algorithms in the telemetry manager can implement functions related to data aggregation, inference, etc. Once processed, the output data is published in the local Redis DB (4) and can be stored in the telemetry DB (5) and/or be exported to external systems (6). Interestingly, algorithms in the telemetry manager can communicate with those in the telemetry agents using the gRPC interface (7-8). Examples of such communication include parameter tuning, among others.

Functional Validation	Tests done to validate the component: - We generated measurements using the data source and the telemetry adaptor and verified that: 1) they are received by the selected algorithm in the telemetry agent; and 2) they are sent to the telemetry manager and stored in the measurements DB. - We generated events with the SDN controller and verify that they are stored in the events DB.
Component Integrations	The Telemetry system integrates with any data source provided that they implement the telemetry adaptor. Examples of data sources for measurements are transponders, ROADMs, and OSAs. Examples of data sources for events are controllers and agents. The telemetry manager integrates also with any other management system using external delivery systems, like Kafka. The Telemetry system was been demonstrated in OFC 2023 [Gon23b], where we showed integration with network devices from Nokia and ADVA, as well as with the CTTC's SDN controller.
Component KPIs	Each constellation sample includes 2048 symbols from a 16-QAM optical signal and has a size of 16,384 bytes. 1. Autoencoder: Trained for the maximum compression that produces a reproduction error < 2%, which results in vectors Z of size 32 bytes. Vectors Z are output as JSON objects, where each component is represented with 11 characters, and then compressed, which resulted in 607 bytes. When the message arrives at the telemetry manager through the gRPC interface, it is used as input to the decoder that generates a sample X*, which is finally stored in the telemetry DB. In our tests, both, data encoding and decoding took 60 ms 2. Supervised feature extraction: the algorithm in the telemetry agent applies GMM fitting to every constellation sample X received and generates outputs of m = 16 vectors with 5 features each. This process, outputs a JSON object with 1,159 characters, which is then conveyed through the gRPC interface using 1,545 bytes.
Status, availability, repository	The pseudocode of the algorithms together with a thorough description, and the performance evaluation has been reported in D4.3. An example of data source has been made available to the project's partners and it was used for the OFC 2023 demonstration [Gon23b]. The performance evaluation is available in [Vel23a] and [Gon23a]
Additional Remarks	No contribution to standard/open source

3.13 FlexTelemetry Agent (Adtran)

Component Name: Adtran FlexTelemetry agent	
Summary	*Flex-Telemetry* is an application that periodically requests and collects performance measurements from optical transport network devices. It utilizes NETCONF and supports both open (OpenConfig) and proprietary data models to ensure comprehensive data collection. The program features a modular plugin system that provides a Northbound Interface (NBI), capable of delivering stable telemetry streams to various mediums, including time-series databases, in-memory databases, and International Data Spaces (IDS). This modular approach allows Flex-Telemetry to seamlessly integrate with diverse data storage and processing systems, facilitating efficient, scalable access to performance data across different platforms.
Description and Internal architecture of the component	The FlexTelemetry Agent is a Python-based application designed to collect and stream telemetry data from multiple network devices in real-time. It uses NETCONF for general telemetry data collection and SNMP specifically for amplifier data. The collected telemetry data can be sent to various northbound interfaces, such as Redis, Apache Kafka, MQTT (Mosquitto), and InfluxDB, providing a flexible and scalable solution for monitoring and analyzing performance metrics. The agent supports a range of devices, including optical transponders, Carrier Ethernet switches, and optical amplifiers, making it versatile for different network environments. One of the core features of the FlexTelemetry Agent is its integration with Telegraf for real-time data handling. When data is sent to message brokers like Kafka and MQTT, the agent automatically spawns a Telegraf instance that collects and routes the metrics to InfluxDB. This enables applications needing real-time data—such as machine learning models—to access performance data immediately, while historical data is stored in a time-series database for later analysis, supporting both real-time and retrospective analytics. The FlexTelemetry agent was successfully tested in the OFC 2023 demo. The KPIs to measure the saleability and performance of the FlexTelemetry agent are the minimum time interval between reads and number of supported parameters/devices with different SBI drivers and NBI plugins.
Interface Specification	The agent uses NETCONF and SNMP in the southbound to retrieve the data from the devices in the data plane. The agent exposes various output plugins in the north bound including InfluxDB, kafka, MQTT and Redis for data collection and processing. .
Functional Validation	Tests done to validate the component: - Tested data retrieval and exposure of all the plugins on the real network devices deployed in the lab.

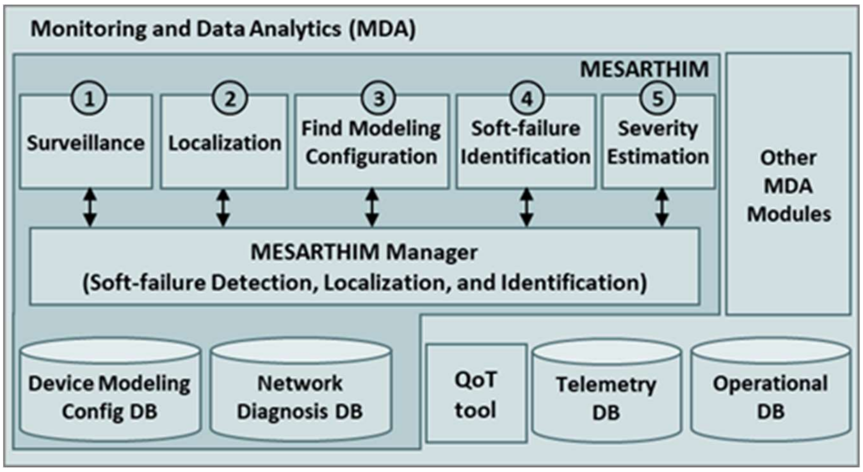
Component Integrations	The FlexTelemetry agent integrates with a data analytics pipeline using Kafka, Redis, or direct time-series database interface like InfluxDB. In B5G-OPEN, the FlexTelemetry agent acts as a Telemetry Adaptor and interfaces to the Telemetry architecture using Redis. Examples of data sources for measurements are transponders, ROADMs, and inline amplifiers. The FlexTelemetry agent has been demonstrated in OFC 2023 [Gon23b], where we showed integration with the Telemetry manager from UPC.
Component KPIs	
Status, availability, repository	The pseudocode of the algorithms together with a thorough description, and the performance evaluation has been reported in D4.3. An example of data source has been made available to the project's partners and it was used for the OFC 2023 demonstration [Gon23b].
Additional Remarks	No contribution to standard/open source

3.14 MESARTHIM – FAILURE MANAGEMENT USING A SNR DIGITAL TWIN (UPC)

Component Name: MESARTHIM	
Summary	The performance of optical devices can degrade because of aging and external causes like, for example, temperature variations. Such degradation might start with a low impact on the Quality of Transmission (QoT) of the supported lightpaths (soft-failure). However, it can degenerate into a hard-failure if the device itself is not repaired or replaced, or if an external cause responsible for the degradation is not properly addressed. MESARTHIM compares the QoT measured in the transponders with the one estimated using a QoT tool. Those deviations can be explained by changes in the value of input parameters of the QoT model representing the optical devices, like noise figure in optical amplifiers and reduced Optical Signal to Noise Ratio in the Wavelength Selective Switches. By applying reverse engineering, MESARTHIM estimates the value of those modelling parameters as a function of the observed QoT of the lightpaths.
Description and Internal architecture of the component	The optical layer consists of a disaggregated set of ROADMs and TRXs, and a set of optical links with a number of In-Line OAs interconnecting ROADMs. The control plane includes: i) a Network Controller to program the network devices; and ii) a Monitoring and Data Analytics (MDA) system that includes the telemetry system in charge of collating measurements from the data plane, analyses the data and issues recommendations to the network controller, as well as notifications regarding failures. The MDA system stores a replica of the operational

databases (DB) that are synchronized from the network controller. A QoT digital twin based on GNPy that estimates the SNR of the lightpaths is used for connection provisioning and for failure analytics. In addition, it collects measurements from the optical devices with a given periodicity and stores them in a TelemetryDB. These measurements are used by MESARTHIM to: i) estimate those modeling parameters related to optical devices (resources); ii) analyze the evolution of the measured SNR and that of the modeling parameters to detect any degradation as soon as it appears; and iii) determine the severity of the degradation based on the foreseen impact on the performance of the lightpaths.

The figure next sketches the MESARTHIM methodology implemented in the MDA system.

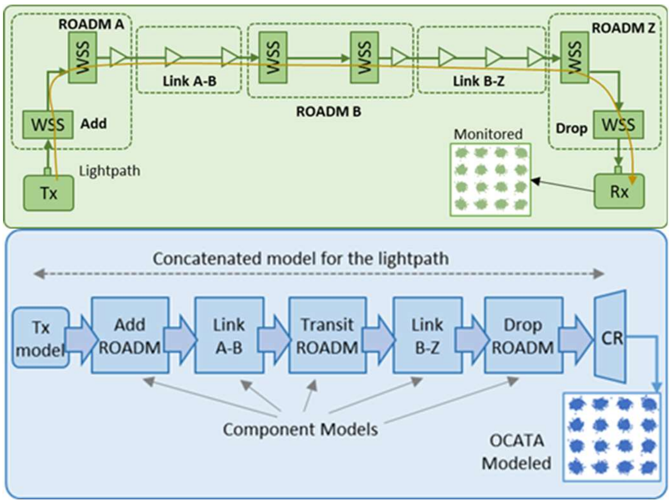

Monitoring and Data Analytics (MDA)
MESARTHIM
1
Surveillance
2
Localization
3
Find Modeling Configuration
4
Soft-failure Identification
5
Severity Estimation
Other MDA Modules
MESARTHIM Manager
(Soft-failure Detection, Localization, and Identification)
Device Modeling Config DB
Network Diagnosis DB
QoT tool
Telemetry DB
Operational DB

Specifically, the following building blocks can be identified: (1) the Surveillance block that analyses the SNR measurements and the value of modelling parameters to detect any meaningful degradation (e.g., by threshold crossing); (2) the Localization block that localizes the soft-failure; (3) the “Find Modelling Configuration” block that finds the most likely value of the modelling parameters of a given resource, so it results into SNR values of the lightpaths being supported by such resources similar to those that have been actually measured; (4) the soft-failure Identification block that, assuming a resource has been localized as the source of the soft-failure, finds what is the modelling parameter responsible for such failure; and (5) the Severity Estimation block that estimates whether and when the soft-failure will degenerate into a hard-failure. In addition, two internal repositories are used: i) the Device Modelling Config DB with the evolution of the value of modelling parameters along time for every resource; and ii) the Network Diagnosis DB that stores historical data for analysis purposes. The MESARTHIM manager coordinates those blocks to achieve intelligent QoT analysis.

Interface Specification	MESARTHIM runs as part of the MDA, and it can access: - the telemetry measurements DB to analyse constellation samples - the Operational DB to get the route of the lightpaths. - the QoT tool
Functional Validation	Tests done to validate the component: - The Find Modeling Configuration has been evaluated experimentally in the testbed in collaboration with CNIT. - A simulation environment was used to validate and assess the rest of the components over a German-like network scenario. The optical data plane was simulated by a GNPy instance. We generated SNR measurements for every lightpath by varying every modelling parameter of every intermediate OAs and A/D WSSs in the ROADMs in the network, independently. - The resulting samples were stored in the simulated control plane and fed the module implementing the MESARTHIM methodology.
Component Integrations	This component is integrated in an MDA system and with the telemetry system, which provides measurements.
Component KPIs	- Find Modeling Configuration block $R^2 > 0.98$ - Anticipation of soft failures > 15% through the estimation of modelling parameters w.r.t. SNR analysis. - Relative average error of the modelling parameters estimation < 8% - Severity estimation anticipation > 40% -
Status, availability, repository	The pseudocode of the algorithms together with a thorough description has been reported in D4.3 and [Vel23b]. The performance evaluation has been reported in D4.3.
Additional Remarks	No contribution to standard/open source

3.15 OCATA - DIGITAL TWIN FOR THE OPTICAL TIME DOMAIN (UPC)

Component Name:	
Summary	The development of Digital Twins to represent the optical transport network might enable multiple applications for network operation, including automation and fault management. OCATA is a deep learning-based digital twin for the optical time domain that is based on the concatenation of deep neural networks (DNN) modelling optical links and nodes, which facilitates representing lightpaths. The DNNs model linear and nonlinear noise, as well as optical filtering. Additional DNN-based models extract useful lightpath metrics, such as lightpath length, number of optical links and nonlinear fibre parameters. OCATA exhibits low complexity, thus making it ideal for real-time applications.

Description and Internal architecture of the component	We assume disaggregated optical networks, with transponders, ROADMs and optical amplifiers from multiple vendors and assume that information regarding the network topology, the type of fibres, etc., as well as the configuration and monitoring data from every optical component is accessible. In this scenario, a lightpath from site A to site Z can be modelled by concatenating models for the different components supporting such lightpath, i.e., transponders, ROADMs, and optical links, where output IQ optical constellation features of one component model are the input features of the following one.  ROADM A WSS Link A-B WSS WSS Link B-Z WSS ROADM Z ROADM B WSS Add Drop WSS Monitored Lightpath Tx Rx Concatenated model for the lightpath Tx model Add ROADM Link A-B Transit ROADM Link B-Z Drop ROADM CR Component Models OCATA Modeled Every component model modifies the input features according to the noise that the specific physical network component introduces. Specifically, a transmitter (Tx) model generates the initial constellation features following a Tx configuration. Then, models for ROADMs and optical links are concatenated in the same order that the respective network components appear in the route of the lightpath. To minimize complexity and ensure component model availability at lightpath provisioning time, such models are trained beforehand using datasets collected from the network and/or coming from simulation. Then, at provisioning time, the specific concatenated model for the lightpath is created by selecting trained component models for the network components in the route of the lightpath, from a model database. Finally, to reduce complexity even more, only the features of a few selected constellation points are propagated. Consequently, a constellation reconstruction (CR) module generates the features of the non-propagated constellation points based on the received features to complete the IQ optical constellation. If the models are accurate enough, the features of constellation samples collected from the optical transponder in Z would match the expected optical constellation features obtained with OCATA.
Interface Specification	OCATA runs as part of the network control, and it access: - the telemetry measurements DB to analyse constellation samples - the LSP DB to get the route of the lightpaths

	The results of the analysis are stored in an internal DB and used for algorithms in the SDN controller.
Functional Validation	- Verify that OCATA gets the route of the right lightpath from the LSP DB - Verify that OCATA gets the right constellation samples from the telemetry DB - Verify that OCATA determines the lightpath length and compares to that stored in the LSP DB.
Component Integrations	This component has been integrated with the telemetry system, which provides measurements of optical constellations.
Component KPIs	**Numerical evaluation**: A simulator of a digital coherent system implemented in MATLAB was employed to reproduce scenarios and generate datasets. OCATA DNN models were trained used the generated dataset. The considered scenario consisted of a lightpath passing through 8 ROADMs and a total fibre length of 1,120 km. - Evaluation of the supervised feature-extraction-methodology-based on GMM fitting. We found that constellation points can be accurately modelled as Gaussian distributions for all the considered distances, since the obtained p-value of the test always exceeded the commonly accepted significance level of 0.05. - For lightpath modelling we obtained negligible errors for features μ (max error < 2%) independently of the link length, whereas max error for σ features was is around 30% for low σ values although it decreases when the path length increases, becoming under 15%, which is, in general, a good enough performance to validate the models. - The reconstruction of the features of the non-selected constellation points showed an accuracy of 97%. - Lightpath length analysis showed average error for lightpath estimation < 5% for lightpaths over 500Km and average error for estimation of number of hops of the lightpath < 5%. - Algorithms for failure detection based on OCATA are able to detect filter related failures under non-ideal network conditions in few hours after the degradation starts and achieved anticipation of more than one day before the hard-failure. **Experimental evaluation**: The models were evaluated using experimental measurements and showed that the Euclidean distance comparing features from experimental and OCATA IQ constellation samples is below 0.1 in all the cases and the maximum relative error on the average σ features after different number of spans is below 14%.
Status, availability, repository	The pseudocode of the algorithms together with a thorough description, and the performance evaluation has been reported in D4.3. The main methodology is available in [Rui22]. Algorithms have been experimentally assessed in collaboration with HHI and INF [Dev23a]. Applications have been developed for failure management in [Dev23b]

Additional Remarks	No contribution to standard/open source

REFERENCES

[BBF] Broadband Forum YANG models on GitHub: https://github.com/BroadbandForum/yang

Boi24 F. Boitier et al, “Monitoring of Chromatic Dispersion in Multiband Access and Metro Converged Optical Network”, ECOC’2024 Frankfurt, Germany 22-26/09/2024.

[Cas24a] R. Casellas et al., "Multi-Domain Transparent Service Provisioning in Multi-Band Optical Networks," 2024 24th International Conference on Transparent Optical Networks (ICTON), Bari, Italy, 2024

[Cas24b] Ramon Casellas, Laia Nadal, Ricardo Martinez, Ricard Vilalta, Raul Muñoz, and Michela Svaluto Moreolo, "Photonic device programmability in support of autonomous optical networks," J. Opt. Commun. Netw. 16, D53-D63 (2024)

[CMIS] “CMIS” [Online]. Retrieved April 27, 2022, https://www.oiforum.com/wp-content/uploads/OIF-CMIS-05.2.pdf

[Del19a] C. Delezoide et al., "Automated Alignment Between Channel and Filter Cascade," in Optical Fiber Communication Conference (OFC) 2019, OSA Technical Digest (Optical Society of America, 2019), paper Th2A.48.

[Del19b] C. Delezoide et al., “Marginless Operation of Optical Networks,” J. Lightwave Technol. 37, 1698-1705 (2019)

[Dev23a] M. Devigili, D. Sequeira, C. Santos, M. Ruiz, B. Shariati, N. Costa, A. Napoli, J. Pedro, J. Fischer, and L. Velasco, "Experimental Validation of Deep Learning-based Models for Optical Time Domain Analysis," in Proc. Conference on Lasers and Electro-Optics (CLEO), 2023.

[Dev23b] M. Devigili, M. Ruiz, S. Barzegar, N. Costa, A. Napoli, J. Pedro, and L. Velasco, "Degradation Detection and Severity Estimation by Exploiting an Optical Time and Frequency Digital Twin," in Proc. Optical Fiber Communication Conference (OFC), 2023.

[Dut22] Eric Dutisseuil, Arnaud Dupas, Alexandre Gouin, Fabien Boitier, Patricia Layec, "Hitless Transmission Baud Rate Switching in a Real-Time Transponder Assisted by an Auto-Negotiation Protocol," OFC 2022

[Fer20] A. Ferrari et al., “GNPy: an open source application for physical layer aware open optical networks”. Journal of Optical Communications and Networking, Vol. 12, Issue 6, 2020, Pp. C31-C40, 12(6), C31–C40. https://doi.org/10.1364/JOCN.382906

[Gio23] A. Giorgetti, A. Sgambelluri, F. Cugini, E. Kosmatos, A. Stavdas, J. M. Martinez-Caro, P. Pavon, O. Gonzalez De Dios, R. Morro, L. Nadal, R. Casellas "Modular Control Plane Implementation for Disaggregated Optical Transport Networks with Multi-band Support", ECOC2023

[GNPy] GNPy [Online]. Retrieved October 7, 2022, from https://gnpy.readthedocs.io/en/master/

[Gon23a] P. Gonzalez, L. Velasco and M. Ruiz, "A Distributed Telemetry Architecture for Optical Networks," in Proc. IEEE International Conference on Transparent Optical Networks (ICTON), 2023.

[Gon23b] P. Gonzalez, R. Casellas, J-J Pedreno-Manresa, A. Autenrieth, F. Boitier, B. Shariati, J. Fischer, M. Ruiz, J. Comellas, and L. Velasco, "Distributed Architecture Supporting Intelligent Optical Measurement Aggregation and Streaming Event Telemetry," in Proc. Optical Fiber Communication Conference (OFC), 2023.

[Gou21] A. Gouin, A. Dupas, Ll. Gifre, A. Benabdallah, F. Boitier, P. Layec, "Real-time optical transponder prototype with auto-negotiation protocol for software defined networks," Journal of Optical Communications and Networking, vol. 13, no 9, p. 224-232, 2021

[GNMI] gRPC Network Management Interface (gNMI) [online], Retrieved October 5, 2022, from https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md

[IBN] L. Velasco, S. Barzegar, F. Tabatabaeimehr, and M. Ruiz, "Intent-Based Networking for Optical Networks [Invited Tutorial]," IEEE/OPTICA Journal of Optical Communications and Networking (JOCN), vol. 14, pp. A11-A22, 2022.

[K8s] Kubernetes [Online]. Retrieved October 5, 2022, from https://kubernetes.io/docs/home/

[KAFKA] [Online] https://kafka.apache.org/

[Kos23] E. Kosmatos et al., "SDN-enabled path computation element for autonomous multi-band optical transport networks," in Journal of Optical Communications and Networking, vol. 15, no. 11, pp. F48-F62, November 2023, doi: 10.1364/JOCN.492244.

[Man21] C. Manso et al., “TAPI-enabled SDN control for partially disaggregated multi-domain (OLS) and multi-layer (WDM over SDM) optical networks [Invited]”. Journal of Optical Communications and Networking, 13(1), 2021, pp. A21–A33. https://doi.org/10.1364/JOCN.402187

[Mor24] R. Morro et al,. “Field Trial of Transparent Multi-band Multi-domain Disaggregated IPoWDM Networks”, ECOC’2024 Frankfurt, Germany 22-26/09/2024.

[OIF] “OIF” [Online] https://www.oiforum.com/

[ONF] Open Networking Foundation [Online]. Retrieved October 5, 2022, from https://opennetworking.org/

[ONOS] "ONOS" [Online], https://opennetworking.org/onos/, https://github.com/opennetworkinglab/onos

[OpenROADM] OpenROADM [Online], http://openroadm.org/

[P4] “P4” [Online]. Retrieved October 5, 2022, https://p4.org/

[Pav15} P. Pavon-Marino and J. L. Izquierdo-Zaragoza, “Net2plan: An open source network planning tool for bridging the gap between academia and industry,” IEEE Netw, vol. 29, no. 5, pp. 90–96, Sep. 2015, doi: 10.1109/MNET.2015.7293311.

[RESTCONF] A. Bierman, M. Björklund and K. Watsen, RFC 8040 "RESTCONF Protocol", January 2017

[Rui22] M. Ruiz, D. Sequeira, and L. Velasco, "Deep Learning -based Real-Time Analysis of Lightpath Optical Constellations [Invited]," IEEE/OPTICA Journal of Optical Communications and Networking (JOCN), vol. 14, pp. C70-C81, 2022.

[SAI] "Switch Abstraction Interface" [Online] https://github.com/opencomputeproject/SAI

[Sga24] A. Sgambelluri et al., "Failure recovery in the MANTRA architecture with an IPoWDM SONiC node and 400ZR/ZR+ pluggables," in Journal of Optical Communications and Networking, vol. 16, no. 5, May 2024.

[SONIC] "SONiC" [Online] https://sonic-net.github.io/SONiC/

[Swe22] N. L. Swenson, “Open XR Concept Introductory White Paper,” Tech. rep., Open XR Forum (2022).

[TAPI] “ONF Transport API SDK” [Online], retrieved December 9, 2022 from https://github.com/OpenNetworkingFoundation/TAPI/releases/

[Vel23a] L. Velasco, P. Gonzalez, and M. Ruiz, "An Intelligent Optical Telemetry Architecture," in Proc. Optical Fiber Communication Conference (OFC), 2023.

[Vel23b] L. Velasco, S. Barzegar, and M. Ruiz, "Using a SNR Digital Twin for Failure Management," in Proc. IEEE International Conference on Transparent Optical Networks (ICTON), 2023.