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 D4.1

Infrastructure Control and Service Management architecture, initial requirements, and existing frameworks

Editor O. González de Dios (TID)

Contributors CTTC, TID, UC3M, TIM, INF-P, ADVA SE, NBLF, CNIT, CNR, UPC, OLC-E, ELIG, TuE, PLF

Version 1.0

OBJ

REVISION HISTORY

Revision	Date	Responsible	Comment
0.1	Sept, 9, 2022	Oscar Gonzalez	Initial ToC
0.2	Oct, 18,2022	ALL	First integrated version before Plenary meeting
0.3	Nov, 8,2022	ALL	Integrated version with missing contributions, ready for review in WP4 call
0.4	Nov, 15, 2022	ALL	Sections reviewed.
0.5	Nov, 21, 2022	ALL	Reviewed version.
0.6	Nov 28th, 2022	ALL	Clean-up, updated reviews, Executive summary, conclusions
1.0	Dec 20th, 2022	ALL	Final version with reviewed sections, all missing contributions, updated references and updated executive summary and conclusions.

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
ADVA SE	ADVA Optical Networking SE	José Juan Pedreño Manresa, Achim Autenrieth
CNIT	CNIT	Davide Scano, Filippo Cugini
CNR	CNR	Alessio Giorgetti
ELIG	E-lighthouse Network Solution	Pablo Pavón Mariño José-Manuel Martínez-Caro
NBL	Nokia Bell Labs	Fabien Boitier, Patricia Layec
UPC	Universitat Politecnica de Catalunya	Luis Velasco, Marc Ruiz, Jaume Comellas, Salvatore Spadaro, Davide Careglio, Josep Prat
OLC-E	OpenLightComm Europe s.r.o.	Alexandros Stavdas, Evangelos Kosmatos, Christos Matrakidis, Dimitris Uzunidis
PLF	pureLiFi Ltd	Rui Bian
INF-P	Infinera Unipessoal Lda	João Pedro

GLOSSARY

Acronyms	Description	Acronyms	Description
1:1	One to one	NoSQL	Not only SQL - Non-relational
5G	Fifth Generation	NS	Network Service
5GPPP	5G Infrastructure Public Private Partnership	O/E/O	Optical to Electrical to Optical
6G	Sixth Generation	OaaS	Optimisation-as-a-Service
AA	API Aggregation	OCP	Open Compute Project
AAI	Active and Available Inventory	ODL	OpenDaylight
AI	Artificial Intelligence	ODN	Optical Distribution Network
AP	Access Point	ODTN	Open and Disaggregated Transport Network
API	Application Programming Interface	OIF	Optical Internetworking Forum
ASIC	Application Specific Integrated Circuit	OLS	Open Line system
B2B	Business to Business	OLT	Optical Line Terminal
B5G	Beyond 5G	OMB	Optical Multi-Band
B5G-ONP	Beyond 5G - Open Network Planner	ONAP	Open Network Automation Platform
BBF	Broadband Forum	ONF	Open Networking Foundation
BER	Bit Error Ratio	ONL	Open Network Linux
BF	Bloom Filter	ONOS	Open Network Operating System
BGP	Border Gateway Protocol	ONU	Optical Network Unit
CLI	Command Line Interface	ONP	Open Network Planner
CMIS	Content Management Interoperability Services	OPCE	Optical Path Computation Element
CMS	Count-Min Sketch	OptC	Optical Controller
CNF	Cloud-native Network Function	OS	Operative System
COE	Container Orchestration Engines	OSA	Optical Spectrum Analysers
Conc	Concurrent	OSM	Open-Source Mano
CPU	Central Processing Unit	OSNIR	Optical Signal-To-Noise plus Interference Ratio
CRD	Custom Resource Definition	OSNR	Optical Signal-To-Noise Ratio
CRUD	Create, Read, Update and Delete	OSPF	Open Shortest Path First
CS	Connectivity Service	OTs	Open Terminals
DB	Database	OTSi	Optical Tributary Signal
DC	Data Center	P2P	Point-To-Point
DCAE	Data Collection, Analysis, and Events	P2MP	Point-To-MultiPoint
DCO	Digital Coherent Optics	PCE	Path Computational Engine
DevOps	Development and Operations	PckC	Packet Controller
DHCP	Dynamic Host Configuration Protocol	PINS	P4 Integrated Network Stack
DNN	Deep Neural Network	PL	Physical Layer
DNS	Domain Name System	PLA	Physical Layer impairment Aware
DSP	Digital Signal Processing	PLI	Physical Layer impairment
DSR	Digital Signal Rate	PoM	Policy Manager

DWDM	*Dense Wavelength Division Multiplexing*	***PON***	*Passive Optical Network*
E2E	*End-To-End*	***PSU***	*Power Supply Unit*
EMS	*Element Management System*	***QoS***	*Quality of Service*
ESO	*European Standards Organization*	***QoT***	*Quality of Transmission*
ETSI	*European Telecommunication Standards Institute*	***REST***	*Representational State Transfer*
ETSI MANO	*ETSI NFV Management and. Orchestration*	***RMSA***	*Routing, Modulation and Spectrum Assignment*
Excl	*Exclusive*	***RO***	*Resource Orchestrator*
FEC	*Forward Error Correction*	***ROADM***	*Reconfigurable Optical Add/Drop Multiplexer*
FPM	*Forwarding Plane Manager*	***RPC***	*Remote Procedure Call*
GMPLS	*General Multiprotocol Label Switching*	***RSA***	*Routing and Spectrum Assignment*
gNMI	*gPRC Network Management Interface*	***RSS***	*Received Signal Strength*
gNOI	*gRPC Network Operations Interface*	***SAI***	*Switch Abstraction Interface*
gPRC	*gPRC Remote Call Procedure*	***SBI***	*Southbound Interface*
GUI	*Graphical User Interface*	***SCTP***	*Stream Control Transmission Protocol*
HLL	*Hyper-LogLog*	***SDC***	*Service Design and Creation*
HrC	*Hierarchical Controller*	***SDK***	*Software Development Kit*
HTTP	*Hypertext Transfer Protocol*	***SDN***	*Software Defined Networking*
HW	*Hardware*	***SRG***	*Shared Risk Group*
IBN	*Intent-Based Networking*	***SIP***	*Service Interface Point*
ID	*Identifier*	***SMA***	*Spectral and Modulation Assignment*
IDS	*International Data Spaces*	***SNMP***	*Simple Network Management Protocol*
IETF	*Internet Engineering Task Force*	***SO***	*Service Orchestration*
IGP	*Interior Gateway Protocol*	***SONiC***	*Software for Open Networking in the Cloud*
IIoT	*Industrial IoT with cloudification*	***SQL***	*Structured Query Language*
ILA	*In-Line Amplifier*	***SSID***	*Service Set Identifier*
IoT	*Internet-of-Things*	***SW***	*Software*
IP	*Internet Protocol*	***TAI***	*Transponder Abstraction Interface*
IPFIX	*Internet Protocol Flow Information eXport*	***TAPI***	*Transport API*
IS-IS	*Intermediate System to Intermediate System*	***TCP***	*Transmission Control Protocol*
IT	*Information Technology*	***TDM***	*Time Division Multiplexing*
JSON	*JavaScript Object Notation*	***Te2e***	*Time end-to-end*
K8s	*Kubernetes*	***TFS***	*TeraFlowSDN*
KPI	*Key Performance Indicator*	***TIP***	*Telecom Infra Project*
LAG	*Link Aggregation Protocol*	***TP***	*Transponders*

LiFi	*Light Fidelity*	***Tx***	*Transmission*
LKM	*Loadable Kernel Modules*	***UDP***	*User Datagram Protocol*
LLDP	*Link Layer Discovery Protocol*	***UE***	*User Equipment*
LSP	*Label Switched Path*	***UI***	*User Interface*
MANO	*Management And Orchestration*	***VCA***	*VNF Configuration and Abstraction*
MB	*Multi-Band*	***VIM***	*Virtualized Infrastructure Manager*
MC	*Media Channel*	***VNF***	*Virtualized Network Function*
MEC	*Multi-access Edge Computing*	***VNFM***	*Virtualized Network Function Manager*
ML	*Machine Learning*	***VRF***	*Virtual Routing and Forwarding*
MON	*Monitoring*	***VTI***	*Virtual Tunnel Interface*
MPLS	*Multiprotocol Label Switching*	***WiFi***	*Wireless Fidelity*
MSO	*Master Service Orchestrator*	***WP***	*Work Package*
N2VC	*Network Service to VNF Communication*	***XGS***	*Symmetric 10G-PON*
NBI	*North Bound Interface*	***XR Optics***	*Infinera Technology for point to muti-point optics*
NCACM	*Network Configuration Access Control Model*	***BM***	*Burst-Mode*
NE	*Network Element*	***C-RAN***	*Centralized-Radio Access Network*
NF	*Noise Figure*	***OLT***	*Optical Line Terminal*
NFVI	*Network Function Virtualization Infrastructure*	***OSS***	*Operation Support Systems*
NFVO	*Network Function Virtualization Orchestration*	***VLAN***	*Virtual Local Area Network*
NMC	*Network Media Channel*	***WB***	*White-Box*
NOS	*Node Operating System*	***YANG***	*Yet Another Next Generation*

EXECUTIVE SUMMARY

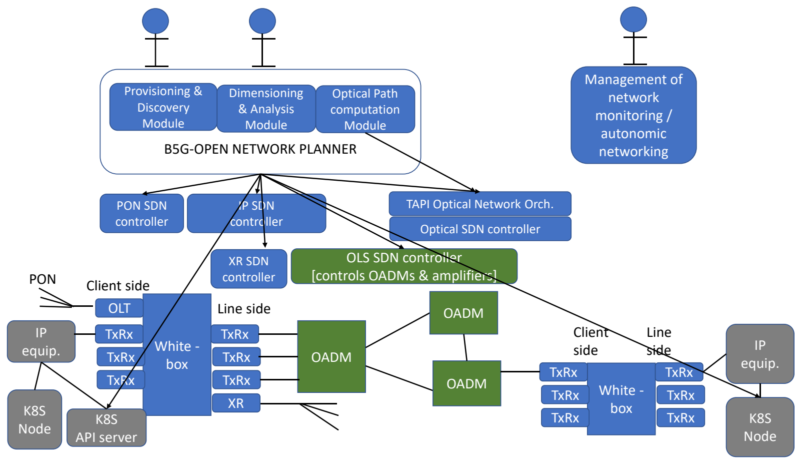
This document presents in detail the Infrastructure Control and Service Management platform architecture to be implemented in the B5G-OPEN Control, Orchestration and Telemetry System (referred to as the *control plane*, for short), along with a set of initial requirements, and existing frameworks. Such platform is aimed at supporting B5G-OPEN key features and goals:

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.

After the introduction in Section 1, Section 2 presents the B5G-OPEN set of **Control and Service Management use cases**, carefully developed within WP2 and extended to the control plane. The use cases target the **provisioning of services over the B5G-OPEN domainless multi-band network**, the **discovery** of the existing resources and topology, **planning** of the network resources, advance use cases to support **fault management**. In addition, use cases for **autonomous network operation** are also described.

B5G-OPEN **Control plane services** documented in section 3 include **point-to-point** optical connectivity, **point-to-multipoint** XR connectivity, **IP link** provisioning, **B5G-OPEN Slice** and **telemetry** services.

To realise such services, the control plane architecture must comply with the following set of requirements: perform **context, connectivity, topology and physical impairments discovery** and **Asynchronous** notification of topology and connectivity object changes


Provisioning & Discovery Module
Dimensioning & Analysis Module
Optical Path computation Module
B5G-OPEN NETWORK PLANNER
Management of network monitoring / autonomic networking
PON SDN controller
IP SDN controller
TAPI Optical Network Orch.
Optical SDN controller
XR SDN controller
OLS SDN controller [controls OADMs & amplifiers]
PON
Client side
OLT
Line side
IP equip.
TxRx
White - box
XR
OADM
K8S Node
K8S API server
Client side
Line side

The document **describes in detail the proposed control plane architecture,** summarized in section 5 and shown in the figure above. The major parts of the architecture include: **service orchestration and planning** (sections 6, 7 and 8), **packet optical-integration** systems (section 9), **telemetry** (section 10) and **intent-based networking** (section 11).

An initial proposal of the necessary interfaces and protocols to be used among the different control plane components and towards external systems and network elements is presented in Section 12. The detailed specification will be provided in the next deliverable.

In a nutshell, this document introduces a number of control plane innovations to be developed within WP4, namely:

- Control of optical multiband network, that is, to exploit the multiband capabilities of optical devices, both transmission and switching elements.
- Physical layer impairments of multiband optical networks, where the increasing number of non-linear effects need to be considered in the control plane for a better management and orchestration of the network and services.
- Control of transparent multi-domain, that is, the ability to setup connections in a transparent manner, across multiple domains and network segments.
- Packet/optical integration, where the gradual introduction of pluggable interfaces directly in the switches need to be incorporated into the control plane.
- Access/Metro integration, with control of different access technologies (PON, LiFi).
- Telemetry, where the ability to continuously monitor the network is critical as a first step to diagnose its behaviour and take decisions in case of malfunctioning.
- External planning tools, where multiple algorithms can be defined to optimise network behaviour and operations, making use of quality data at different levels (optical and packet level).
- Network automation, which ultimately implies autonomous operations of the network to further reduce human intervention, leading to a self-managed network based on telemetry data and historical experience, along with AI/ML techniques for closing the observe-decide-act loop.

TABLE OF CONTENTS

List of figures

1 Introduction

This document details the Infrastructure Control and Service Management architecture to be implemented in the **B5G-OPEN Control, Orchestration and Telemetry System (*control plane* for short)**, along with a set of initial requirements, and existing frameworks. The document comprises three main parts, each one containing multiple chapters.

The key B5G-OPEN goals and major innovations, as outlined in the proposal, include the design of a network architecture featuring multi-band, optical continuum, integrated access, end-to-end network orchestration and autonomous operations. The first part includes Section 2 which overviews a set of initial requirements and use cases, carefully developed with WP2, and extended to the control plane. Section 3 further develops on the *services* that the control plane of B5G-OPEN must support, for instance, point-to-point optical connectivity, point-to-multipoint XR connectivity, IP link provisioning, telemetry and optical topology services, etc.

In the second part of the document, Section 4 provides a thorough review of existing frameworks in the literature, some of them will participate in the design of the control plane architecture. Examples include an overview of most popular control plane frameworks and network operating systems, different tools for implementing network telemetry systems, main orchestration frameworks and Quality of Transmission (QoT) estimation tools.

The third part of the document comprises multiple chapters devoted to the definition of different parts of the control plane architecture. In this sense, Section 5 overviews the major parts of the architecture, including service orchestration and planning, optical-packet integration systems and telemetry and intent-based networking. Section 6 introduces the SDN control of optical multi-band networks, while Section 7 explains the control of different access technologies (PON, LiFi, etc). Section 8 is devoted to the IT and network resource orchestration platform, and Section 9 to the packet-optical integration. Section 10 overviews the architecture of the telemetry system, in charge of monitoring the quality of network connectivity at the optical level and packet flow level. Section 11 overviews the requirements and architecture for autonomous network operations. Next, Section 12 provides further details on the interfaces and systems participating in the whole control plane architecture, described from Section 5 until Section 11.

Finally, Section 13 concludes this document with a summary of its main contributions to the B5G-OPEN project and next steps.

2 INITIAL REQUIREMENTS AND USE CASES

The B5G-OPEN control and management plane requirements follow the need to support the project's key features showed in the following table and derived from Deliverable D2.1:

Key Feature	Description
Multi-Band operation	Availability of bands O, E, S, C, L in optical fibres to provision: a) the required capacity, and b) service based on requirements
Optical continuum	Operate connectivity extending the principles of optical bypassing of nodes in the Multi-Band B5G-OPEN network, allowing 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. 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

Table 1.1: B5G-OPEN key features

Multi-Band technologies offer the potential to facilitate the implementation of an optical bypass at the central office for selected traffic, leading to the concept of **optical continuum** and removing unnecessary electronic intermediate terminations. Based on the requirement of the traffic that need to be transported, the electronic termination may be located at different points of the network. Moreover, components such as transmission systems and optical switches/ROADMs may operate in one or multiple bands depending on vendors' choices and carriers' network implementation. Clearly, such an architecture goes beyond the concept of network segmentation since no clear and uniform demarcation points are identifiable. Also, the availability of programmable pluggable modules implementing sophisticated functions that can be fitted into Ethernet white boxes makes weaker the boundaries between network segments. In fact, the same boxes, having packer layer switching capabilities, could host both programmable optical pluggable modules and/or modules implementing access functionalities (PON transceivers) resulting in packet/optical devices **integrating aspects typical for both access and metro/regional segments**. Moreover, it can be easily foreseen that, in the near future, other pluggable modules will appear implementing functions now typical of monolithic devices. In such a context, the B5G-OPEN control plane must overcome the traditional approach where every network segment is under the control of a dedicated controller, eventually coordinated by a parent one. Also, the rigid subdivision of control functions dedicated to a specific network layer (packet or optical) must be replaced by a more flexible approach to simplify the interaction with the novel pluggable modules that, despite being fitted into a packet device, need configurations typical of other network segments. To reach this goal, the B5G-OPEN control plane could leverage concepts from the cloud environment and adopt a microservice architecture based on containers that offer easy and rapid introduction of new features as soon as the data plane technology evolves and seamless migration of functionalities between modules to follow network evolutions, e. g. when pluggable modules integrating new features are introduced. A containers' orchestration solution such as Kubernetes could provide even more flexibility in terms of scalability and coordination among all the control plane modules.

Such a modular architecture must rely on standard and open interfaces between the control plane functions and towards the devices. The Open Networking Foundation (ONF) [ONF] has developed a standard API called Transport API (TAPI) which has become a de-facto standard in the field of SDN controllers. At device level, instead, two data models are catalysing the attentions of the market, with different degree of adoption mainly depending on the device type (X-ponder or ROADM): OpenConfig and OpenROADM. In particular, to support Multi-Band, the control plane must take into account all the constraints required to provision end-to-end media channels (i.e. spectral resources) in such an environment and provide procedures such as automatic inventory management, Multi-band (MB) impairment-aware path computation, MB topology abstractions and automatic identification of candidate pre-validated configurations. Generally speaking, the above-mentioned existing data models are already (or require very little adaptation for) supporting MB networks. What needs to be defined is a kind of "rule of thumb" for using them in a common way. Nevertheless, path computation algorithm must be extended to support the presence of several transmission bands. Also, new protection/restoration schemes must be developed to exploit the larger bandwidth available thanks to MB.

Since the B5G-OPEN reference network architecture is intrinsically 'domain-less', in the sense that, as stated above, there are no clear demarcation points between the network segments that today correspond to the access, metro and part of the core, the B5G-OPEN control plane must implement **end-to-end orchestration** of services from the Access Point to the Cloud node with new and innovative models beyond the traditional packet over optical approach. In this field, several open-source initiatives are available with different degrees of complexity and flexibility, but the "domain-less" perspective of the B5G-OPEN network requires a re-consideration of the traditional MANO architecture and a different approach to the control modules interface, also adopting a lightweight virtualization method relying on container management or serverless computing.

The high bandwidth availability guaranteed by Multi-Band is an opportunity for efficient network sharing at the optical layer. Also, innovative pluggable modules in the access segment (e. g. Tibit) and the novel optical point-to-multipoint pluggables (e. g. XR-optics) are key drivers for sharing aggregation resources. Network slicing allows taking full advantage from the opportunity offered by these technologies and, therefore, it is another feature that the B5G-OPEN control plane must support. For what concerns XR-optics, even if developing a control plane for it is not an objective of the project, the B5G-OPEN control plane should support network services exploiting the capabilities offered by the novel point-to-multipoint pluggable transceivers. Currently, among the most used data model for optical networks, only T-API encompasses the concept of point-to-multipoint connectivity, while device models like OpenConfig and OpenROADM require specific extensions.

Intent-based networking and **autonomous operations** are mandatory for the innovations envisioned by the project. The control plane architecture should provide an abstracted view of the network and close-control loops operated at various levels, from device to network, must be implemented in support of distributed AI/ML techniques to enable coordinated decision-making across control domains. Advances in optical technology with the introduction of coherent transmission thanks to DSP-based transceivers offered unprecedent opportunities for massive monitoring of the physical layer to detect and proactively correct soft-failures. However, collection of real time monitoring data can potentially saturate both the bandwidth of management interfaces and the CPU power of the servers. The B5G-OPEN control plane should adopt a collaborative approach between the network controller and node agents for the implementation of the streaming telemetry systems to reduce the amount of monitored data and an architecture based on the publish/subscribe paradigm to selectively address data only to the interested modules.

2.1 INITIAL CONSIDERATIONS

The first main functionality to be provided by B5G-OPEN Control plane is the **Multi-Band Optical Network operation.** B5G-OPEN Use cases are based on ONF TF-547 v1.1 [TR-547], provides a set of use cases for control and management of optical networks based on TAPI, which have been adopted by Telecom Infra Project (TIP) MUST Optical sub-group. This is motivated by the fact that B5G-OPEN partners are actively participating in such standardization activities and are in an excellent position to provide feedback on implementation, along with preliminary considerations regarding their applicability to multi-band networks (since TR-547 mostly considers single band operation),

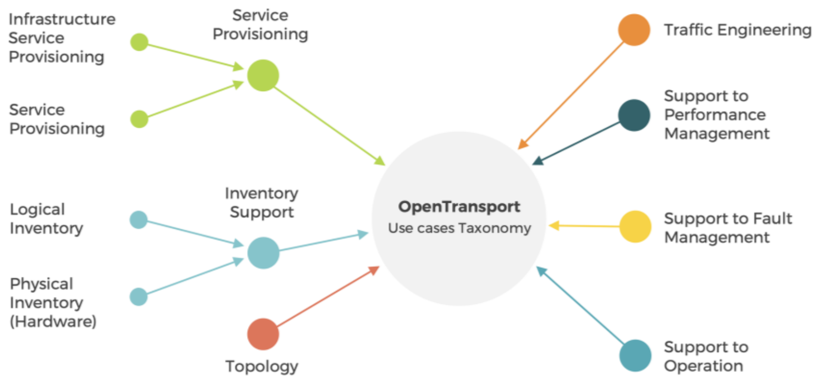


Fig. 2.1 Use case taxonomy (source TIP)

A set of use cases that showcase the B5G-OPEN COM capabilities are described below:

2.2 SERVICE PROVISIONING/ACTIVATION OVER DOMAINLESS MULTI-BAND NETWORK

This set of use cases comprise setting up and tearing down the required Connectivity services over a multi-band enabled network, including point-to-point and point-to-multi-point, which are described in section 3. The provisioning operation needs to be triggered via programmatic interface. The request might come directly from the network operator, or via another component of B5G-OPEN Control and Management plane.

It is important to consider that, even though the network can be comprised of multiple segments (access/aggregation/core), B5G-OPEN control plane will treat the network as “domainless” and will allow connectivity to be requested among any point of the network. This might comprise that the service can cross multiple OLS.

The services to activate are detailed in section 3.

[REQ.PROV.1] The provisioning should consider multiple constraints in the requests, in order to fulfil the requirements of the applications described in WP2. Those constraints should consider **bandwidth**, **delay**, **jitter** and **reliability.**

2.3 Discovery Use cases

This group of use cases targets the retrieval of information available from B5G-OPEN control, orchestration and telemetry system including topology, service-interface-points, connectivity-services and connections. This section addresses Discovery Aspects from the point of view of an end user, typically understood as a human actor or an Operator Business Support System or Operations Support System. This does not exclude the fact that, internally and considering partially disaggregated architectures with a dedicated OLS controller, B5G-OPEN control plane needs to obtain Open transponders information directly from the open terminals (OTs) devices and IP/Optical whiteboxes using Netconf/OpenConfig, which is part of the discovery process

The following operations need to be supported:

[REQ.DIS.1] *Context discovery*: A "context" in an abstraction that allows logical isolation and grouping of network resources. The context in B5G-OPEN will expose the set of Service-interface-points, which represent the available customer-facing point from where connectivity network services can be requested.

[REQ.DIS.2] *Connectivity discovery*: B5G-OPEN will automatically discover network connectivity services in the DSR layer, which model digital signals and PHOTONIC_MEDIA layer, in particular media channels as per [ITU-T G.872]. The list of connectivity services created within the context will need to be provided.

[REQ.DIS.3] *Topology discovery*: The control plane needs to provide a dynamic representation of the network based on a synchronization with the network elements. The topology is described in terms of nodes and links that enable the forwarding capabilities of the network resources. Nodes are an abstraction of the forwarding capabilities of a network element and will contain collection of ports and the potential to enable forwarding between those ports. The links are an abstract representation of the adjacency between nodes in the topology.

[REQ.DIS.4] *Asynchronous notification of topology changes:* B5G-OPEN control system needs to send events exposing changes in links, nodes, and node edge points. These events are triggered, for example, upon network failures or due to network upgrades.

[REQ.DIS.5] *Asynchronous notification of connectivity object changes*: B5G-OPEN control system needs to send events exposing changes in the connectivity services.

[REQ.DIS.6] *Physical layer impairments discovery*: WP3 has defined an impairment model for the muti-band optical layer. The B5G-OPEN control plane needs to retrieve from the network the necessary information to feed the model specified in Deliverable D3.1.

2.4 Design use cases

There use cases are aimed at providing support for the Network Operator to design the packet/optical network. The B5G-OPEN COM will provide:

- **[REQ.UC.DES.1]** *Network Dimensioning/ Capacity planning*: These use cases correspond to a network design phase, when the network needs to be conceived from scratch (greenfield design), or new equipment should be added to an existing network (brownfield design). The output produced is commonly composed of a bill-of-materials of the network equipment to deploy, as well as traffic engineering policies to enforce. Optimization problems involved are typically large, considering aspects like fault tolerance targets, and worst-case latency constraints policies. In this use cases, it is accepted algorithm running times in the order of minutes or even longer, since their results are not applied immediately to the network.
- **[REQ.UC.DES.2]** *Network provisioning:* These use cases correspond to the on-demand allocation of resources, to be completed in nearly real time, and that end-up in the reconfiguration of the existent equipment (i.e. but not the commissioning of new equipment in the network). For instance, capacity on-demand use cases result in situations where new incoming service requests should be provisioned, with target times in the order of seconds. These time constraints ay become more stringent when the algorithms should satisfy resource allocations corresponding to network recovery use cases.

2.5 Support to Fault Management Use cases

Pain-points of Network operators include understanding the cause of services not working as expected. The B5G-OPEN Control and Management system will provide advanced functionalities for the network operator in terms of supporting the fault management process.

- **[REQ.UC.FM.1]** Degradation detection/location in single domain/multi-domain

This use case targets at modelling and evaluating the performance of lightpaths traversing multiple domains. To this end, models that characterize an optical connection within an optical domain can be shared, so models for multiple domains lightpaths can be created.

Armed with those models, telemetry measurements are received and compared against what is expected from the models. This strategy allows detecting degradations, identifying them, evaluating its severity, and localizing the cause of the failure causing the degradation.

- **[REQ.UC.FM.2]** Recovery after failure/degradation

This use case complements the previous one by taking action in case that some links and/or optical bands suffer important Quality of Transmission (QoT) degradation. In such a case, the control plane will detect such degradation via telemetry and provision an alternative path and/or band to re-allocate the flows.

2.6 Autonomous Network Operation use cases

In addition to being able to set up the necessary connectivity, B5G-OPEN Control System will provide **autonomous network operation** by having closed closed-control loops at various levels, from device to network.

- **[REQ.UC.AN.1]** Quality assurance based on Intent-based Networking (IBN)

  This use case envisions using IBN-based applications to assure the quality of the multi-band optical transport network. A specific IBN application need to accurately model optical links, nodes and the lightpaths, considering physical layer modeling (noise, filtering…). Real-time telemetry will continuously feed the IBN applications to ensure the awareness of the network state. The IBN applications will be able to anticipate quality issues in the network and take the necessary steps to solve them.

- **[REQ.UC.AN.2]** Automatic assignment of flows to multi-band slices

Thanks to the multi-band capabilities, B5G-OPEN network architecture is able to allocate multiple per-band slices. This use case is aimed at providing an autonomous operation where the control plane identifies the flows and based on AI/ML, takes the decision on assigning flows to a relevant slice. In the case that there are no slices that can guarantee the connectivity needs of the flows, the control plane will automatically create a new slice.

3 B5G-OPEN CONTROL PLANE SERVICES

The considered use cases, related requirements, and detailed network architecture from WP2 have been used as a starting point to identify a set of *Control Plane Services*, to be elaborated within the scope of WP4.

In this context, a *control plane service* is understood as a network service whose lifetime is managed by the B5G-OPEN control plane – in other words, it is responsibility of the B5G-OPEN control, orchestration and telemetry system – and it is established via one or more North Bound Interfaces (NBI), dynamically and upon demand. This can apply regardless of the time scale of the provisioning.

Similar to the aforementioned use case taxonomy, control plane services are macroscopically grouped into *Provisioning & Discovery*, offering APIs and GUI for providing network topology info, and for allowing provisioning use cases. In addition, there are additional services related to *Telemetry* and *Path Computation* along with *Network dimensioning* & *analysis* module. This includes required resource allocation & capacity planning algorithms based on resource occupation information.

This is shown in the Figure 3-1, with a simplified representation of the control plane architecture. The architecture is described later on in Section 5 (overview and main considerations), Section 6 (Optical network control), Section 7 (Access control integration) Section 8 (IT orchestration), Section 9 (for packet/optical integration) and Section 10 (for Telemetry).

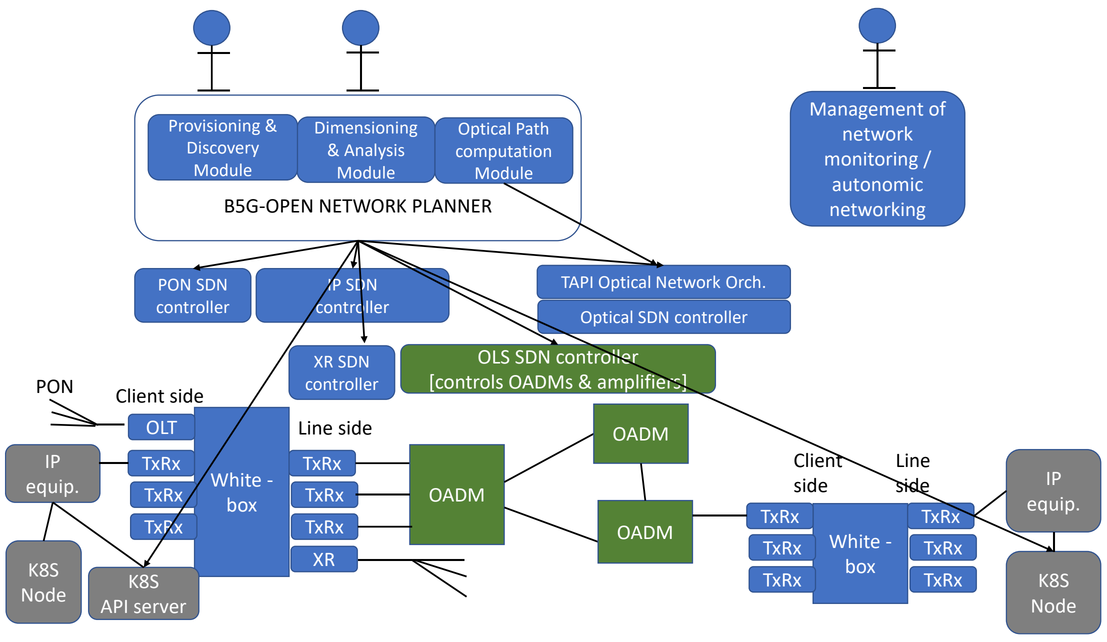

Provisioning & Discovery Module
Dimensioning & Analysis Module
Optical Path computation Module
B5G-OPEN NETWORK PLANNER
Management of network monitoring / autonomic networking
PON SDN controller
IP SDN controller
TAPI Optical Network Orch.
Optical SDN controller
XR SDN controller
OLS SDN controller [controls OADMs & amplifiers]
PON
Client side
OLT
TxRx
White - box
Line side
XR
OADM
IP equip.
K8S Node
K8S API server
Client side
Line side

Figure 3-1 Macroscopic B5G-OPEN architecture and Service instantiation interfaces.

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

3.1 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 (when the configuration focuses on the media channel layer) as shown in Figure 3-2. It mainly involves the provisioning of a media channel (provisioning of raw optical spectrum) within a given optical band, and it is characterized by its *effective frequency slot*. The dynamic provisioning and deployment of the service involves different elements of the control plane architecture (see Section 5). This can be done in an integrated scenario or in a full or partially- disaggregated scenario, via a dedicated Open Line System (OLS) controller, as shown. It is worth mentioning that this service applies at the photonic media layer only and deals with allocation of media channels (variable sized frequency ranges corresponding to optical spectrum). As shown in the figure, macroscopically, it relies on an arrangement of controller(s) -- including the OLS controller in a partially disaggregated scenario – exporting standard interfaces. Further details will later be introduced regarding the final decomposition of the controllers and applicable TAPI North Bound Interfaces specifications and modelling.

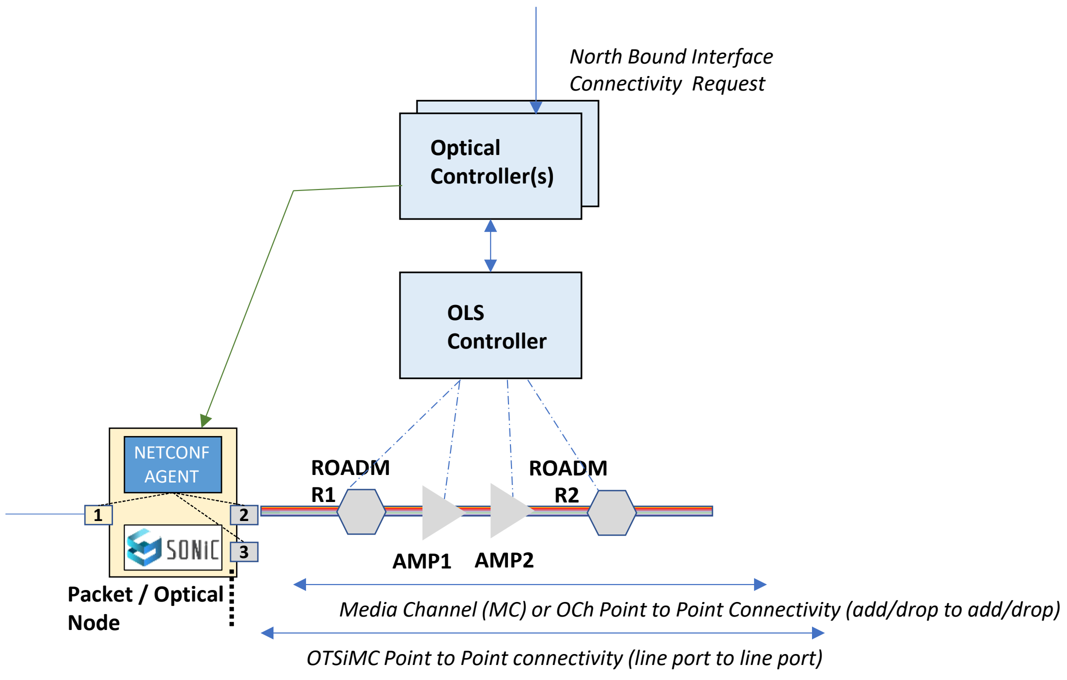


Figure 3-2 Point to Point Optical Connectivity Service.

3.2 Point to Point DSR Connectivity

This service addresses Digital Signal Rate (DSR) provisioning between two stand-alone transceivers or whiteboxes with integrated transceivers. It is part of IP link provisioning between elements (packet/optical nodes) and relates to creating, dynamically and 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).

3.3 Point to Multipoint XR connectivity

This service addresses the provisioning of a point to multipoint connection from a hub to several leaves. The service will be realised by means of OpenXR [Wel21] 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.

3.4 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 3-3, and the relevant list of operations to perform can cover e.g., interface activation, IP address configuration, etc.

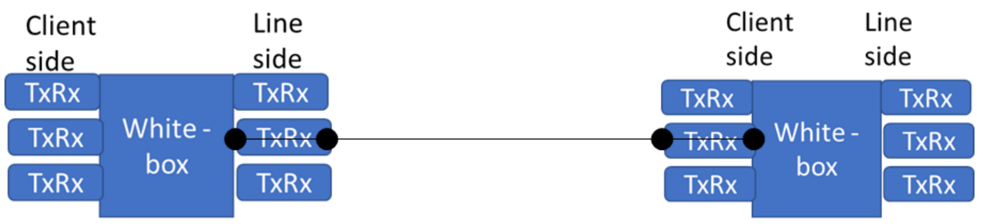


Figure 3-3 IP link provisioning between the whiteboxes.

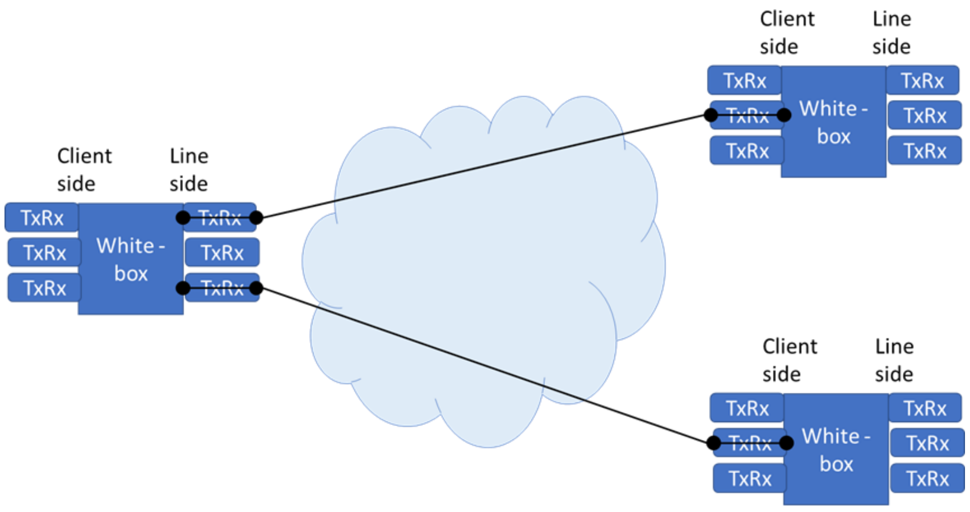


Figure 3-4 multiple IP link provisioning between the whiteboxes using P2MP XR.

3.5 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.

3.6 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 7

3.7 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.

Computing functions are instantiated within computing servers or nodes, and they are interconnected using dynamic network connectivity (thus relying on the previously mentioned services). They may correspond to containers (e.g., Cloud Native Functions, CNF) or Virtualized Network Functions (VNF).

Such service information model shall contain a list of functions, their interconnection, and related constraints in terms of bandwidth and other KPIs. Two critical KPIs to consider are:

- End-to-end latency: measured as the maximum delay between network functions.
- End-to-end estimated jitter: measured as an estimation of e.g., the standard deviation of the end-to-end latency in the connection between network functions.

In this regard, the B5G-OPEN orchestrator needs to be able to provision slices using Kubernetes [K8s] nodes (see Figure 3-5) as follows: given a set of K8sS services to instantiate, and optionally the K8s clusters where they should be instantiated (if not, enough info for optimizing the placement should be given), and a list of service-to-service connections to configure (s1, s2, Gbps, end-to-end KPIs), then compute the optimal service placement, IP capacity, optical capacity needs, satisfying the end-to-end KPIs. Then, the instances of the services in the K8S are automatically provisioned, as well as the planned network resources.

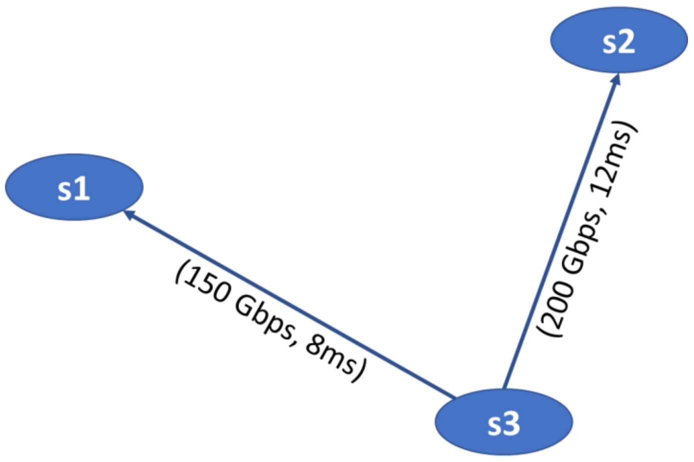


Figure 3-5 Example of slice information model

3.8 OTHER SERVICES

3.8.1 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. The expected behaviour of clients is to connect to the Telemetry System, as described in Section 10.

3.8.2 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.

3.8.3 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. The details are provided in Section 6.5

4 Existing Frameworks

4.1 Control Plane Frameworks

Regarding the control plane, this section reports the open-source initiatives for the SDN control of the network. The B5G-OPEN controller will be required to control not only the disaggregated optical transport network but also the packet-based network (e.g., supporting P4-based devices).

The main open-source initiatives raised from the traditional SDN community are OpenDaylight [ODL] and the Open Network Operating System [ONOS]. Both include some support for the control of optical transport networks. More recently, the TeraFlow project [Vil21] started the development of a new open-source SDN controller specifically oriented toward a cloud-native architecture, currently this controller does not support optical transport networks.

4.1.1 ONOS

The ONOS (i.e., Open Network Operating System) SDN controller is an open-source initiative promoted by the Open Networking Foundation (ONF) with the support of many of the most important telecommunication vendors and operators [ONOS]. ONOS has been designed to meet the scalability and reliability needs of operators wishing to build carrier-grade solutions that leverage the economics of white box merchant silicon hardware while offering the flexibility to create and deploy new dynamic network services with simplified programmatic interfaces. For this purpose, ONOS is based on a modular architecture that facilitates the development and the deployment of new modules such as: novel network applications, additional northbound interfaces, additional southbound drivers, and protocols.

Within the ONF community, ONOS is part of a wider set of initiatives which includes other related projects such as the development of switch operating systems and modules for P4-based hybrid packet/optical devices (i.e., STRATUM, and PINS), the modelling of standard interfaces toward transport networks (i.e., TAPI), the development of a packet-based network emulator (i.e., MININET) and several specific applications developed over the ONOS controller. Such applications are devoted to specific use cases, e.g., the Open and Disaggregated Transport Network (ODTN) project that in the recent years leaded the extension of ONOS to control and monitor disaggregated optical transport networks.

With the practical aim of building up on the ONOS controller components developed within the METRO-HAUL and ODTN projects the B5G-OPEN consortium agreed to select ONOS for the implementation of the functionalities required at the SDN controller level, both for the optical layer as well as the packet and aggregation layer.

4.1.2 OpenDayLight

OpenDaylight (ODL) is a modular open platform for customizing and automating networks of any size and scale. The ODL Project [ODL] arose out of the SDN movement, with a clear focus on network programmability. It was designed from the outset as a foundation for commercial solutions that address a variety of use cases in existing network environments. Indeed, ODL code has been integrated or embedded in more than 35 vendor solutions and apps, and can be utilized within a range of services. It is also at the core of broader open-source frameworks, including ONAP [ONAP].

Also, the ODL controller is organized following a modular architecture where underlying network devices and network applications are all represented as objects, or models, whose interactions are processed within the ODL core.

The ODL controller is not proposed as a part of the B5G-OPEN control plane, however, it could be considered to support already existing deployments in view of the existence of devices already integrated in this platform.

4.1.3 TeraFlow

The TeraFlow project [Vil21] is developing a novel cloud native SDN controller for beyond 5G networks. This new SDN controller is able to integrate with current NFV and MEC frameworks as well as to provide revolutionary features for flow aggregation, management (service layer), network equipment integration (infrastructure layer), and AI/ML-based security and forensic evidence for multi-tenancy. The project proposes an integrated solution for tackling various challenges of B5G networks to support service providers and telecommunication operators in their journey towards future networks.

TeraFlow has launched the first release of the TeraFlow OS SDN controller. It has become an open-source project under the umbrella of ETSI [TFS]. ETSI announced in May 2022 their decision of hosting the recently created TeraFlowSDN open-source group. ETSI is officially recognized by the European Union as a European Standards Organization (ESO). The evolution of this project is being accurately followed by the B5G-OPEN consortium. We will explore collaboration opportunities, such as the joint Teraflow-B5G-OPEN special session that took place in EUCNC2022.

4.2 EXISTING NOS FRAMEWORKS

This section reviews currently available open-source tools and standard models for the implementation of the Network Operating System (NOS) of packet/optical nodes and traditional optical devices such as transponders, ROADMs and OLSs.

4.2.1 Packet/Optical nodes

The recent advances in transmission technology have driven the introduction of coherent pluggable transceivers (i.e., *pluggables*) that can be equipped within packet switching devices thus building up a hybrid device with packet switching and advanced optical transmission capabilities. For example, Digital Coherent Optics (DCO) pluggables are commercially available at rates of 400 Gbps with configurable transmission parameters in different form factors, such as CFP2 and the smaller QSFP-DD 400ZR. The replacement of standalone transponders with pluggables in the packet devices directly connected to the switching elements of the optical network (e.g., ROADMs) drives significant benefits in terms of capital expenditures, power consumption and footprint. Furthermore, it enables a tight integration between packet and optical networks. For example, a single device can provide both intra-data center traffic aggregation and, thanks to coherent pluggables, data center to data center interconnection.

The hybrid nodes should support programmability on both technological sides. Specifically, most advanced solutions can be typically programmed exploiting P4 data plane functionalities and utilizing the P4-Runtime protocol on the packet side, while the optical side that requires simpler configuration is typically programmed operating on the device YANG model. This point can be achieved using a variety of protocols depending on the specific implementation (e.g., NETCONF, gRPC, gNMI). Thus, controlling packet-optical solutions requires a complete Network Operating System (i.e., NOS), that is more complex than traditional software agents employed in standalone transponders [Sga20, Gio20].

This section describes the different open-source initiatives that are currently targeting to the implementation of a NOS for hybrid packet optical nodes.

4.2.2 SONiC

SONiC (Software for Open Networking in the Cloud) is an open-source network operating system based on Linux that can run on switches produced by multiple vendors, based on several ASICs [SONIC]. This solution offers a full-suite of network functionality, (i.e., BGP) that has been production-hardened in the data centers of some of the largest cloud-service providers. It offers the flexibility to create the needed network solutions while leveraging the collective strength of a large ecosystem and community.

SONiC is already widely used in production intra-DC networks and it is also considered a strong candidate for packet-optical nodes although some operational extensions are needed to fill the existing architectural gaps. For example, the current SONiC distribution does not natively support NETCONF and it does not encompass the needed software components to operate on coherent pluggable transceivers. The upgrades needed for the support of hybrid packet/optical nodes are in via of developments in the initiatives reported in the following.

- PINS (P4 Integrated Network Stack): P4 Integrated Network Stack (PINS) is an industry collaboration among ONF, Intel and Google, bringing SDN capabilities and P4 programmability to traditional routing devices that rely on embedded control protocols. Specifically, this project targets the deployment of a dedicated container on network devices running SONiC. It uses P4 to model the pipeline switch abstraction interface SAI [SAI], adds externally programmable extensions to the pipeline and introduces P4Runtime as a new control plane interface for controlling the pipeline.
- GoldStone: this project utilizes many existing open-source components which have been developed in Open Compute Project [OCP] and Telecom Infra Project [TIP] including Open Network Linux [ONL], SONiC, SAI and Transponder Abstraction Interface [TAI] to provide a full-fledged open-source solution. ONL is used as the base operating system and provides a wide range of open network device support. On top of ONL, Kubernetes is employed to enable containerized application management, which realizes flexible and modular software composition. SONiC/SAI is deployed as a fleet of containers when the target hardware comprises Ethernet switch ASIC, whereas TAI is used when the target hardware has coherent transponder components. Because of its modular architecture, Goldstone can be extended to support networking devices, which don't have Ethernet ASIC, but may include conventional transponders, ROADMs or amplifiers in the future.
- Proprietary SONiC distributions: several hardware vendors (e.g., Edgecore, DELL) are currently providing extended SONiC distributions to assure the support of their hardware and to offer advanced application while keeping the NOS openness to the deployment of user applications. Specifically, the support of optical coherent pluggables is progressing slowly within the community SONiC distribution, but it is already provided by the Edgecore SONiC.

Being SONiC based on a modular architecture exploiting the concept of containers, it guarantees high flexibility in terms of deployable features. Specifically, users could implement the required additional features on a custom-built container to be later deployed on the SONiC operating on the network device. Within the B5G-OPEN this flexibility is fundamental because it allows the deployment of the network agents and interfaces toward the SDN controller (e.g., NETCONF agents) building on the work done on previous research projects. Therefore B5G-OPEN project will concentrate on the utilization of SONiC as the primary solution as NOS of packet/optical nodes.

Moreover, among the aforementioned projects, PINS is currently the most active for introducing the full configurability of the packet switching based on P4 and P4Runtime. Its development and deployment steps are well detailed in the SONiC development roadmap periodically published on the main SONiC. Therefore B5G-OPEN project will closely monitor the evolution of PINS for implementing a SDN interface toward the controller.

Finally, since the implementation of the tools enabling the configuration of optical pluggables is slowly progressing in the community SONiC distribution, B5G-OPEN will consider the utilization of proprietary SONiC distributions (EdgeCore) for the control of coherent pluggables.

4.2.3 Stratum

Stratum [Stra] is an open-source silicon-independent NOS for SDN-based networks developed by the Open Networking Foundation [ONF]. It is building an open, minimal production-ready distribution for white box switches. Stratum exposes a set of next-generation SDN interfaces including P4Runtime and OpenConfig, enabling interchangeability of forwarding devices and programmability of forwarding behaviours.

Stratum avoids the vendor lock-in of today’s data planes (i.e., proprietary silicon interfaces and closed software APIs) and enables easy integration of devices into operator networks. It delivers a complete white box switch solution to realize the ‘software defined’ promise of SDN. The Stratum project broadens the scope of SDN to include full lifecycle control, configuration and operations interfaces. Envisioned as a key software component of SDN solutions of the future, Stratum implements the latest SDN-centric northbound interfaces, including P4, P4Runtime, gNMI/OpenConfig, and gNOI. It does not embed control protocols, but instead is designed to support either an external NOS or to work with NOS functions running on the same embedded switch.

Some partners within the B5G-OPEN consortium have experience with this NOS, that however is not considered as the first option because the planned development is more oriented toward the support of the packet side of the device with limited focus toward the support of pluggables. However, it could be considered for operating devices already available at the partners laboratories, especially if advanced features have to be implemented based on P4/P4Runtime.

4.2.4 Proprietary solutions

Besides open-source initiatives, there are some proprietary NOS available on the market offering a standard model of the device that can be used for both device configuration and monitoring. Among these tools, it is worth mentioning the OcNOS system provided by IPinfusion [OCN] because it provides the support for the Cassini Whiteboxes including the utilization of coherent pluggables.

Being this tool not open-source it will be considered only as a backup solution for operating devices already available at the partners laboratories.

4.3 Telemetry Frameworks

This section overviews different frameworks related to telemetry. It covers: 1) protocols and platforms related to measurements and events data streaming, and 2) engines for search and visualization.

4.3.1 IPFIX

Internet Protocol Flow Information Export (IPFIX) is an IETF protocol, as well as the name of the IETF working group defining the protocol [RFC7011]. It was created based on the need for a common, universal standard of export for Internet Protocol flow information from routers, probes and other devices that are used by mediation systems, accounting/billing systems and network management systems to facilitate its services. The IPFIX standard defines how IP flow information is to be formatted and transferred from an exporter to a collector.

The IPFIX standards requirements were outlined in the original [RFC3917]. Cisco NetFlow Version 9 was the basis for IPFIX, the basic specifications for IPFIX are documented in [RFC7011] through [RFC 7015] and [RFC5103].

A pool of Metering Processes collects data packets at one or more Observation Points, optionally filters them and aggregates information about these packets. An Exporter then gathers each of the Observation Points together into an Observation Domain and sends this information via the IPFIX protocol to a Collector. Exporters and Collectors are in a many-to-many relationship: One Exporter can send data to many Collectors and one Collector can receive data from many Exporters.

Like the NetFlow Protocol, IPFIX considers a flow to be any number of packets observed in a specific timeslot and sharing several properties, e.g., “same source, same destination, same protocol”. Using IPFIX, devices like routers can inform a central monitoring station about their view of a potentially larger network.

IPFIX is a push protocol, i.e., each sender will periodically send IPFIX messages to configured receivers without any interaction by the receiver.

The actual makeup of data in IPFIX messages is to a great extent up to the sender. IPFIX introduces the makeup of these messages to the receiver with the help of special Templates. The sender is also free to use user-defined data types in its messages, so the protocol is freely extensible and can adapt to different scenarios.

IPFIX prefers the Stream Control Transmission Protocol (SCTP) as its transport layer protocol, but also allows the use of the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP).

4.3.2 gRPC

gRPC is an open-source high performance Remote Procedure Call (RPC) framework that can run in multiple environments [GRPC22]. It can efficiently connect services with support for load balancing, tracing, health checking and authentication. gRPC was initially created by Google, which used a single general-purpose RPC to connect the large number of microservices running within and across its datacentres for over a decade. In March 2015, Google decided to build a new version and make it open source. The result was gRPC, which is now used in many organizations to power use cases from microservices to the “last mile” of computing (mobile, web, and Internet of Things).

In gRPC, a client application can directly call a method on a server application on a different machine as if it were a local object, making it easier for you to create distributed applications and services. As in many RPC systems, gRPC revolves around the idea of defining a service, specifying the methods that can be called remotely with their parameters and return types. On the server side, the server implements this interface and runs a gRPC server to handle client calls. On the client side, the client has a stub (referred to as just a client in some languages) that provides the same methods as the server.

gRPC clients and servers can run and talk to each other in a variety of environments and can be written in one of the many languages supported by gRPC. So, for example, a gRPC server can be developed in Java with clients in Python.

By default, gRPC uses Protocol Buffers, an open-source mechanism for serializing structured data (although it can be used with other data formats such as JSON).

Synchronous RPC calls that block until a response arrives from the server are the closest approximation to the abstraction of a procedure call that RPC aspires to. On the other hand, networks are inherently asynchronous and in many scenarios, it is useful to be able to start RPCs without blocking the current thread. The gRPC programming API in most languages comes in both synchronous and asynchronous flavours.

Other protocols for configuration manipulation and state retrieval, like gNMI, are built on top of gRPC. Within B5G-OPEN, gRPC could be used to interface the several components control and data plane architecture such as for enabling topology synchronization in multi-controller environment or implement communication between the SDN controllers and the telemetry tools.

4.3.3 Kafka

Apache Kafka [KAFKA] is an open-source distributed *event streaming* platform used for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications. Kafka is a distributed system consisting of servers and clients that communicate via a high-performance TCP network protocol. Apache Kafka added benefit of data persistence, i.e., it combines three key capabilities:

- To publish (write) and subscribe to (read) streams of events, including continuous import/export of data from other systems.
- To store streams of events durably and reliably.
- To process streams of events as they occur or retrospectively.

This functionality is provided in a distributed, highly scalable, elastic, fault-tolerant, and secure manner. Kafka can be deployed on bare-metal hardware, virtual machines, and containers, and on-premises as well as in the cloud.

Kafka conforms to a publisher-subscriber architecture. Producers are those client applications that publish (write) events to Kafka, and consumers are those that subscribe to (read and process) these events. In Kafka, producers and consumers are fully decoupled and agnostic of each other, which is a key design element to achieve the high scalability that Kafka is known for.

Events are organized and durably stored in topics. Topics in Kafka are always multi-producer and multi-subscriber: a topic can have zero, one, or many producers that write events to it, as well as zero, one, or many consumers that subscribe to these events. Events in a topic can be read as often as needed, i.e., events are not deleted after consumption. Instead, you define for how long Kafka should retain your events through a per-topic configuration setting, after which old events will be discarded. Kafka's performance is effectively constant with respect to data size, so storing data for a long time is perfectly fine.

Topics are partitioned, meaning a topic is spread over a number of "buckets" located on different Kafka brokers. This distributed placement of your data is very important for scalability because it allows client applications to both read and write the data from/to many brokers at the same time. When a new event is published to a topic, it is appended to one of the topic's partitions. Events with the same event key (e.g., a customer or vehicle ID) are written to the same partition, and Kafka guarantees that any consumer of a given topic-partition will always read that partition's events in exactly the same order as they were written.

To make data fault-tolerant and highly available, every topic can be replicated, even across geo-regions or datacentres, so that there are always multiple brokers that have a copy of the data just in case things go wrong, owner wants to do maintenance on the brokers, and so on. A common production setting is a replication factor of 3, i.e., there will always be three copies of the data. This replication is performed at the level of topic-partitions.

4.3.4 Logstash and Elastic Search

Logstash [Logstash] is an open server-side data processing pipeline that ingests data from a multitude of sources, transforms it, and then sends it to any repository. The Logstash event processing pipeline has three stages: inputs → filters → outputs. Inputs generate events, filters modify them, and outputs ship them elsewhere. Inputs and outputs support codecs that enable to encode or decode the data as it enters or exits the pipeline without having to use a separate filter.

Elasticsearch [Elasticsearch] is a search engine that provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. It provides scalable search, has near real-time search, and supports multitenancy. Elasticsearch tries to make all its features available through the JSON and Java API. It supports faceting and percolating (a form of prospective search), which can be useful for notifying if new documents match for registered queries. Another feature, "gateway", handles the long-term persistence of the index; for example, an index can be recovered from the gateway in the event of a server crash. Elasticsearch supports real-time GET requests, which makes it suitable as a NoSQL datastore, but it lacks distributed transactions.

4.3.5 InfluxDB, Telegraf, and Grafana

InfluxDB [InfluxDB] is a high-performance time series engine designed to handle high write and query loads. InfluxDB is meant to be used as a backing store for any use case involving large amounts of timestamped data, including DevOps monitoring, application metrics, IoT sensor data, and real-time analytics. Some of the features that InfluxDB currently supports are:

- Custom high performance datastore written specifically for time series data.
- Written entirely in Go. It compiles into a single binary with no external dependencies.

- Simple, high performing write and query HTTP APIs.
- Expressive SQL-like query language tailored to easily query aggregated data.
- Tags allow series to be indexed for fast and efficient queries.
- Retention policies efficiently auto-expire stale data.
- Continuous queries automatically compute aggregate data to make frequent queries more efficient.

Telegraf [Telegraf] is an agent for collecting, processing, aggregating, and writing metrics. It is based on a plugin system to enable developers to easily add support for additional metric collection. There are four distinct types of plugins:

- Input Plugins collect metrics from the system, services, or 3rd party APIs
- Processor Plugins transform, decorate, and/or filter metrics
- Aggregator Plugins create aggregate metrics (e.g., mean, min, max, quantiles, etc.)
- Output Plugins write metrics to various destinations

Grafana [Grafana] is a multi-platform open-source analytics and interactive visualization web application that provides charts, graphs, and alerts for the web when connected to supported data sources. End users can create complex monitoring dashboards using interactive query builders. As a visualization tool, Grafana is a popular component in monitoring stacks, often used in combination with time series databases, monitoring platforms, and other data sources.

Grafana comes with a plethora of features that provide value straight out of the box:

- *Visualization*: Grafana possesses a huge variety of visualization options to help you view and understand your data easily. These options are split into "panels" which are then used to build the Grafana dashboard. A panel is the most granular visualization building block in Grafana, and is used to display data that has been queried from the data source attributed to that panel. This information is being pulled from the data source attributed to that panel and can be a type of graph (gauge, histogram, bar chart, etc.), or logs and alerts.
- *Alerting*: When monitoring applications, it is essential to be made aware the second something goes wrong, or is abnormal. This is vital to keeping your systems healthy and reducing downtime. Grafana has built-in support for a huge number of notification channels, e.g., email, REST endpoints etc.
- *Annotations:* Grafana allows leaving notes directly on graphs. This simple but powerful feature provides a way to seamlessly mark important points on graphs. This serves as a reminder for further action in the future, to provide context to an onboarding team member, or to simply mark a special event on your graph.

4.4 ORCHESTRATION

The Network Function Virtualization (NFV) concept enables the full automation of many processes that were previously manual, slow, and expensive. NFV is an indispensable component in 5G services, available in mobile and fixed access networks. In this context, the Network Function Virtualization Orchestrator (NFVO) is responsible for managing the Network Service (NS) life-cycle, orchestrating Network Function Virtualization Infrastructure (NFVI) resources, and optimizing the resource allocation and service connectivity. These resources are managed to offer high availability, the agility to swiftly deploy new services, and unlock the full potential of virtualization of network functions: scaling, faster deployment of services, simplified operations.

The NFVO system interacts with a number of other elements, Figure 4-1 shows a typical application scope. The main frameworks for NFVO are reviewed below: Open Source MANO (OSM), Open Network Automation Platform (ONAP), as well as other container-based alternatives.

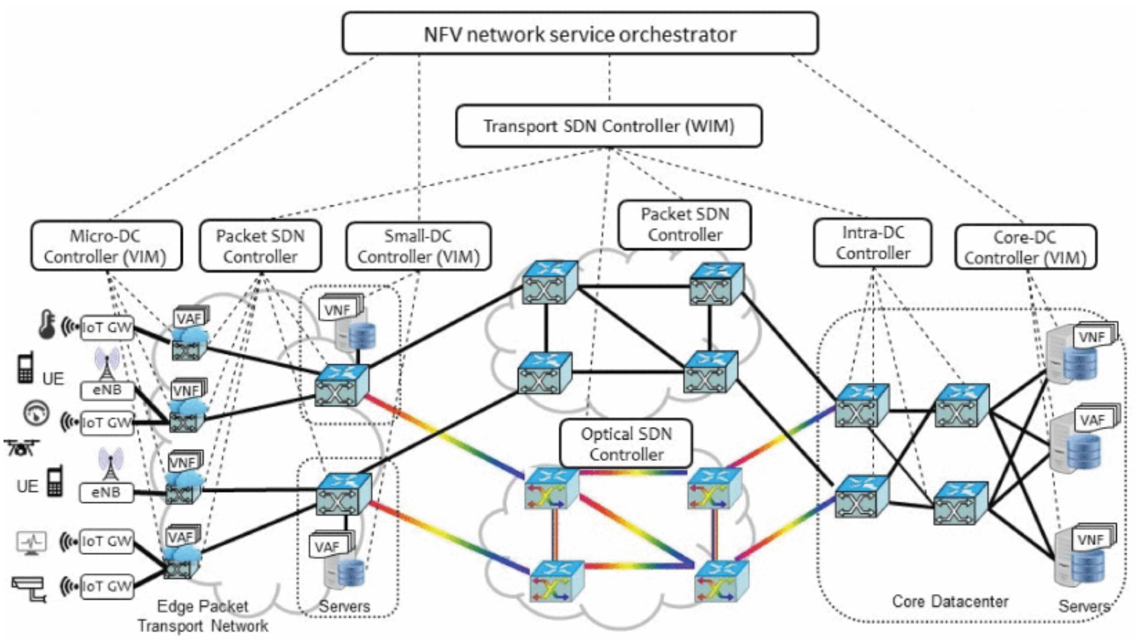


Figure 4-1 NFV Orchestration scope (source: [Mamu19])

4.4.1 OSM

Open-Source MANO (OSM) [OSM] is an open-source NFV manager and orchestrator of software stacks for NFV architecture developed by ETSI. OSM enables an ecosystem between operators and vendors to deliver and deploy services cost-effectively. The main architecture modules of OSM are (see Figure 4-2):

- User interface (UI) allows the OSM's service management as Launchpad module an interactive graphical interface. The main functions are:
  - Present NF and NS run-time statistics.
  - Detailed view of computing and network topologies.
  - Access credential management for VIM environments.
- Service Orchestrator (SO) manages the workflow via OSM. The main objectives are:
  - Life-cycle management and primitive execution of the service.
  - Project and user support.
  - NS and VNF life-cycle operation (CRUD)
- Resource Orchestrator (RO) is responsible for resource management and coordination from multiple VIMs or SDN controllers.
- VNF Configuration and Abstraction (VCA) sets the initial VNF design and enables the actions, configurations, and notifications from iterations between VNFs.
- Network Service to VNF Communication (N2VC) is a key module managing the communication between the SO and VCA modules.

- Kafka bus enables a new channel for asynchronous communication between all the components and eases the OSM integration with new modules (see Figure 4-3).
- Virtual Infrastructure Manager (VIM) is a fundamental element in MANO architecture, but it is not present in the OSM stack. Instead, OSM is open to integrate with main ones like OpenVIM, VMWare vCloud Director, Amazon Web Services, and OpenStack.
- Monitoring (MON) integrates the monitoring tool according to the project requirements and OSM architecture.
- Northbound Interface (NBI)
- Policy Manager (PoM) to add notifications where a new metric or alarm is relevant to the MON module.

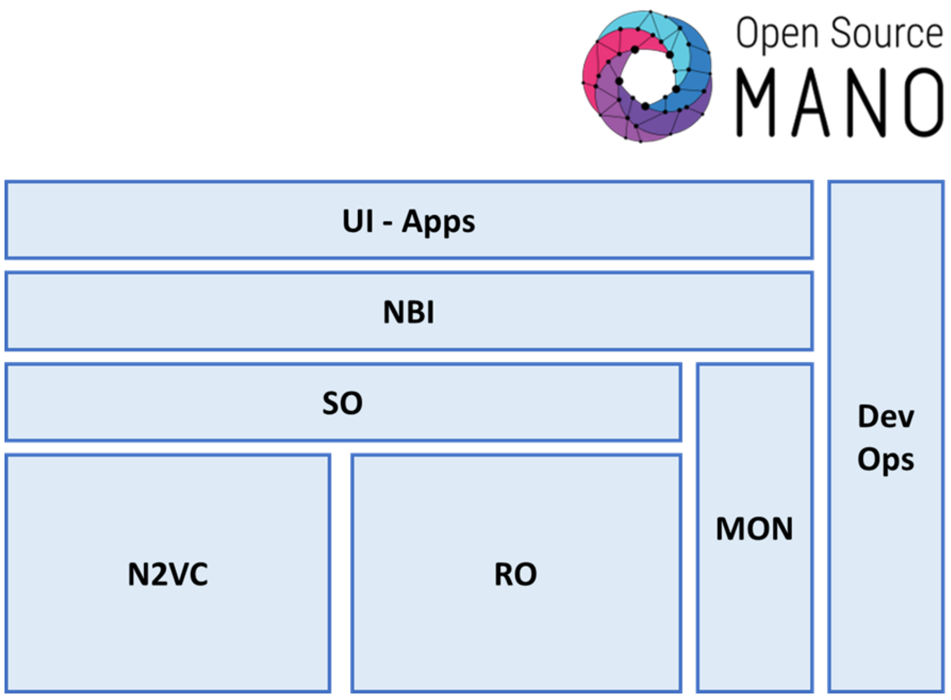


Figure 4-2 OSM Modules (source: [OSMRel4])

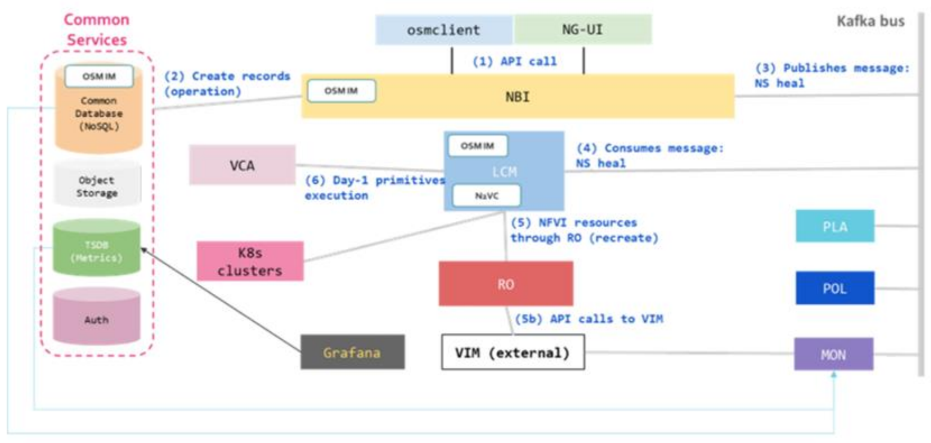


Figure 4-3 OSM life-cycle management for a new dedicated channel for asynchronous communications between components (source: [OSMRel4])

OSM has been successfully used in a number of European projects like MetroHaul [MHEC], 5G-CrossHaul [5GCH] to name a few, and has a fair support by the community. As drawbacks, OSM complexity and steep learning curve makes it less attractive. Additionally, its support of lightweight containers, a major driver in 5G/6G infrastructures, is not as complete as other alternatives.

4.4.2 ONAP

Open Network Automation Platform (ONAP) [ONAP] is an open-source software platform developed under the Linux Foundation that enables the design, creation, and orchestration of services on the infrastructure layer sitting on top of individual VNF or SDN, or a combination of both. The main goal is to address efficient provisioning and end-to-end infrastructure management serving up on-demand services and optimizing the automation of the associated processes using big data and Artificial Intelligence (AI) [Mam19]. ONAP environment consists of two major architectural frameworks: design-time environment and execution-time environment (see Figure 4-4) [Sli17].

- The design-time environment is a development environment with functions and libraries needed for the development of new capabilities. It entails a visual tool for the design and modelling of assets used in ONAP components with a subsystem to make policies and conditional rules. The design-time environment has the following subcomponents:
    - Service Design and Creation (SDC) is an environment that describes based on multiple levels of assets how VNFs or services are managed, including resources and services requirements.
    - Policy Creation is an ONAP subsystem that contains a set of control rules, orchestration and management policies. The VNF placement rule specifies where VNFs should be placed according to the constraints.
- The execution-time environment is an environment to execute the policies and rules prepared in the design-time environment. The policies and rules are responsible for data collection, analytics, and resource inventory. This environment includes service orchestration for end-to-end service automation, performance monitoring, and security based on Enhanced Control, Orchestration, Management & Policy (EOCMP). The main modules are:
    - Active and Available Inventory (AAI): This is continually updated to provide a real-time view of the topology underlying the available resources.
    - Controllers: One controller manages the state of a single resource. ONAP uses multiple controllers to execute resource configuration and instantiation to configure the network and manage VNFs. On the other hand, the Application Controller manages more complicated VNFs and services. Finally, the Infrastructure controller orchestrates and manages the resources within the local or cloud infrastructure.
    - Master Service Orchestrator (MSO): It handles capabilities of end-to-end service provisioning from the top level.
    - Data Collection, Analysis, and Events (DCAE): The main role is the telemetry data collection from VNFs to detect network anomalies and determine the corrective actions.

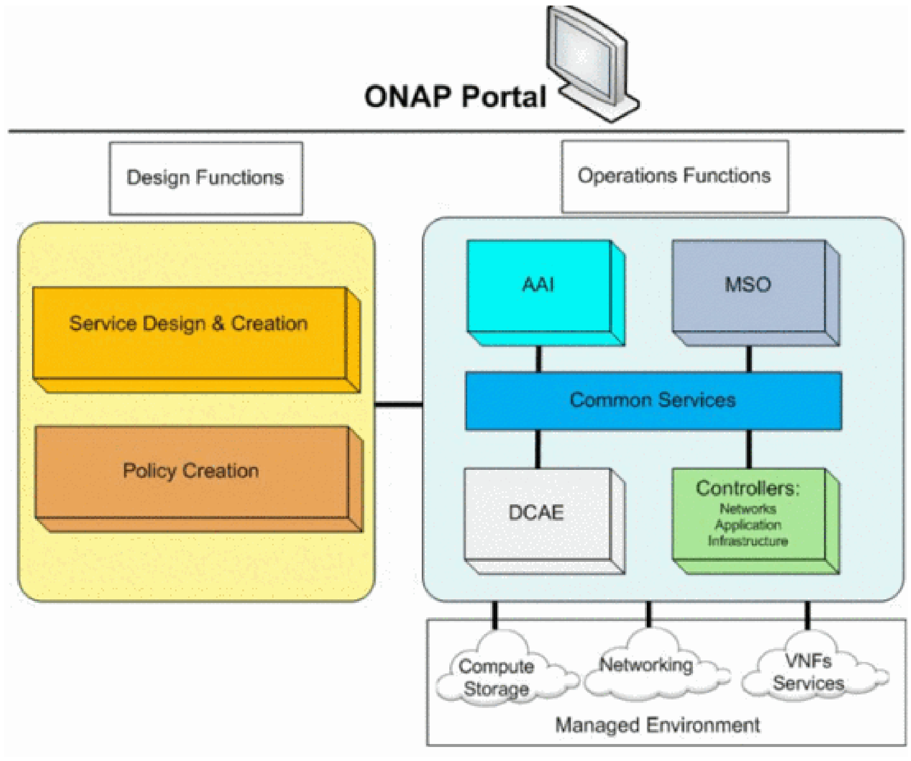


Figure 4-4 ONAP system (source: [Slim17])

Similarly to OSM, and even more, ONAP software has become a very complex software suite, difficult to manage and deploy. It has a growing community, and an increasingly better documentation quality. Still, the learning curve is also steep, which is hindering its adoption.

4.4.3 Alternatives based on containers

In recent years, a growing interest exists in the industry for the exploration of simpler NFVO strategies, based on smaller and well-known frameworks with well-defined APIs and interactions, instead of large monolithic software suites like ONAP or OSM. In this line, the Container Orchestration Engines (COE) [ETSI19] are specialized tools, which are capturing the industry attention as a simpler form to address the NFV and on-demand application deployment and scaling needs.

COEs automate the containers' deployment, management, scaling, and networking over a cluster of computers by an API for life-cycle management, schedule the containers based on available resources and needs. Its key role: automating in a simple form the painful management of the container life-cycle when its number increases dynamically with demand.

The most popular and widely adopted option is Kubernetes [Red22]. Below key aspects of Kubernetes will be reviewed, as well as Open Baton, an (also open-source) alternative, connected to the ETSI NFV MANO umbrella.

4.4.3.1 *Open Baton*

Open Baton [OpenBaton] is an open-source platform that presents a comprehensive development of the ETSI NFV MANO specification for virtual network infrastructures (see Figure 4-5). This tool ports, adapts, manages, and orchestrates the network functions in a specific network environment to improve the performance and grant security of the overall infrastructure. This solution can provide on-demand complete virtualized infrastructures such as IaaS or PaaS enabling the elastic deployment of cost-efficient network infrastructures. The main components are:

- A generic VNFM and generic EMS to manage the VNF life-cycle based on their descriptors.
- Many VNFM drivers to select.
- Multi-VIM without having to re-write the orchestration logic.
- Event mechanism based on pub/sub mechanism for dispatching the execution of the life-cycle events.
- Autoscaling engine for automatic runtime management of scaling operations of the VNF.
- Fault management system at any level.
- Monitoring system based on Zabbix
- Network slicing to ensure a specific QoS for the NS.
- Set of libraries (Java, Go, and Python) for building your VNFM

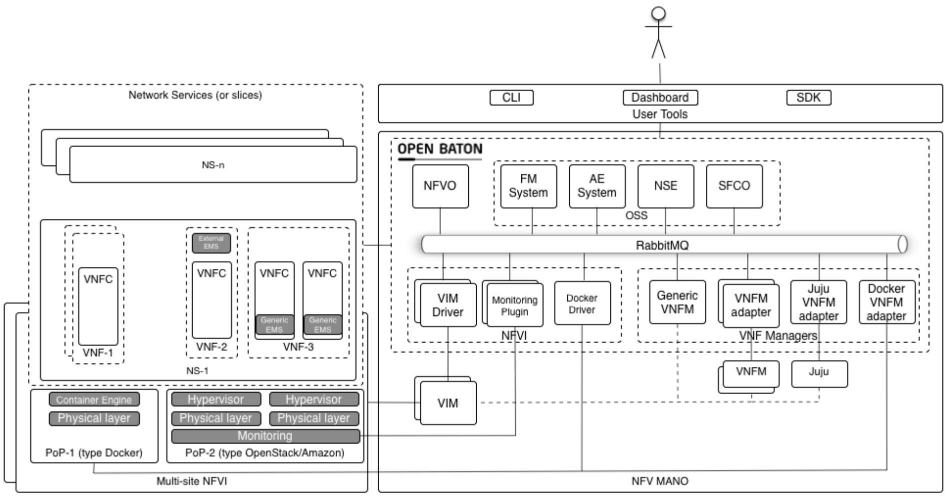


Figure 4-5 Open Baton Components (source: [OpenBaton])

Open Baton has been chosen in a number of research projects as NUBOMEDIA [Nubomedia] Mobile Cloud Networking [MCN] and SoftFIRE [SoftFIRE], exploring its application in different 5G/6G contexts. The official Open Baton online documentation [OpenBaton] includes the User Tool section with information related to Dashboard, CLI, SDKs and REST APIs. Nowadays, Open Baton is an attractive platform but is not widely adopted by the industrial or researching community in contrast to Kubernetes.

4.4.3.2 Kubernetes

Kubernetes (K8s) [K8s] is an open-source platform for container orchestration developed by Google in 2014 with virtualization orchestration in mind. The main functions are automated arrangement, coordination, and cluster management of containerized applications. Kubernetes becomes extremely useful and powerful as a solution for most challenging functionalities (service discovery, load balancing, health checks, auto-scaling containers, nodes, etc.).

Kubernetes is portable, configurable, modular, and offer features like container auto-placement, auto-restart, auto replication, and auto-healing. The main element of Kubernetes are [Chifor17]:

- Master node assures everything is running with multiple controllers in charge of the cluster health, replication, scheduling, endpoints (Services and Pods), Kubernetes API, etc. This node is present in a separate virtual machine in the same physical host.
- Worker node runs the Kubernetes agent and is responsible for running Pod containers, mounts Pod volumes, does health checks, and reports the Pods and the node status to the rest of the system.
- Pod is the smallest and simplest unit in the Kubernetes model, represents a running process in the clusters, and contains one or more containers.
- Deployment provides declarative updates for Pods and ReplicaSets.
- DaemonSet ensures that the nodes run a copy of a Pod (nodes added/removed to the clusters, Pods added/garbaged to them).
- ReplicaSet is a controller that ensures a specified number of Pods replicas running at any given time.
- Service is an abstraction that defines a logical set of Pods and a policy to access them. It exposes the Pods to other services within the cluster or externally.

For B5G-OPEN, the benefits of Kubernetes are:

- Its specialized focus on the use of lightweight containers, the ideal mechanism to host the microservice-based applications, a clear industry trend, where each application can be composed of tens of containers.
- In contrast to all other platforms, the Kubernetes concept is simple with sophisticated code and documentation. It exports a REST-based API, to efficiently handle the resource lifecycle.
- Kubernetes is also an enabler for the so-called serverless approach [Moh19]. The idea under such strategy includes a diverse set of techniques, which pursue simplifying even further the development and lifecycle management of micro-service-based applications. For instance, Amazon AWS offer as a serverless function, the so-called lambda-service. This permits the user the definition of *Zero-touch functions* (lambdas) that are e.g., independent AWS containers without management tasks. In this form, more and more of the management burden would be actually handled by the cloud provider. In conclusion, Kubernetes would open the door to exploring the advantages of serverless approaches in the network operation architecture.

4.5 QoT ESTIMATION TOOLS

As a fundamental initial hypothesis of B5G-OPEN, optical transmission exploiting multiple optical bands is the most promising solution in the context of wavelength routed networks that increases network capacity without compromising network node connectivity and without exhausting the operator’s deployed fibre reserves. A disaggregated, vendor-agnostic, optical multi-band transport ecosystem is necessary to ensure the cost-effective deployment of these systems, which are facing considerable challenges in physical network design and network planning: in an OMB system, physical layer phenomena impose additional and complex performance limitations. Additionally, highly desirable features are network automation and interoperability by means of open interfaces and an SDN-enabled control/management framework. Such interfaces and frameworks are not currently designed with multi-band support, which poses additional challenges. It is necessary to develop modular SDN architecture where a physical layer impairment aware (PLA) routing engine and QoT estimation tools can be used for use cases such as path validation and path computation, ideally relying on open interfaces and standard data models extended for such purpose.

4.5.1 GNPy

GNPy is an open-source, community-developed library for building route planning and optimization tools in real-world mesh optical networks [GNPY]. The project is driven by a consortium of operators, vendors, and academic researchers sponsored via the Telecom Infra Project's OOPT/PSE working group, building as tool for rapid development of production-grade route planning tools which is easily extensible to include custom network elements and performant to the scale of real-world mesh optical networks.

GNPy can act as a Path Computation Engine, tracking bandwidth requests, or advising the SDN controller about a best possible path through a large DWDM network. For example, it takes into account detailed models of data plane devices and optical fibres to provide an accurate estimation of the quality of transmission reachable along a specified path.

4.5.2 OLC-E Tool

The OLC-E tool is used either as a stand-alone planning tool or as the main element of a Path Computational Engine (PCE). The OLC-E tool, as detailed in [UZU21], it 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. This task is completed in the 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 continuous 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.

With the aid of the notations and the parameters listed in Table 4.1, the implementation of the multi-band routing engine is as below:

Table 4.1: Variables and parameters used in the multi-band routing engine.

Variable	*Description*	*Variable*	*Description*
G	network topology graph	d_s	source of demand *d*
N	set of network nodes	d_n	destination of demand *d*
E	set of bidirectional optical fibre links (edges)	d_t	duration of demand *d*
A	Set of amplifiers in the network	d_{lr}	Requested line-rate for demand *d*
B	set of active optical bands	k	number of shortest paths used in Yen's algorithm
C_B	set of available frequency slot units - FSUs for each optical band in the set *B*	K	set of k-shortest paths calculated using Yen's algorithm
T	Input traffic matrix	p_c	candidate path assigned to demand *d*
D	set of demands in increasing time of arrival order	r_c	candidate transmitter type assigned to demand *d*
D_t	average time duration of the demands	f_c	candidate set of FSUs assigned to demand *d*
D_i	average inter-arrival time between two consecutive demands	p_a	final path assigned to demand *d*
LR	set of available line-rates	r_a	final transmitter type assigned to demand *d*
R_B	set of available transmitter types for each optical band in *B*, in increasing required FSU order	f_a	final set of FSUs assigned to demand *d*
r	transmitter type (macroscopic parameters)	w_a	final transmitter power for demand *d*
r_d	Maximum reach of a transmitter type r	S_t	total simulation time
r_f	number of consecutive FSUs consumed per transmitter type r	t	current simulation time
d	One particular traffic demand (source-destination)	d_{rt}	boolean, TRUE if demand *d* is routed, FALSE otherwise

INPUT: Network topology including nodes, edges and amplifiers *G(N, E, A)*. Define the optical bands *B engaged.* Definition of the capacity C_B per band. Definition of traffic matrix *T*. Average time duration D_t of the demands and average inter-arrival time D_i between two consecutive demands. Definition of the available line-rates *LR* and their distribution on the demands.

Stage 1: Network Topology Implementation:

```
1: for all $n_i \in N$ do
2:   for all $n_j \in N$ do
3:     Compute k shortest-paths using the Yen's algorithm and store the results to $K(n_i,n_j)$
```

```
4:   end for
5:  end for
6:  t=0
7:  while t< $S_t$
8:    Generate a new demand $d$; Add $d$ to $D$;
9:  end while
Stage 2A: Spectral and Modulation Assignment (SMA) and PL entanglement:
10:  while $D$ is not empty do
11:    Consider the first demand $d \in D$;
12:    $B_d$=$B$; $K_d$=k-shortest paths from $d_s$ to $d_n$ in $K(n_i,n_j)$; $d_{rt}$=false;
13:    while $B_d$ is not empty AND $d_{rt}$=false do
14:      Consider the first band $b \in B_d$;
15:      while $K_d$ is not empty AND $d_{rt}$=false do
16:        Consider the first path $p_c \in K_d$;
17:        Calculate the set of $r \in R_B$ denoted as $R_d$ assuming band $b$ and line-rate $d_{lr}$;
18:        for all $r \in R_d$ do
19:          if ($r_d < p_c$ distance)
20:            Remove $r$ from $R_d$;
21:          end if
22:        end for
23:        while $R_d$ is not empty AND $d_{rt}$=false do
24:          Consider the first $r_c \in R_d$ assuming band $b$ and line-rate $d_{lt}$;
25 :         while $FSU_i < C_b$ AND $d_{rt}$=false do
26:          Calculate the next available set of FSUs $FSU_n$ in the path $p_c$ starting from $FSU_i$ using the First Fit (FF) route algorithm
27:            if (route found in FF)
Stage 2B: Physical Layer Performance:
28:              Execute Physical Layer Check (PLC) using Path OSNIR
29:              if PLC == true
30:                Assign the demand using Path Allocation $(d,b,pc,d_{bw})$;
31:              end if
32:            end if
33:          end while; Remove $r_c$ from $R_d$;
34:        end while; Remove $p_c$ from $K_d$;
35:      end while; Remove $b$ from $B_d$;
36:    end while
37:    if ($d_{rt}$=false)
38:      Block demand $d$;
39:    end if
40:  end while
Stage 3: Path Allocation:
41:  Allocate path $p_a$= $p_c$ in the network;
42:  Allocate set of frequency slot units $f_a$=$f_c$ in the optical band $b$ across the path $p_a$ in the network;
43:  Allocate transmitter $r_a$=$r_c$ across the path $p_a$ in the network;
44:  Set $w_a$ as the power of transmitter $r_a$;
OUTPUT: Utilisation of frequency slot units, finalisation of modulation format for a given line-rate, the consumed optical
band and the power channel for all demands in the set $D$.
```

The OLC-E tool (v1.0) has the following features:

- The physical layer performance is estimated via closed-form expressions something that allows to get the results in real-time. In particular, the OLC-E tool admits, assesses, and routes thousands of call set-up requests within the timeframe of few minutes. Moreover, the OLC-E tool (v1.0) has integrated an advanced power optimization methodology that allows to tailor the physical layer performance of each optical band according to the high-level objectives set by the network operator as, for example, whether the operators wish all optical bands to have the same optical reach or whether the optical reach of a band (or bands) is higher at the expense of the optical reach of other bands.. Examples of such optimizations are detailed in D3.1.
- It supports both transparent and translucent modes of operation. In the latter mode, a path that is rejected during STAGE-II, it is re-iterated via the “Path-Split Routine” which allows to split a rejected transparent optical path in two shorter-length transparent paths with full o/e regeneration at an intermediate node. After this

implementation, the two shorter length paths may exploit a higher cardinality SMA and/or a lower symbol-rate source, which are, possibly, independently selected in the two paths. This way, the total number of optical slots consumed in the two independent paths are reduced alleviating blocking due to spectral unavailability. Similarly, higher line-rates might be employed in the two shorter-length paths that reach the QoT threshold that otherwise might be unattainable.

The flowchart illustrated in Figure 4-6 summarizes the logical implementation of the OLC-E tool.

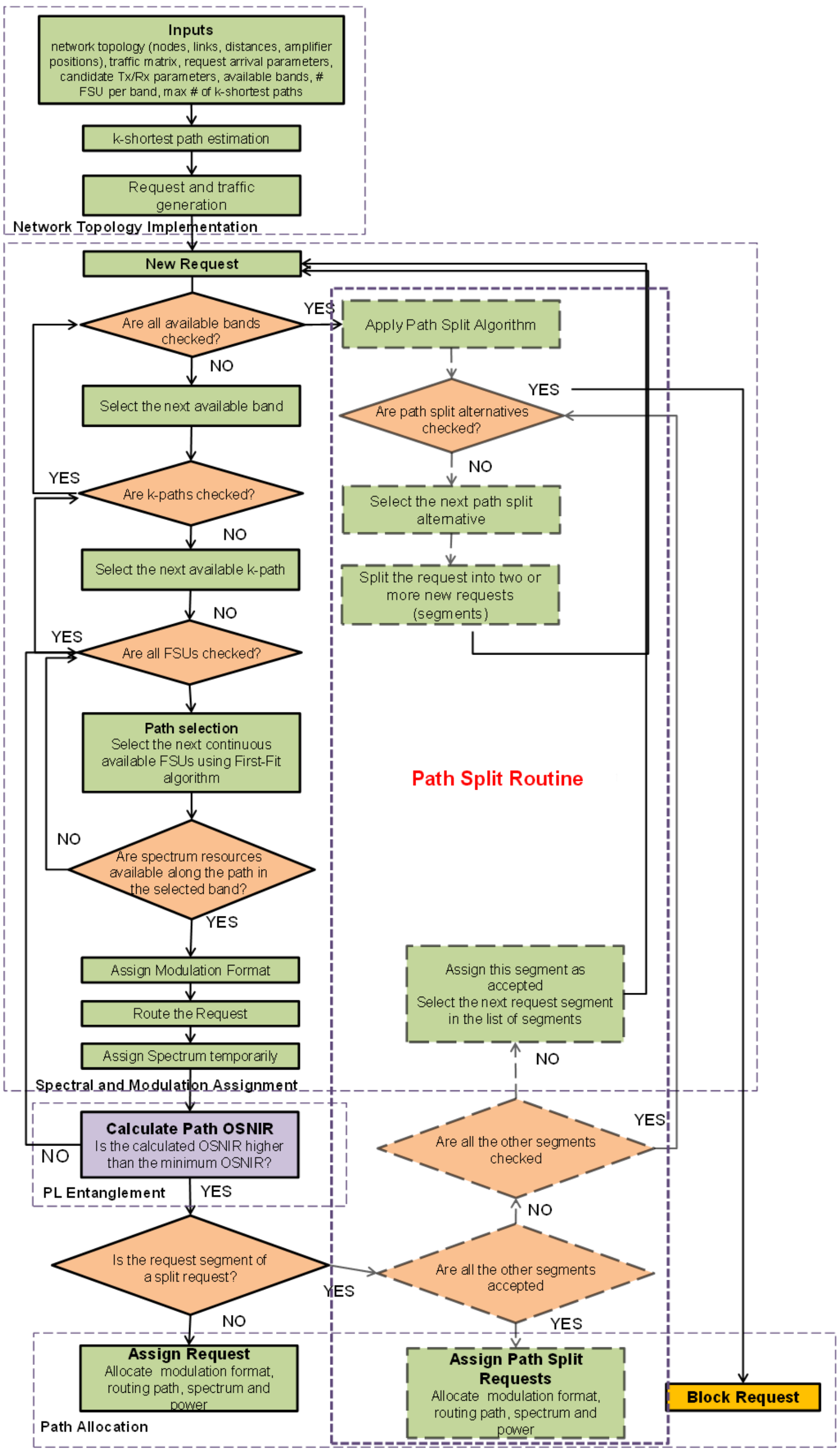

Inputs
network topology (nodes, links, distances, amplifier positions), traffic matrix, request arrival parameters, candidate Tx/Rx parameters, available bands, # FSU per band, max # of k-shortest paths
k-shortest path estimation
Request and traffic generation
Network Topology Implementation
New Request
Are all available bands checked?
YES
NO
Select the next available band
Are k-paths checked?
YES
NO
Select the next available k-path
NO
Are all FSUs checked?
YES
Path selection
Select the next continuous available FSUs using First-Fit algorithm
NO
Are spectrum resources available along the path in the selected band?
YES
Assign Modulation Format
Route the Request
Assign Spectrum temporarily
Spectral and Modulation Assignment
Calculate Path OSNIR
Is the calculated OSNIR higher than the minimum OSNIR?
NO
YES
PL Entanglement
Is the request segment of a split request?
YES
NO
Assign Request
Allocate modulation format, routing path, spectrum and power
Path Allocation
Apply Path Split Algorithm
Are path split alternatives checked?
YES
NO
Select the next path split alternative
Split the request into two or more new requests (segments)
Path Split Routine
Assign this segment as accepted
Select the next request segment in the list of segments
NO
Are all the other segments checked
YES
NO
Are all the other segments accepted
YES
Assign Path Split Requests
Allocate modulation format, routing path, spectrum and power
Block Request

Figure 4-6 The flow chart of the OLC-E's multi-band routing engine.

5 Overview of B5G-OPEN Control, Orchestration, and Telemetry

The B5G-OPEN control, orchestration, and telemetry system (often referred to as the control plane, for short) is the software systems that provides the ability to provision, dynamically and upon demand, B5G and 6G services, as presented in the previous sections.

The following sections macroscopically present the functional architecture of the B5G-OPEN control plane, initially targeting single domain networks, which was initially proposed in Milestone M4.1 and will be refined along the project execution. This section presents, macroscopically, the most relevant elements of the architecture. The different functional elements (often referred to as components) are identified for the purposes of service orchestration and device configuration (incl. resource control).

5.1 Main Innovations at the B5G-OPEN control plane

The main innovations for the control plane of B5G-OPEN are:

- [**multiband control**] Control of optical multi-band network, this means being able to exploit the multiband capabilities of optical devices such as transmission (Tx) elements – transceivers) or switching (multi-band ROADMs).

  This is detailed in Section 6, showing the B5G-OPEN approach.

- [**transparent multi-domain**, *domain-less*] The ability to setup 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.

  This is covered in Section 6, as well as considerations regarding Access Segment integration elaborated in Section 7 as well as considerations regarding integration with IT (computing, storage) and orchestration in Section 8.

- [**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 limited to transceiver to transceiver 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.

This, along with control of multiband networks, is a critical innovation of B5G-OPEN. A dedicated section is provided (Section 9)

- [**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 interop 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 or Net2Plan 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 10.

- [**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).

Such aspects are elaborated on in Section 11.

5.2 Initial Assumptions on Optical Device Configuration and Control

The definition of the architecture relies on a set of initial assumptions, namely that the devices are client agnostic (they export several configuration endpoints, based on separation of concerns, functionality, or administrative assignment), export several telemetry endpoints, with same considerations since configuration and Telemetry endpoints may have different access requirements, visibility, and interfaces should be homogeneous. For devices that export multiple configuration endpoints, it is expected that the scope of each endpoint is clearly defined, and/or side effects are well-known (i.e., no overlapping models).

The architecture (reflected in Figure 5-1) is defined targeting two main models: i) partial disaggregation with a 2-level control hierarchy, where there is a dedicated OLS controller, responsible for the ROADM and ILA nodes (note that ROADM/ILA nodes MAY export other interfaces (e.g., streaming telemetry) towards other entities, and ii) Full Disaggregation, with a single SDN controller. Both models may include additional functional elements, notably in support of path computation, resource allocation, or function placement.

As addressed in the previous section, the control plane architecture assumes several key services, such as the provisioning of DSR or Media Channel connectivity services and contemplates two main blocks: Multi-band Optical Network SDN control and Domain Telemetry Collector. The MB Optical Network Control is fully decomposed on TAPI adapter, Path Computation Servers and Optical Controllers (e.g., in the case of partial disaggregation additional OLS controllers will be considered). For the packet domain, different options are addressed. Packet controllers can cover one or multiple packet domains and rely on pure SDN (e.g., P4) or hybrid SDN/IP in which the SDN control plane is mostly used to configure IP processes running in the packet/optical boxes. In the case of multi-OLS scenarios, B5G-OPEN will consider B2B deployments with Transparent Configuration.

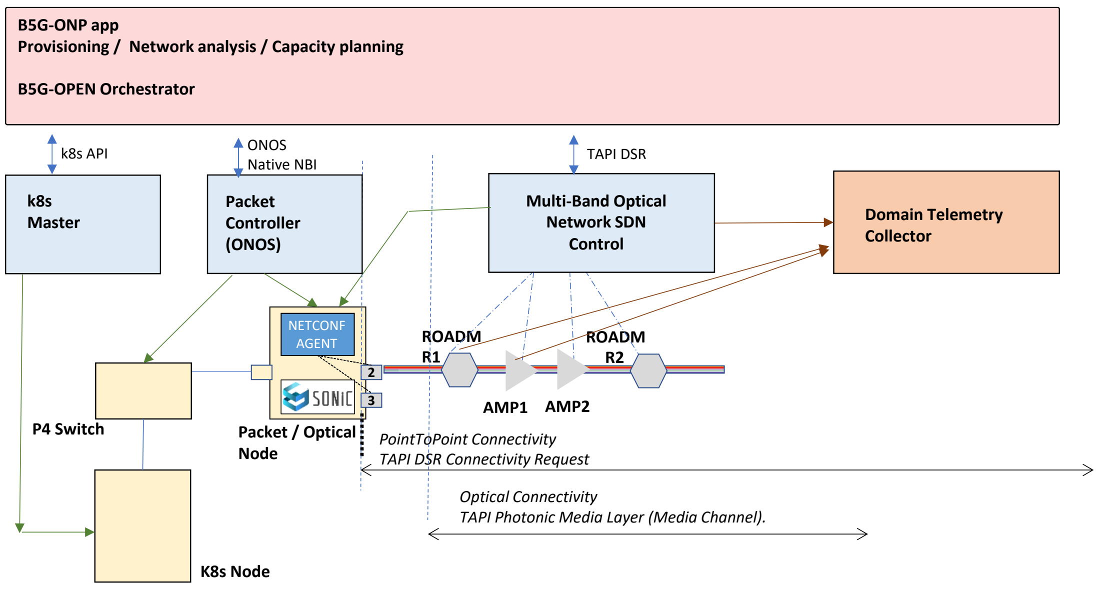

B5G-ONP app
Provisioning / Network analysis / Capacity planning
B5G-OPEN Orchestrator
k8s API
ONOS Native NBI
TAPI DSR
k8s Master
Packet Controller (ONOS)
Multi-Band Optical Network SDN Control
Domain Telemetry Collector
NETCONF AGENT
SONIC
ROADM R1
ROADM R2
AMP1
AMP2
2
3
P4 Switch
Packet / Optical Node
PointToPoint Connectivity
TAPI DSR Connectivity Request
Optical Connectivity
TAPI Photonic Media Layer (Media Channel).
K8s Node

Figure 5-1 B5G-OPEN Control, Orchestration and Telemetry architecture.

5.3 Service Orchestration and Planning

Aspects related to Service orchestration for provisioning, planning, or network analysis are responsibility of the Service Orchestrator (referred to as the B5G-ONP). Such element sits on top of the Kubernetes controller, the SDN controllers and the Domain Telemetry collector.

5.4 Optical Packet Integration

This section overviews the control plane assumptions related to Optical and Packet integration, on top of which the different control plane architectural solutions (e.g., exclusive, concurrent) can be defined. B5G-OPEN will focus on the concurrent solution, in which different controllers can have access to the packet / optical nodes. This is driven by criteria related to implementation simplicity, but it does not mean that the other solutions are not appropriate.

In either case, the data plane assumes pluggable interfaces in the packet / optical nodes, there is no discrete optical element.

5.4.1 Campus Mode

In the Campus mode (Figure 5-), packet forwarding is based on P4 runtime. Consequently, the SDN controller for the packet layer is responsible for configuring flow forwarding as defined by the P4 standard documents.

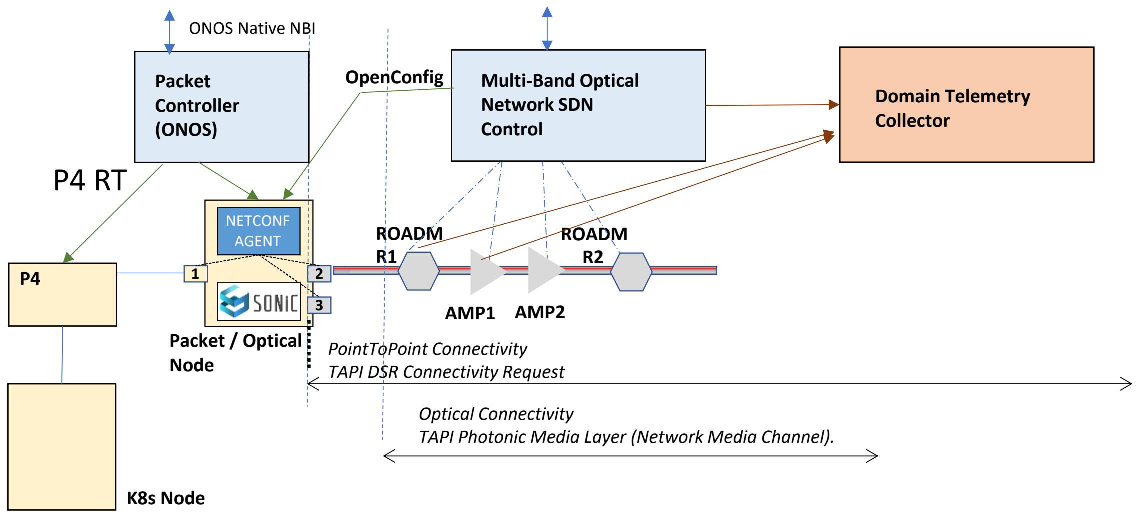

ONOS Native NBI
Packet Controller (ONOS)
OpenConfig
Multi-Band Optical Network SDN Control
Domain Telemetry Collector
P4 RT
NETCONF AGENT
SONIC
P4
1
2
3
ROADM R1
ROADM R2
AMP1
AMP2
Packet / Optical Node
PointToPoint Connectivity
TAPI DSR Connectivity Request
Optical Connectivity
TAPI Photonic Media Layer (Network Media Channel).
K8s Node

Figure 5-2 Packet/Optical integration (campus / p4 modes)

5.4.2 Telco Mode

In the telco mode, it is assumed that packet/optical nodes are actually IP routers and that there are one or more routing processes (e.g., BGP / OSPF) running on the node. SDN applicability in this mode mainly refers to the fact that actual configuration of the routers is driven by the SDN controller (see Figure 5-).

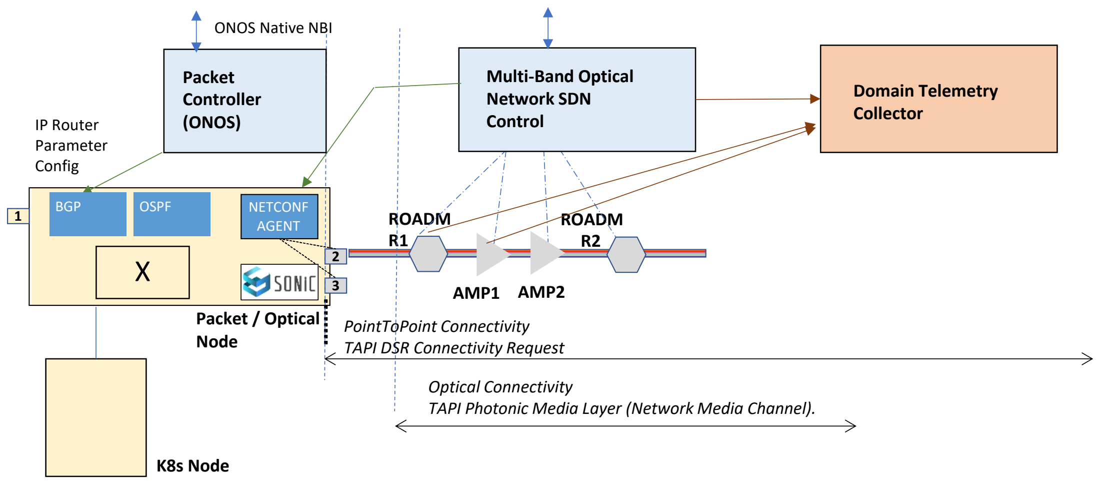

ONOS Native NBI
Packet Controller (ONOS)
Multi-Band Optical Network SDN Control
Domain Telemetry Collector
IP Router Parameter Config
1
BGP
OSPF
NETCONF AGENT
X
SONIC
2
3
ROADM R1
ROADM R2
AMP1
AMP2
Packet / Optical Node
PointToPoint Connectivity
TAPI DSR Connectivity Request
Optical Connectivity
TAPI Photonic Media Layer (Network Media Channel).
K8s Node

Figure 5-3 Packet/Optical integration (telco / router modes)

5.5 TELEMETRY AND INTENT BASED NETWORKING

The domain telemetry collector architecture has also been defined (see Figure 5-). 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.

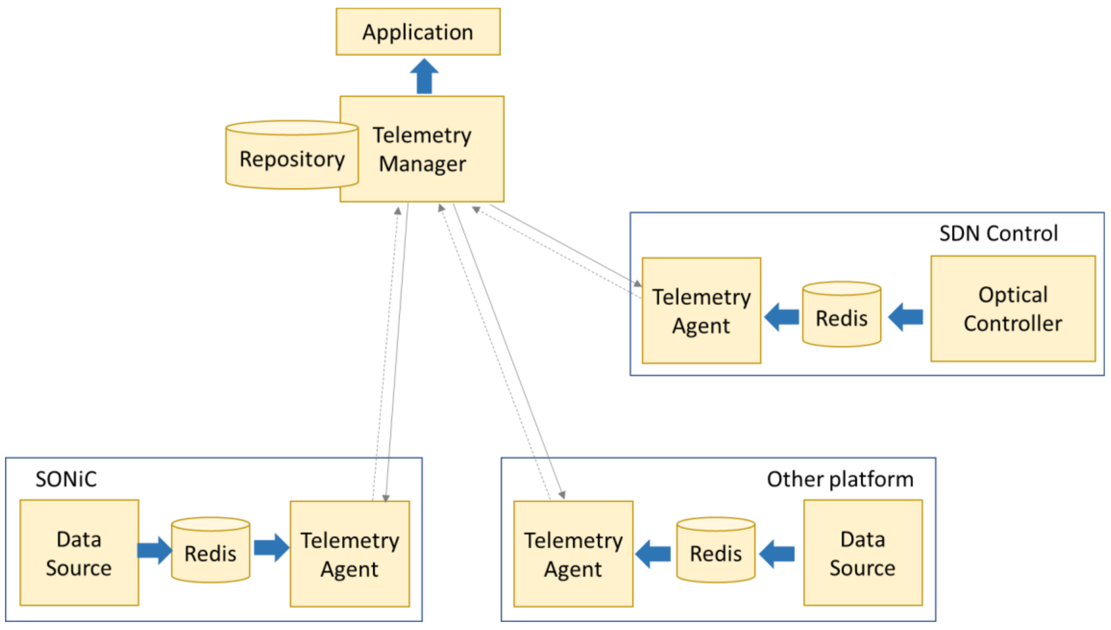


Figure 5-4 B5G-OPEN Control and Orchestration architecture

Finally, the B5G-OPEN architecture operates service and network operations 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 5-5.

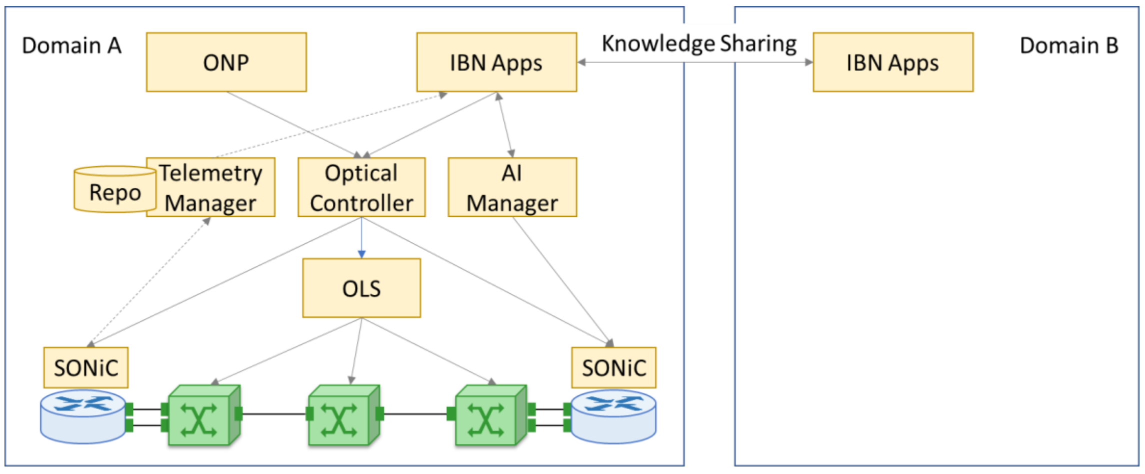


Figure 5-5 B5G-OPEN Intent Based Applications (IBN) and Knowledge-Sharing.

6 SDN CONTROL OF OPTICAL MULTIBAND NETWORKS

6.1 INTRODUCTION

The “optical control” part of B5G-OPEN refers to the control of the transceivers devices and the optical line system. B5G-OPEN considers two different approaches within a disaggregated system: a fully disaggregated (with visibility of the underlying devices) and partially disaggregated. In this case, the SDN control plane for partially disaggregated networks follows a hierarchical arrangement of controllers, in which a first level control-plane disaggregates the transceivers from the OLS. A second level, the OLS controller is responsible for provisioning (Network) Media Channels (MCs/NMCs) between client ports and for configuring ROADM devices and any other applicable device in the optical path.

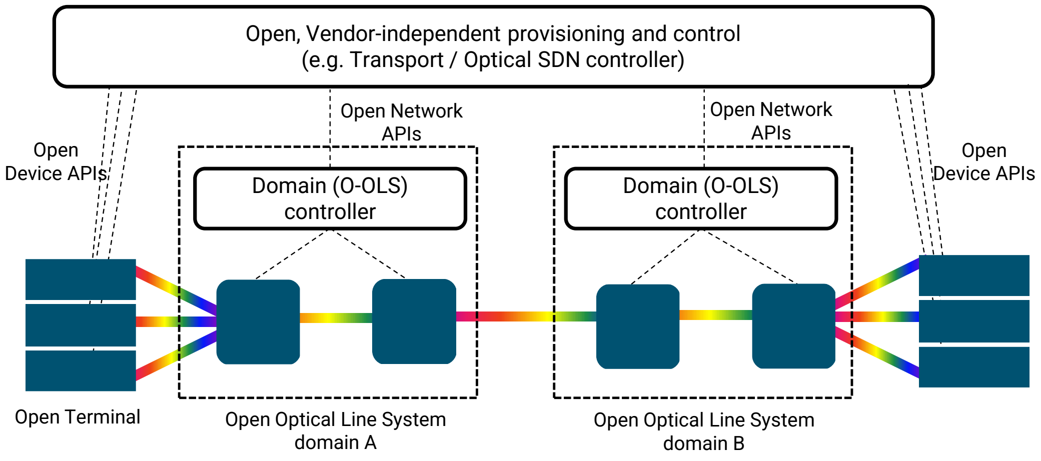

Open, Vendor-independent provisioning and control
(e.g. Transport / Optical SDN controller)
Open Device APIs
Open Network APIs
Open Network APIs
Open Device APIs
Domain (O-OLS) controller
Domain (O-OLS) controller
Open Terminal
Open Optical Line System domain A
Open Optical Line System domain B

Figure 6-1 Control Plane of Partially Disaggregated Optical Networks with OLS controller

In specific scenarios, SDN agents are deployed at each node, which, in turn, acts as local node controllers to configure the different devices (devices are in most cases exposing an SDN device API). When considering “open interfaces” the interfaces towards the individual devices under the control of an OLS controller are commonly not exported and the visibility for higher (e.g., client) applications is limited. In a partially disaggregated system, there are not necessarily SDN agents in the ROADM nodes, as they are configured by the OLS controller using proprietary (non-SDN) interfaces. There may additionally be device SDN API on the ROADM devices for per-device configuration and PM monitoring / streaming telemetry.

6.2 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.

6.2.1 Interfaces

By design, the TAPI Optical Network Orchestrator is an SDN Controller that exports a standard NBI, (based on the standard ONF TAPI interfaces) while orchestrating and coordinating multiple delegated systems, such as the Optical SDN controller, as well as multiple sources of physical impairment information.

- The interface from the TAPI Optical Network Orchestrator to the optical controller will be based on the ONOS native interface, extending the existing implementation to support additional requirements and use cases

It is also responsible for performing path computation to the Optical Path Computation Element (OPCE), running in a dedicated Path Computation Server or as part of a planning software, while also using an open and standard interface for such purpose.

- The interface from the TAPI Optical Network Orchestrator to path Computation engine will be based on a specific instance of path computation interface defined in TAPI.
- Additional interfaces will be defined to support the augmentation of topological elements with physical layer information data.

From an architectural perspective, the TAPI Optical Network Orchestrator (see Figure 6-2) abstracts the optical controller and path computation entities from the upper layers (notably, B5G-OPEN planner and orchestrator).

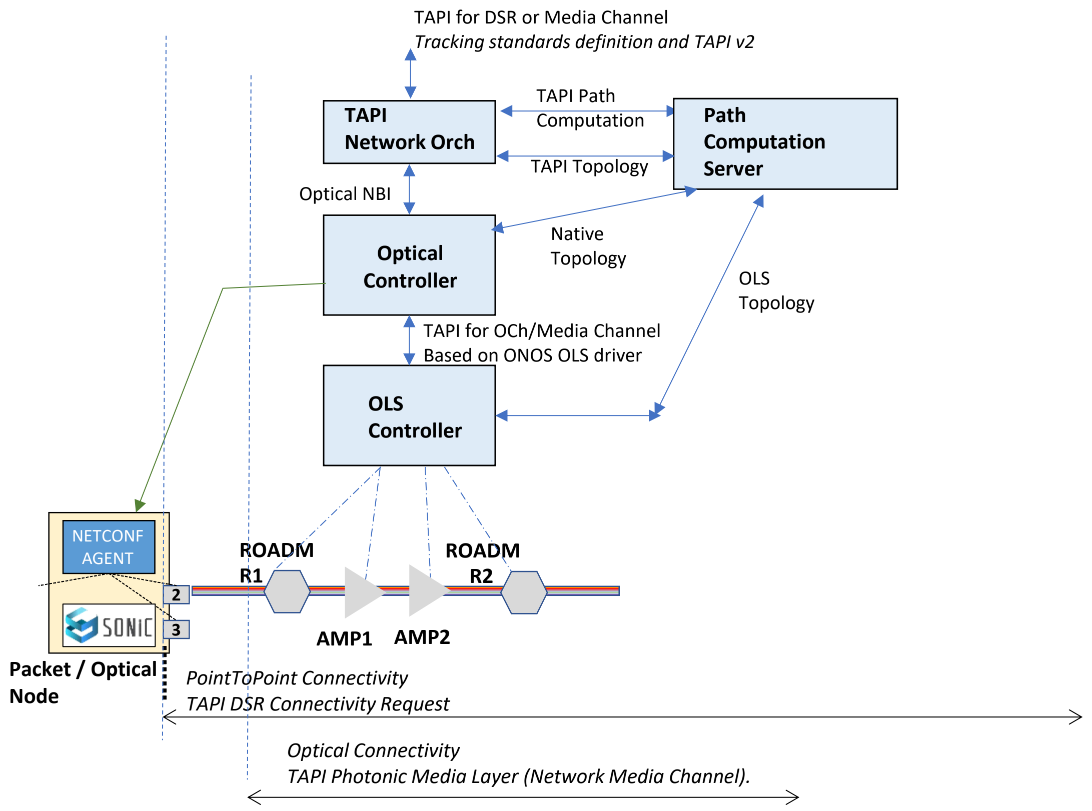


Figure 6-2 Transport API (T-API) Optical Network Orchestrator with the BG5-Open control plane functional architecture, showing the usage of an externalized path computation function

The core of the TAPI Optical Network Orchestrator controller is an asynchronous event loop (see Figure 6-3). 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 OLS 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.

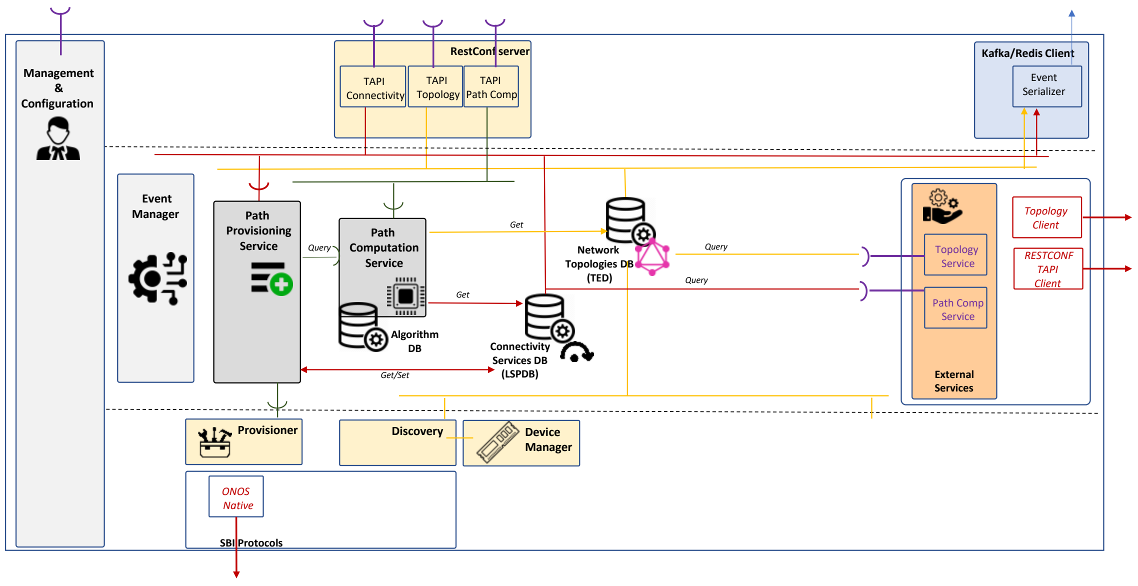


Figure 6-3 Internal diagram of the Transport API (T-API) Optical Network Orchestrator with externalized path computation

6.2.2 Exported North Bound Interface

As stated, the TAPI Optical Network Orchestrator will adopt the uniform TAPI interface. B5G-OPEN contributes actively, in cooperation with WP6, to the standardization of this interface and data models. It defines a set of core information models and layering to represent optical networks and services, as shown in the Figure 6-4.

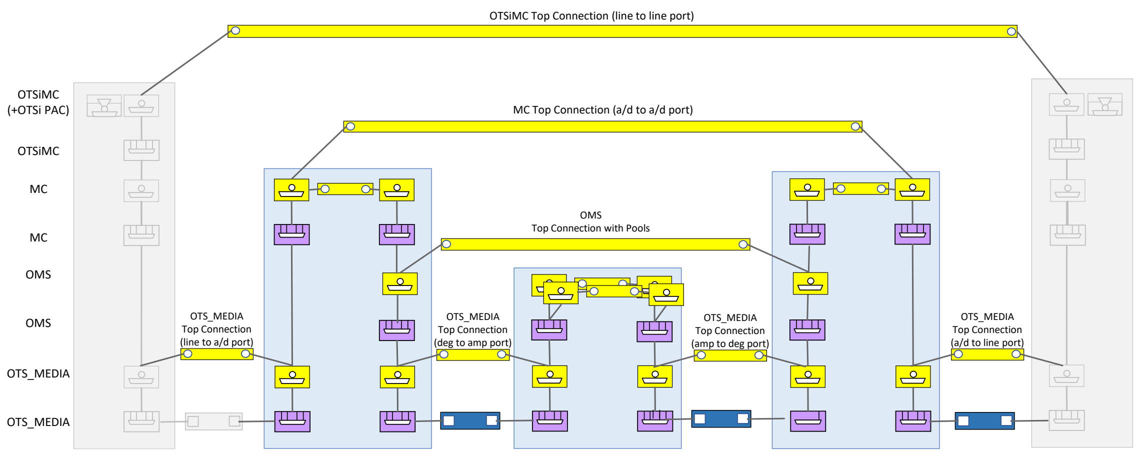


Figure 6-4 Transport API (T-API) Optical Network Orchestrator: logical view of a TAPI topology for an optical domain, based on the TAPI Core Information model

6.3 Optical Controller

The optical controller is based on ONOS SDN controller that provides a wide environment (including the support of all the most relevant control protocols toward the data plane) that besides the control of optical devices and OLS will also be utilized for the control of packet devices. In particular, the main roles of the optical controller are: (i) retrieve devices description from data plane device and abstract it toward the upper layers of the control pane; (ii) receive the service configuration requests by the upper layers of the control pane (e.g., the activation of a point-to-point connectivity service) and translate this request is a set of configuration messages to be forwarded to each involved device.

The main interfaces that will used in ONOS to interact with the other B5G-OPEN control plane elements are: (i) the ONOS native REST-based northbound APIs will be used to interact with the TAPI Optical Network Orchestrator and with the Path computation server, such interfaces can be used for both receiving configuration instruction to be applied on the data plane and exporting topological and physical impairments information; (ii) NETCONF/YANG based interface will be used toward data plane optical devices allowing the configuration and management of such devices whose description could be based on standard (e.g., OpenConfig, OpenROADM) or proprietary YANG models; (iii) a RESTCONF based interface toward the OLS controller. Moreover, ONOS also provides additional interfaces to visualize and configure the underlying network such as a web-based GUI and a CLI.

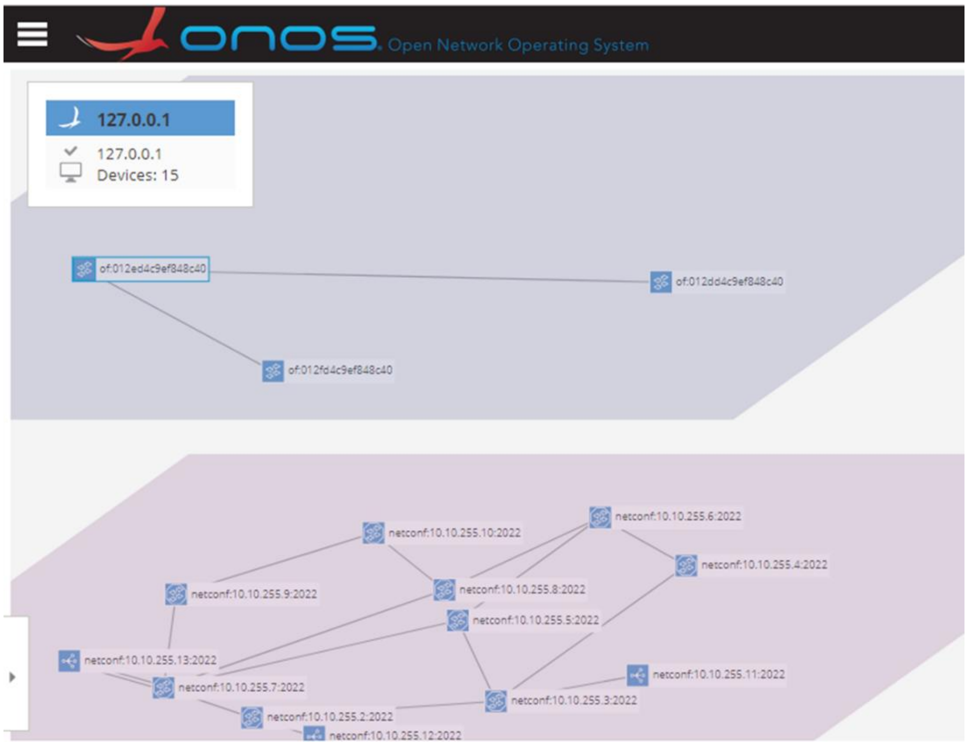


Fig 6.5: Web-based GUI of ONOS where both a number of packet-based and optical devices (fully disaggregated scenario) are controlled.

The current version of ONOS is already able to connect to a variety of packet-based and optical devices. However, interfaces toward the optical devices should be extended through the development of specific drivers, moreover existing drivers should be accurately tested and probably updated against the most recent version of standard models (e.g., last tested version of TAPI drivers toward the OLS controller was based on TAPI 2.1). Other development work will be required in ONOS for: (i) introducing the support of multi-band, (ii) exportation of physical impairment device manifest; (iii) introduce the possibility to activate intents using as end-point the ROADM's ports; (iv) extends the NBI REST APIs to enable proper integration with the TAPI Optical Network Orchestrator and the Path computation engine.

6.4 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.

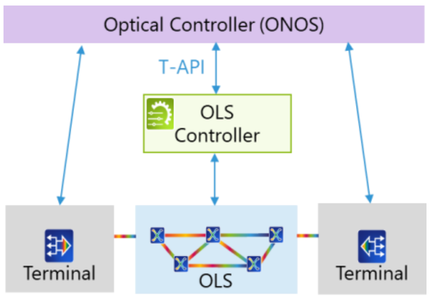


Figure 6-5 ADVA OLS Controller Northbound Interfaces

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. The current release supports only a single topology, therefore the tapi-topology:topology-context/tapi-topology:nw-topology-service object is not currently implemented.

SIPs represent the available service entry points. SIPs associate to all OCh and PHOTONIC_MEDIA NEPs in the network support service configuration. A SIP logically maps to one topology NEP through the tapi-topology:owned-node-edge-point/mapped-serviceinterface-point attribute.

The TAPI topology data model consists of nodes and links. A node is a logical grouping of ports that provide a flexible view definition. For example, one view might represent the topology one-to-one, whereas another view can represent an entire network as a single logical node.

The current implementation delivers a single default context, with a single topology composed of:

- tapi-topology:node
- tapi-topology:link

The interface represents each physical node as a multilayer **tapi-topology:node** object, which creates a 1:1 logical-physical topology. The forwarding domain is the domain associated with the entire physical network element. The OLS TAPI interface does not report any information about the internal structure of the network element. Each node displays: tapi-topology:node-edge-point. Each NEP represents an externally visible port that belongs to the node. The TAPI interface does not report any information about the internal structure of the network element.

Each NEP represents:

- A client port
- An OTS port
- An OMS port
- An OCH trail termination point

The figure below shows an example of a dis-aggregate OLS with three ROADM nodes:

- Node A: a fully flexible ROADM with client traffic that enters from a filter.
- Node B: a pass-through ROADM.
- Node C: as NE A, with client traffic that exits from a filter.

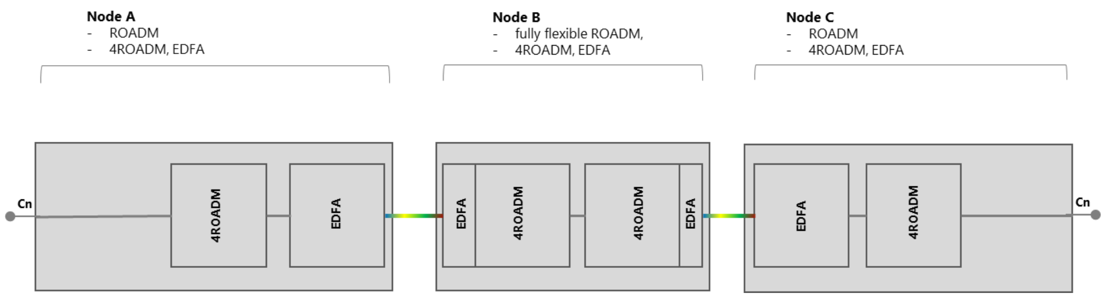


Figure 6-6 ADVA disaggregated OLS network example

This figure below shows the model instantiated on the TAPI interface in this scenario:

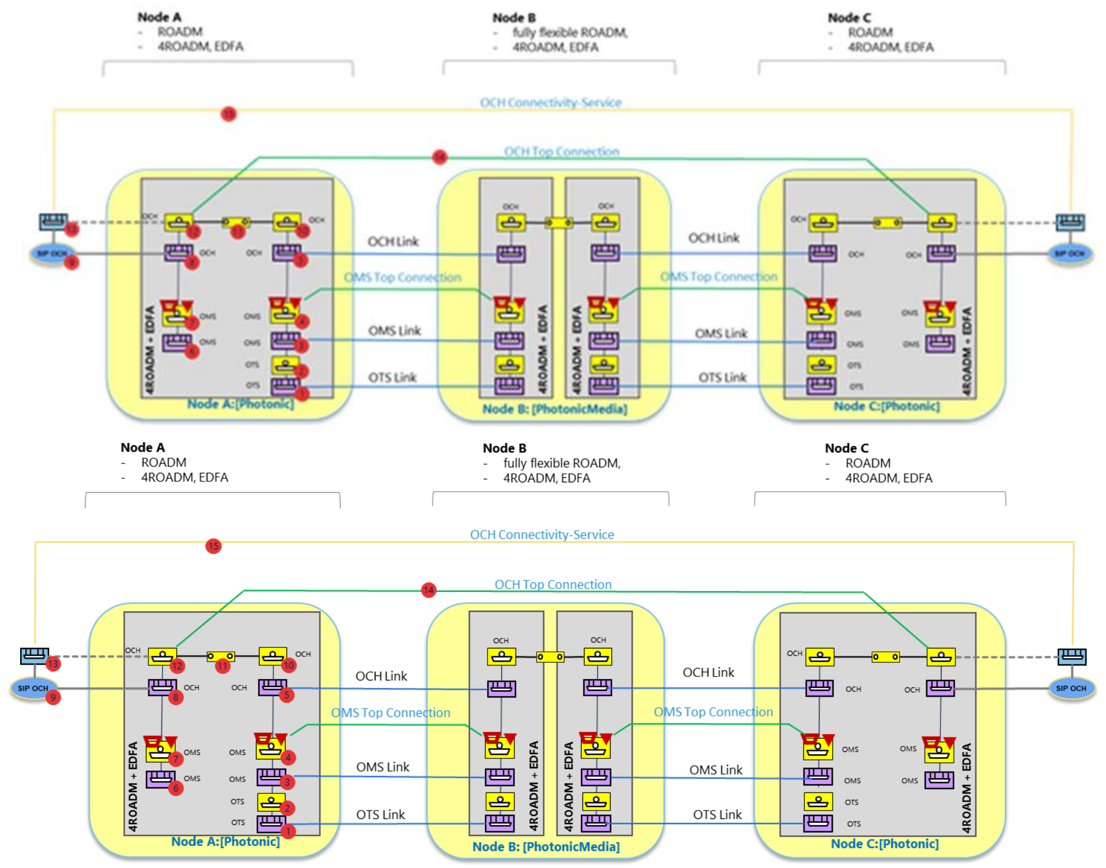


Figure 6-7 Corresponding model instantiated on the TAPI interface

6.5 Optical Path Computation Element

SDN controllers establish connections between the network elements, but they may not have an overall view of the network or may decide to rely on an external path optimization engine to make advanced specialized computations. This approach has been extensively applied in the past in other contexts (e.g., GMPLS or IP). See [Pao13] for a historical review.

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 will be 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.

A representative workflow of a typical interaction will be:

- The Network dimensioning engineer registers the OPCE in the TAPI Optical Network Orchestrator SDN controller by its IP address and port. The service TAPI is known by both entities.
- The TAPI Optical Network Orchestrator receives a new connection request with some requirements (source and destination, bandwidth provision, latency constraints, QoT conditions, etc.). This element detects the registered OPCE and forwards the request to OPCE.
- The OPCE receives the request via TAPI, obtains the parameters as SIPs (end-points) and topology model description and solves the path computation problem.
- Once computed the results, the OPCE sends the reply to the Optical Network Orchestrator Controller. It internally updates a topology resources usage view, that will consider the path as active, until the TAPI SDN controller instructs the OPCE about its release.
- When the Optical SDN Controller receives the reply, it is responsible for the route signalling between network elements. Additionally, as mentioned, it is responsible of informing the OPCE about the path release, when it happens to occur.

The path computation function is constrained to provide optical paths and potentially spectrum assignments, that end into viable network configurations, e.g., without spectrum clashing. Additionally, optical impairment computations may be triggered, to assess the Quality of Transmission (QoT) properties of the new connection, and of the already existing co-propagating connections, that may be affected by the new signal.

The QoT estimation is a very relevant aspect in the presence of optical multiband, where the increase of the fiber propagating total signal power and extended spectrum will stress the OSNR and power margins. For this aspect, the optical OPCE will integrate an internal or external optical signal performances engine. For this, the project may consider different approaches, as the new developments, or modifications/incorporations of existing open-source models like the ones in the GnPy initiative [GNPy]. GNPy is an open-source library for building route planning and optimization tools in real-world mesh optical networks. It is based on the Gaussian Noise Model, and has been present in the last years in different research efforts (e. g. see [Ferra20]).

6.6 MULTI-DOMAIN SCENARIOS

Of special interest for B5G-OPEN is the “multi-OLS scenario”, (see Figure 6-8) 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.

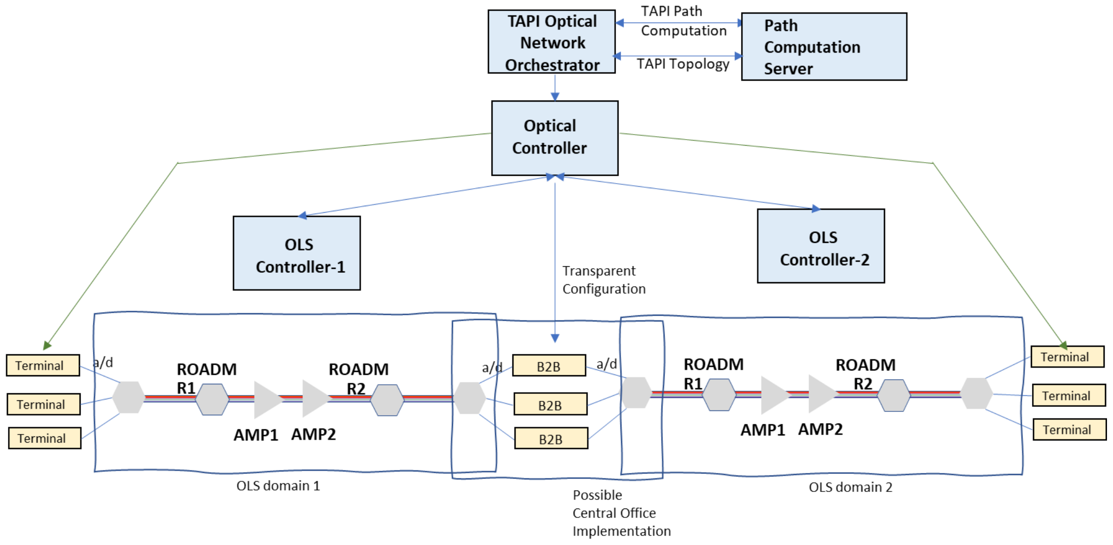


Figure 6-8 Control plane architecture for the multi-OLS scenario, showing a back to back add/drop-add/drop configuration.

7 Access 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. During the next paragraphs, initials assumptions, as well as alternative architectural considerations are presented regarding such control.

7.1 The Framework of TDM-PON Configuration and Control

The B5G-OPEN TDM-PON infrastructure will be realised using an XGS-PON OLT pluggable transceiver (e.g., TiBit pluggable) and a couple of pluggable ONUs (e.g., Tibit ONUs). The OLT will be interfaced directly to a whitebox switch while the OLT is interconnected to the ONUs by means of splitters, forming up an ODN branch.

The TiBit pluggables will be purchased from a third-party company (such as Juniper). Regarding the control s/w for the pluggables, the project will consider the option of adapting the vendor available software or develop an ad-hoc TDM-PON controller based on the OLT PON SDK. The integration of these pluggables with the B5G-OPEN software platform is made feasible at three different levels (from higher to lower layer):

- Via the PON Manager
- Via a PON Controller
- Direct through the OLT PON SDK or CLI

These options lead to four alternatives for the implementation of TDM-PON's control-plane, presented in the next subsections.

7.1.1 First Alternative: Via the PON Manager

In this case, the PON vendor provides both the pluggable devices and open software for the control and management of the TDM-PON. For example, Juniper supports this product via a *MicroClimate* management system which may manage all TDM-PONs in a domain. At the northbound interface, MicroClimate provides a set of APIs based on NETCONF or RESTCONF, while the BBF/ITU YANG model ([BBF-TR385] definition, [BBF-GIT1] implementation) is the common method to model the TDM-PON configuration parameters. Similar solutions are provided from other companies. In this case, the TDM-PON control-plane architecture and the steps to carry out the integration are illustrated in Figure 7-1, and are as follows: a Higher-Layer PON Controller is developed as part of the B5G-OPEN software platform, which includes the following functionalities:

- A NETCONF/REST client on the Southbound Interface (SBI) through which the communication with the PON Manager the vendor has developed (e.g. the MicroClimate in the Juniper's case)
- A set of PON abstractions, the objective of which is to extract the PON parameters and their values and then to expose to the higher layers only the parameters that are valuable for the B5G-OPEN software platform.
- A NETCONF/RESTCONF server at the Northbound Interface (NBI) which exposes a set of APIs that allow the B5G-OPEN app to provision and configure the PONs. This API

is using a simplified (subset) BBU/ITU YANG model which depend on the abstraction and transformation realised in the lower layer.

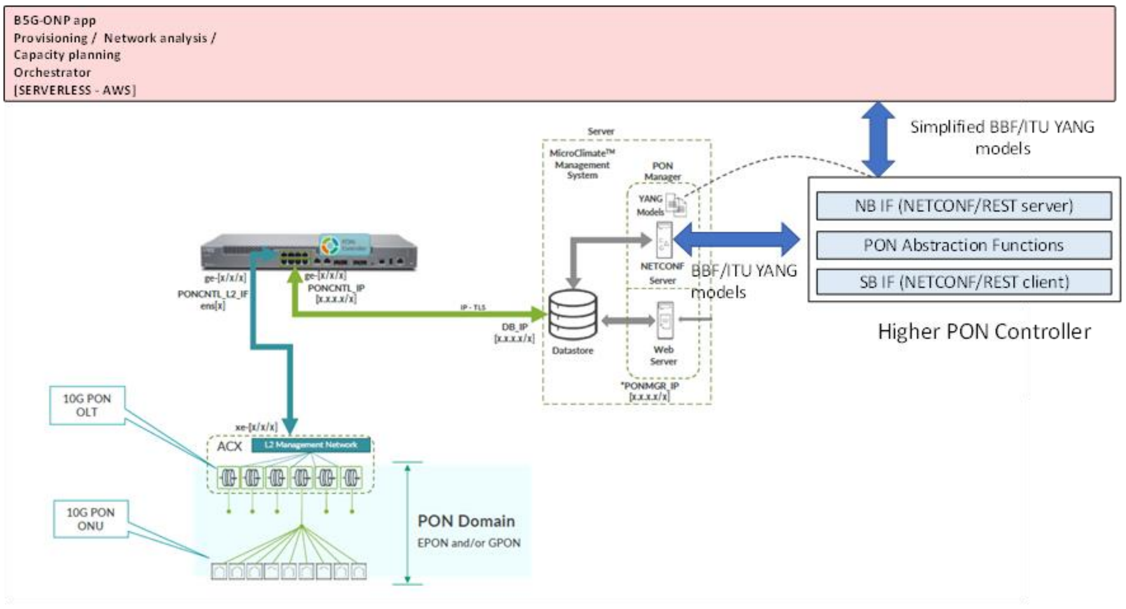


Figure 7-1 B5G-OPEN Control of PON through the PON Manager

7.1.2 Second Alternative: via a PON Controller

In this second 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 7-2.

Since the PON Controller will be provided by the PON vendor, a Higher-Layer PON Controller will be 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.
- Similar to the previous case, a set of PON abstractions is developed that extracts the PON parameters and their values. Only the valuable for the B5G-OPEN software platform set parameters are exposes to the higher layers.
- 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) BBU/ITU YANG model which depend on the abstraction and transformation realised in the lower layer.

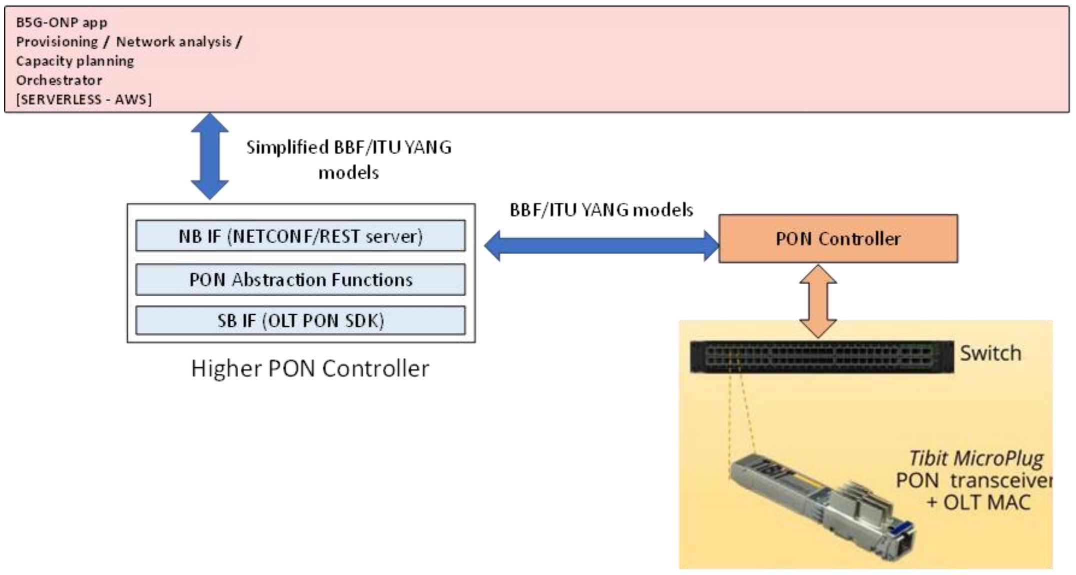


Figure 7-2 B5G-OPEN Control of PON through the PON Controller

7.1.3 Third Alternative: Direct interfacing to the pluggable OLT – Integration to the platform via the B5G-ONP app

In this case, the PON vendor provides only the basic software for the operation of the pluggable. The TDM-PON control-plane architecture and the proposed approach for the integration with the B5G OPEN platform are illustrated in Figure 7-3. Two software components are developed as part of the B5G-OPEN software platform: a) a PON agent that allows a direct communication with the pluggables; b) a PON controller through which the PON is controlled.

The PON agent consists of the following parts:

- A software client is based on an OLT PON SDK or it is set by establishing a CLI (or similar) connection with the OLT pluggable at the NBI.
- A REST server for the communication with the PON Controller.

In addition, the PON controller will include the following functionalities:

- A REST client developed on the SBI for the communication with the PON Agent. The information exchange is again based on the BBU/ITU YANG models.
- A set of PON abstractions, similar to the previous cases.
- A NETCONF/REST server on the NBI for the communication with the B5G-ONP app.

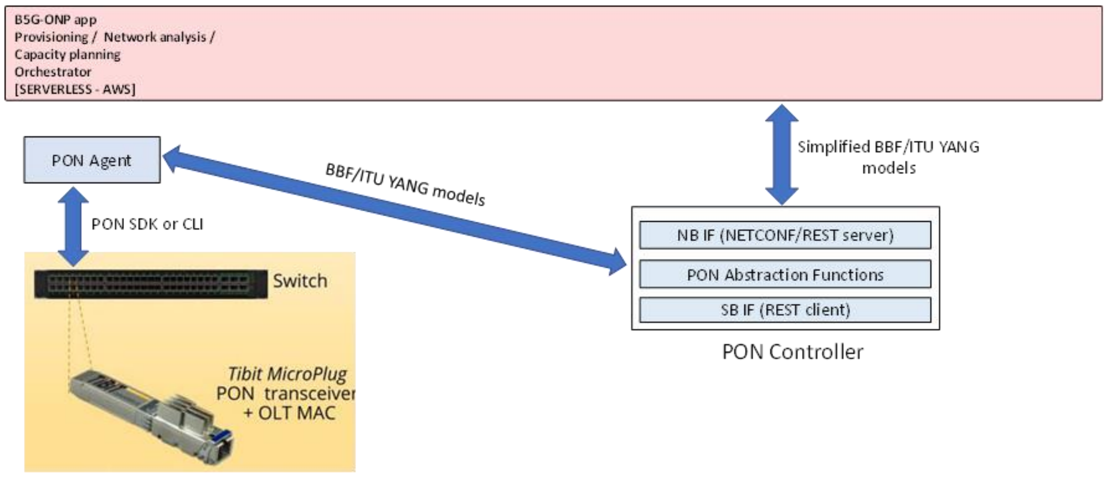


Figure 7-3 Direct communication with the pluggables (B5G-ONP app integration)

7.1.4 Fourth Alternative: Direct interfacing to the pluggable OLT – Integration to the platform via the B5G Packet Controller

This final alternative differs from the previous one in the way the control elements are integrated to the remaining B5G-OPEN software components. Therefore, while in the previous case the integration is realised with the B5G-ONP app, under the current framework, the integration is realised at a lower hierarchical level, i.e., by means of the packet controller. The TDM-PON control-plane architecture and the proposed approach for the integration with the B5G OPEN platform are illustrated in Figure 7-4.

In this case, a PON agent needs is developed as an integral part of the B5G-OPEN software platform, featuring the following functionalities:

- A software client is developed by means the OLT PON SDK or by establishing a CLI (or similar) connection with the OLT pluggable on the SBI.
- A set of PON abstraction functionalities, similar to the previous alternatives. In this case, the PON configuration should be represented as a set of nodes and links. In addition, in order to support some basic QoS on access connections, the upstream queue configuration should be also included in the modelling.
- A REST server/client on the NBI for the communication with the Packet Controller.

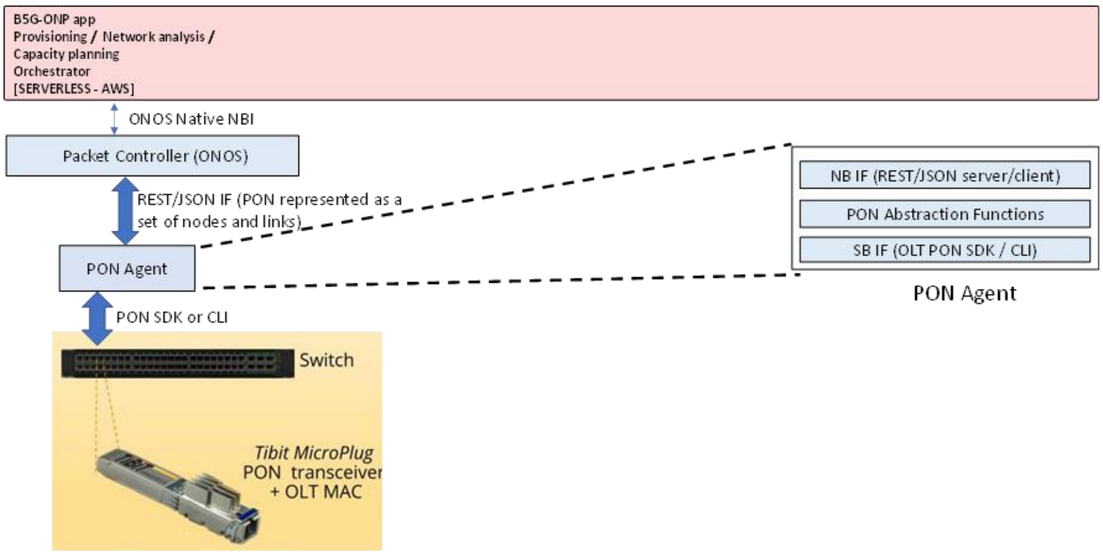

B5G-ONP app
Provisioning / Network analysis /
Capacity planning
Orchestrator
[SERVERLESS - AWS]
ONOS Native NBI
Packet Controller (ONOS)
REST/JSON IF (PON represented as a set of nodes and links)
PON Agent
PON SDK or CLI
Switch
Tibit MicroPlug PON transceiver + OLT MAC
NB IF (REST/JSON server/client)
PON Abstraction Functions
SB IF (OLT PON SDK / CLI)
PON Agent

Figure 7-4 Direct communication with the pluggables (Packet Controller integration)

7.1.5 Discussion on the alternative considerations

The alternatives listed in the previous subsections all have advantages and disadvantages. The option that delegates the integration of the B5G-OPEN platform to a higher layer is not the preferred one as it is neither ubiquitous nor vendor agnostic. Similar arguments hold for the second alterative (i.e., to control the PON by means of a vendor provided controller), although the implications require further study. Therefore, the third and fourth alternatives are attractive as they exploit schemes where the information exchange bypass both the Manager and the Controller. Actually, the fourth alternative has a higher level of universality since the PON control will lay below the packet optical control and therefore it can be controlled like an ordinary switch. However, this fourth alternative is facing its own challenges which emerge from the abstraction models and the representation of PON's QoS parameters that seems not to be trivial. As such, the third alternative presents itself as a good compromise between the pros and cons and it emerges as the preferable option at the time of the writing of this deliverable.

7.2 LIFI CONTROL

The LiFi access networks will be provided by Access Points (APs), named LiFi-XC, provided by pureLiFi. This LiFi AP device, as illustrated in Figure 7-5(a), converts network information coming in from the Ethernet port into wireless light signals via the connected LED lamp for the downlink. The uplink signal will be sent via the user device in Infrared spectrum and is captured by the AP device. The AP structure is shown in Figure 7-5(b). The AP is implemented using an embedded Linux device to bridge Ethernet connection with LiFi interface implemented with baseband processor and analogue front-end devices. The AP supports automatic provisioning using TR-069 protocol as well as simple network management protocol (SNMP) v1, v2c and v3.

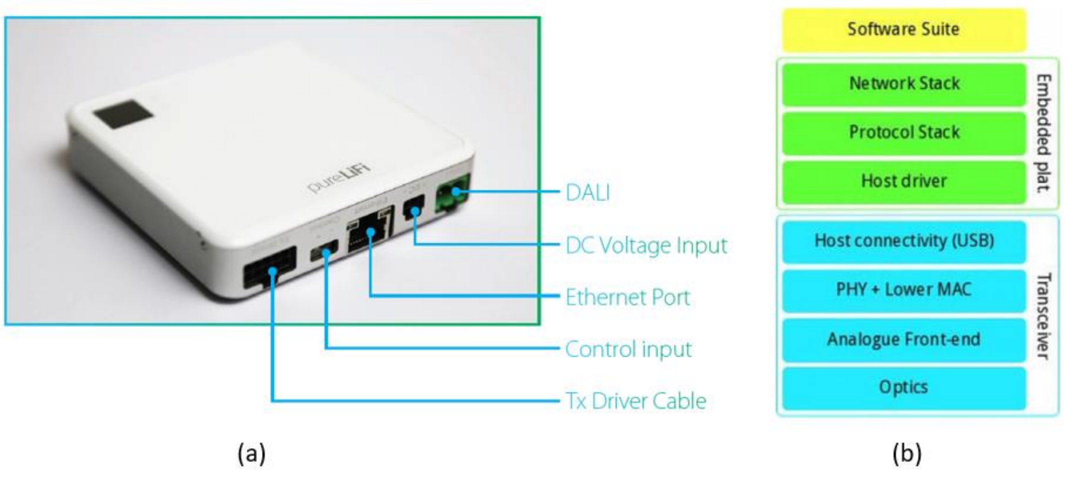


Figure 7-5: LiFi-XC AP

As the development in B5G-OPEN, these following supports will be implemented in the LiFi AP, as shown in Figure 7-6:

1) In the initial assumption of LiFi control for B5G-OPEN, it will be implemented to support for NETCONF interface. A LiFi specific YANG model will be proposed with basic configurations to configure 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) A telemetry adaptor will be implemented within LiFi AP for LiFi telemetry data collection and transmission. Some telemetry data could be used for monitoring the system performance, such as the received signal powers, the transmit and received throughput, etc. For LiFi specifically, some other information could also be used for control purpose. For example, since in LiFi the coverage of each AP is much smaller compared to other radio based wireless access technologies, by simply knowing the SSID of the AP which the user connects to, the location information could be obtained. In addition, the SSID of APs which are not being connected as well as their inactive time could be obtained via telemetry data, then some actions could be taken to save the energy consumption smartly such as by dimming these LED transmitters.

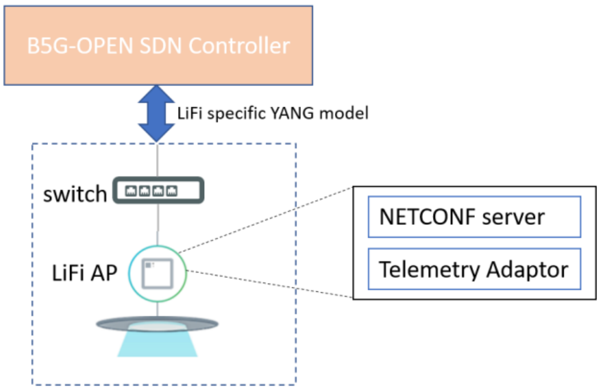


Figure 7-6: Initial assumption for LiFi control

8 ORCHESTRATION

8.1 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. This definition will be sufficient for this section. A discussion on the particular form in which the slice concept is elaborated in B5G-OPEN, is addressed in Section 12.

The coordinated optimization of both IT and network resources has been shown as clearly beneficial in a number of research works in the community [Gar20], [Ped18], [Muq21]. Intuitively, it is easy to find to examples where trivial blocking situations occur if such a coordination is not present (e.g., placing IT application in clusters without enough network connectivity to accommodate the application traffic).

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.

Figure 3-1 represents the macroscopic B5G-OPEN architecture and service interfaces including a representative infrastructure example, that will help us to illustrate how the coordination is performed. The figure focuses on a simplified network composed of two locations (left and right), with an IT cluster and a packet-optical white-box in each of them. The white-box includes a number of optical coherent pluggables. Optical transparent paths are handled by an optical line system (OLS) controlling three ROADMS. The optical network is controlled by the optical SDN controller, which is accessed via a TAPI Optical Network Orchestrator, that exports a standardized TAPI North Bound Interface (NBI). The configuration of the white-boxes in its packet forwarding-related aspects is handled by the IP SDN controller.

During the provisioning process, the B5G-ONP receives the commands from the user (e.g., via a graphical user interface, or via the open API exposed). The type of services that can be provisioned are discussed in Section 12. To accomplish the provisioning, the B5G-ONP leverages on the Kubernetes systems and SDN controllers as shown in the figure.

Note that the described architecture, permits implementing different strategies for the allocation decision process. For instance, the B5G-ONP can delegate in the SDN controllers the network path computations, or alternatively instruct the SDN controllers which paths to allocate according to its centralized decision. We believe such flexibility is a key benefit of this approach, that makes it eligible for different use cases and network scales.

8.2 B5G-ONP MODULES

B5G-ONP consists of three main modules (see Figure 8-1):

- **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 12, 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 will be 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 will be 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.

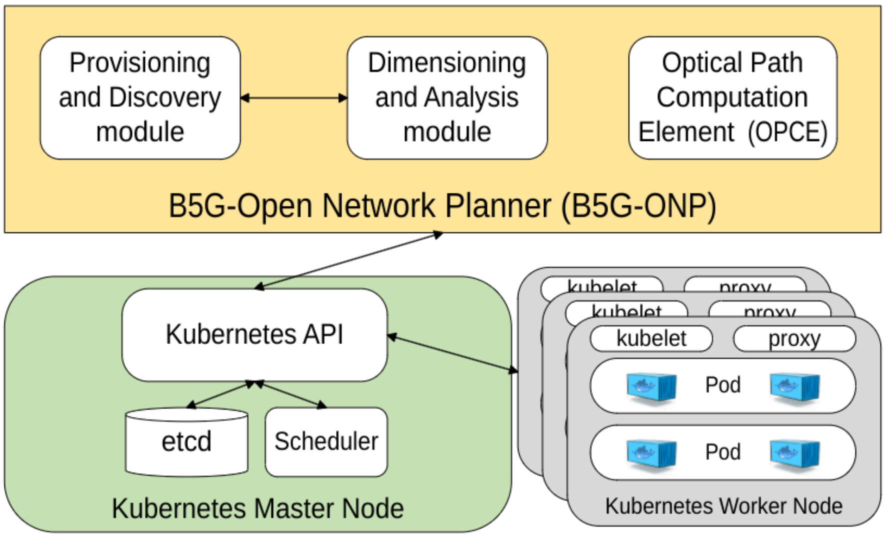

Provisioning and Discovery module
Dimensioning and Analysis module
Optical Path Computation Element (OPCE)
B5G-Open Network Planner (B5G-ONP)
Kubernetes API
etcd
Scheduler
Kubernetes Master Node
kubelet
proxy
Pod
Pod
Kubernetes Worker Node

Figure 8-1 Coordination of Kubernetes cluster from B5G-ONP

8.3 INTERACTIONS OF THE B5G-ONP WITH THE SDN CONTROLLERS

The B5G-ONP will interact with the SDN controllers in order to instruct them in the provisioning workflows. For this, the interactions will be implemented via the regular North Bound Interface (NBI) APIs of the controllers, following the best practices. Additionally, slow-changing monitoring resource occupation information, suitable for provisioning use cases, may be obtained from: i) SDN controller NBIs; ii) network telemetry systems. The decisions on which particular performance indicators will be accessed and from where will be elaborated along the project evolution.

8.4 INTERACTIONS OF THE B5G-ONP WITH THE IT ORCHESTRATOR SYSTEMS

As better discussed in Section 12, B5G-OPEN will explore Kubernetes as the baseline orchestration system in its main efforts, due to benefits like microservice-orientation, agility, maturity of its APIs, and industry adoption.

In a typical microservice-based deployment, the microservices run independently from others and communicate between them using well-defined RESTful APIs and synchronous protocols such as HTTP. This method is light, fast, easy to spin up and allows scaling only those microservices that require more resources to achieve a proper load distribution. Nowadays, it is of common use within the most popular cloud providers such as Google, Microsoft and Amazon. Kubernetes eases the administration tasks because it automates and scales the processes, not being aware of the internal tasks.

Kubernetes coordinates a highly available cluster of computers that are connected to work as a single unit without specifying the individual machines. The containerized applications decouple the deployment and applications from individual hosts. Figure 8-2 shows how the Kubernetes cluster operate. The Kubernetes cluster consists of two resources, a Master Node coordinates the cluster and Working Nodes that run applications (worker machines). The communication between the Master Node and the Working Nodes is realized by the Kubernetes API exposed by the Master Node [K8s].

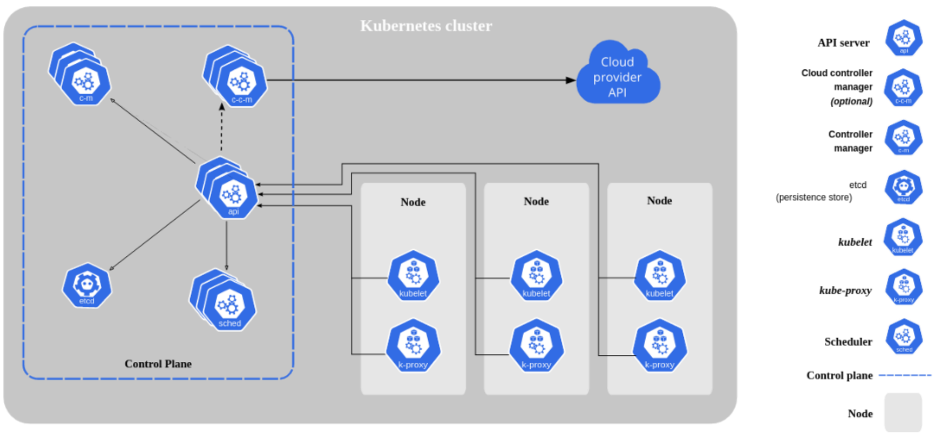


Figure 8-2 Kubernetes cluster module (source: [K8s])

The actual deployed services are distributed in the network onto different physical or virtual nodes and require high-performance network connections to be able to provide optimal communication (e.g., min latency). Kubernetes dynamically orchestrates the services and eases this task for the users.

Kubernetes management is based on two key concepts: a Kubernetes service, and Pods. A Pod is a group of one or more related containers, of the same service, that have to be deployed in the same worker node. A Kubernetes service is a component that typically represents an application, potentially composed of multiple pods, each of them that can be optionally deployed in different worker nodes, but with a common management. According to deployment requirements, the Pods of the service can be present on all the Working Nodes (DaemonSet) or some of them (Deployment).

To cover IT orchestration-related use cases, the key interactions that are anticipated between the B5G-ONP and Kubernetes deployments are:

1. B5G-ONP discovery of Kubernetes deployments. An API should be incorporated in the B5G-ONP to permit the registering of Kubernetes clusters, e.g., identified by its master node access information. Once a Kubernetes system is registered, its internal capabilities should be discovered via the Kubernetes API. After that, the Kubernetes resources will be available for B5G-ONP to allocate new B5G-OPEN IT services, e.g., as part of B5G-OPEN slices.
2. B5G-ONP provision and release of microservice-based applications in the IT resources of Kubernetes systems. The B5G-ONP will be able to jointly optimize the usage of IT and network resources in the allocation of new slices. For this, the B5G-ONP should interact with the Kubernetes APIs for covering the IT part allocation and resource releases in an automatic form.
3. B5G-ONP extraction of occupation and performance KPIs from the Kubernetes. In order to cover its network optimization and planning role, the B5G-ONP should have access to the different KPIs of the registered Kubernetes systems. For this, the project will explore the existing Kubernetes APIs.

Practical aspect of API interactions and potential API extensions

The communication between B5G-ONP and Kubernetes uses the HTTP REST API that Kubernetes exposes. The Kubernetes API [K8sAPI] lets you query and manipulate the state of API objects in Kubernetes (e.g., Pods, Namespaces, ConfigMaps, and Events). Kubernetes supports multiple API versions, each at a different API path, such as */api/v1* or */api/v2*. API resources are distinguished by their API group, resource type, namespace (for namespaced resources), and name. The API server handles the conversion between API versions transparently: all the different versions are representations of the same persisted data (see Figure 8-3). The API server may serve the same underlying data through multiple API versions.

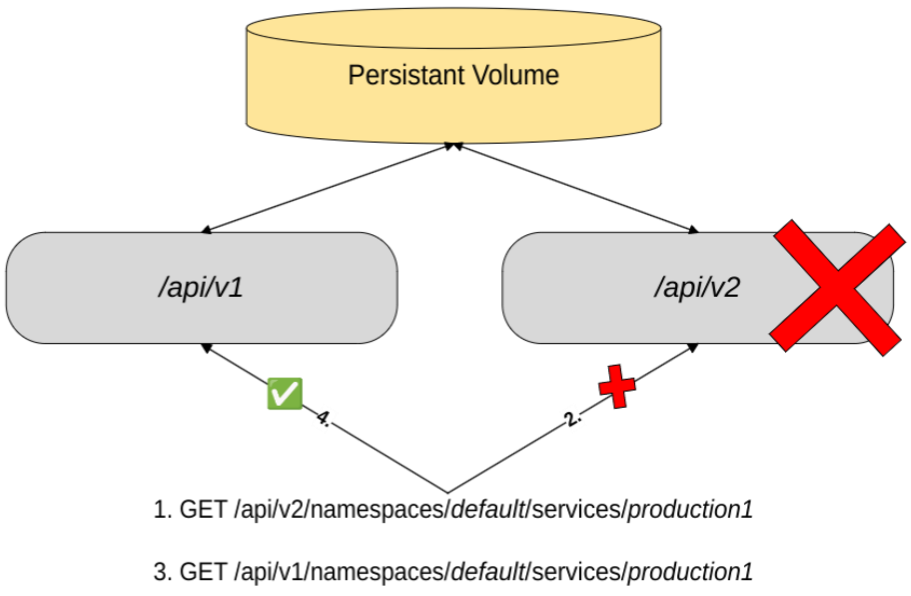

Persistant Volume
/api/v1
/api/v2
4.
2.
1. GET /api/v2/namespaces/*default*/services/*production1*
3. GET /api/v1/namespaces/*default*/services/*production1*

Figure 8-3 Different API versions access to a unique persisted data

Any system needs to grow and change as new use cases emerge or existing one change. Therefore, Kubernetes has designed the Kubernetes API to continuously change and grow. The Kubernetes project aims to not break compatibility with existing clients, and to maintain that compatibility for a length of time so that other projects have an opportunity to adapt. Additionally, Kubernetes provides two ways to add custom resources to your cluster: Custom Resource Definitions (CRDs) and API Aggregation (AA).

- The CRD object definition creates a new custom resource with name and schema served and handled by Kubernetes API, with less flexibility than with AA. The CRD name must be a valid DNS subdomain name. No needs to handle multiple versions of the API, no additional services and does not require programming.
- AA, the user writes and deploys a custom API server allowing specialized implementations for customer resources. Once the main API server receive queries to custom API server, it forwards them.

These two alternatives may be explored in B5G-OPEN, in case that a Kubernetes needs to be extended to accommodate project needs.

9 PACKET/OPTICAL INTEGRATION

9.1 DISCUSSION ON ARCHITECTURAL OPTIONS

Traditional metro networks are composed by packet switching nodes (i.e., routers) interconnected by optical transport links. In this scenario, packet and optical domains are clearly separated, using dedicated controllers. However, standalone muxponders and transponders are going to be replaced in optical metro and transport networks by the utilization of hybrid packet-optical nodes equipped with coherent pluggable transceivers. In this scenario, traditional packet control plane is not adequate because it is unable to manage and fully support the configuration of optical parameters associated to pluggable modules. Moreover, the coordination between the optical and the packet layer within this novel hybrid nodes has not been standardized yet and requires a careful design in order to enable correct configuration and avoid management conflicts.

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 [Sca21, Sga21, Ger22]. This section expands upon the aforementioned works and provides implementation details, experimental comparison and discussion on the possible solutions.

9.1.1 Reference scenario and proposed solutions

Figure 9-1 shows a traditional metro network using packet switching nodes (i.e., routers) and stand-alone transponders interconnected through optical line systems (OLSs). Where OLS are 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.

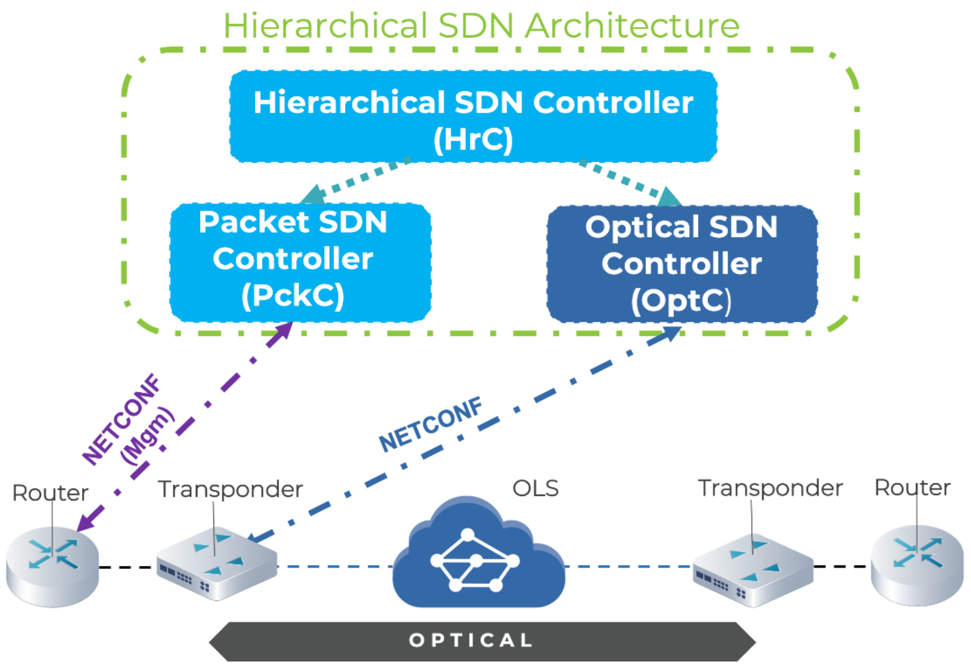

Hierarchical SDN Architecture
Hierarchical SDN Controller (HrC)
Packet SDN Controller (PckC)
Optical SDN Controller (OptC)
NETCONF (Mgm)
NETCONF
Router
Transponder
OLS
Transponder
Router
OPTICAL

Figure 9-1 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.

Figure 9-2and Figure 9-3shows the two solutions considered to provide coordinated control of packet-optical nodes, i.e., standalone optical transponders are not used in this scenario. With both solutions, the NETCONF protocol is considered for the pluggables configuration, while packet configuration can be performed via NETCONF or P4 protocols.

The first approach, here named Exclusive (Excl), is shown in Figure 9-2. The Excl approach provides access to the packet-optical nodes from the PckC only. That is, configurations related to both packet forwarding and optical pluggables are enforced by the PckC. In this case, the optical parameters (e.g., central frequency, TX power, operational mode) are decided a priori by the OptC and exchanged with the PckC through the HrC.

The second approach, here named Concurrent (Conc), is shown in Figure 9-3. The Conc approach relies on the joint control of the packet-optical nodes from both PckC and OptC. Configurations related to packet forwarding are provided by the PckC, while those related to optical pluggable modules are enforced directly by the OptC. In this case, proper solutions are needed to guarantee coordinated access as well as the proper control segregation for avoiding possible conflicts.

The Excl approach is also considered within the TIP project [TipMantra] referred as *single SBI management.* However, TIP also considers an intermediate solution, referred as *dual SBI management,* where the OptC has read access to the packet device transceiver information, but their configuration in actually enforced by the PackC that receives the required configuration values from the OptC via HrC. From an implementation point of view the *dual SBI management* solution significantly simplifies the Conc approach because it avoids the issue of having two controllers with write rights on the same device, but from a functional point of view it is equivalent to the Conc approach only introducing more latency on the communication channel between the OptC and the packet-optical node.

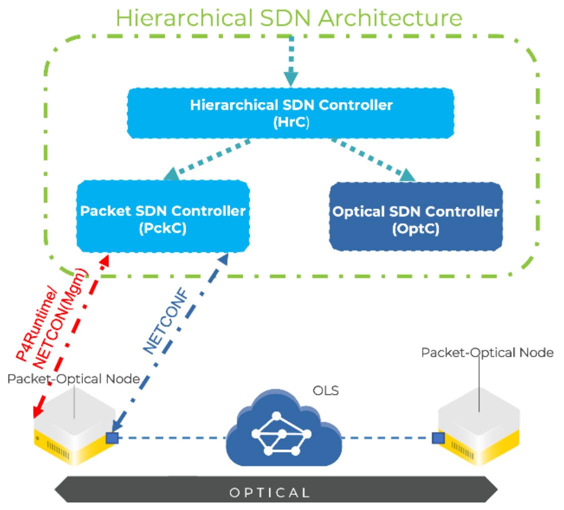

Hierarchical SDN Architecture
Hierarchical SDN Controller (HrC)
Packet SDN Controller (PckC)
Optical SDN Controller (OptC)
P4Runtime/ NETCON(Mgm)
NETCONF
Packet-Optical Node
Packet-Optical Node
OLS
OPTICAL

Figure 9-2 Exclusive hierarchical control plane solution

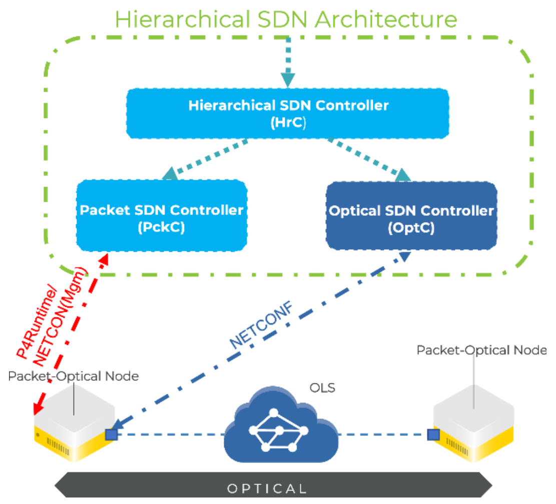

Hierarchical SDN Architecture
Hierarchical SDN Controller (HrC)
Packet SDN Controller (PckC)
Optical SDN Controller (OptC)
P4Runtime/ NETCON(Mgm)
NETCONF
Packet-Optical Node
Packet-Optical Node
OLS
OPTICAL

Figure 9-3 Concurrent hierarchical control plane solution

The workflow for the establishment of end-to-end intents involving both the packet and the optical layer is depicted in Figure 9-4using the Excl approach. At step 1, HrC receives a connectivity request and computes the related end-to-end path over the multi-layer network topology. If the activation of a new lightpath is required, HrC sends an optical intent request to the OptC (step 2). At step 3, OptC configures the OLS devices traversed by the lightpath and, once the optical intent is installed, a notification is sent to HrC, including the utilized optical parameters (step 4). At step 5, HrC shares with PckC the values notified by OptC (i.e., frequency, TX power and operational mode) and requests the setup of packet intent. PckC configures the pluggable modules and, once the link becomes active, it installs the packet intent (step 6). Finally, at step 7, PckC informs HrC that the packet connection was successfully configured.

The workflow for the establishment of an end-to-end intent using the Conc approach is depicted Figure 9-5. At step 1, HrC receives a connectivity request and computes the related end-to-end path over the multi-layer network topology. If the activation of a new lightpath is required, HrC sends an optical intent request to the OptC (step 2). At step 3, OptC configures the OLS devices and the pluggables involved in the lightpath and, once the optical intent is installed, a notification is sent back to HrC (step 4). In this case, the configured optical parameters are not notified to the HrC, because the configuration of the optical domain is fully managed by OptC. At step 5, HrC requests PckC to configure a new packet intent. PckC installs the packet intent (step 6) and informs HrC that the packet connection was successfully configured (step 7). In case a path with enough bandwidth already exists in the optical layer, steps 2, 3 and 4 are skipped by HrC, directly moving to step 5 for the installation of the packet intent. In case a path with enough bandwidth already exists in the optical layer, steps 2, 3 and 4 are skipped by HrC, directly moving to step 5 for the installation of the packet intent without requiring the activation of additional optical pluggables.

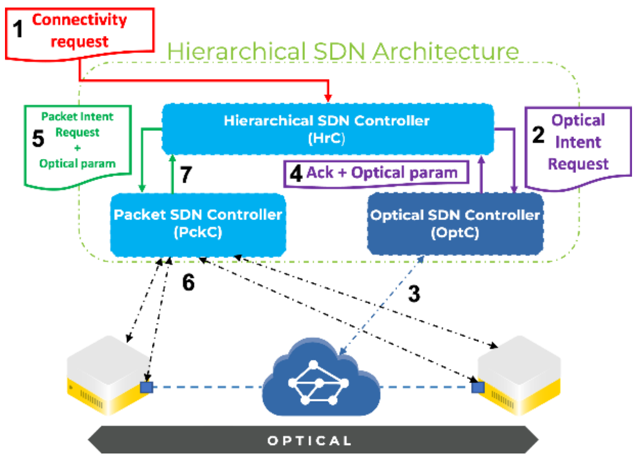

1 Connectivity request
Hierarchical SDN Architecture
Hierarchical SDN Controller (HrC)
5 Packet Intent Request + Optical param
2 Optical Intent Request
7
4 Ack + Optical param
Packet SDN Controller (PckC)
Optical SDN Controller (OptC)
6
3
OPTICAL

Figure 9-4 Exclusive end-to-end intent setup workflow

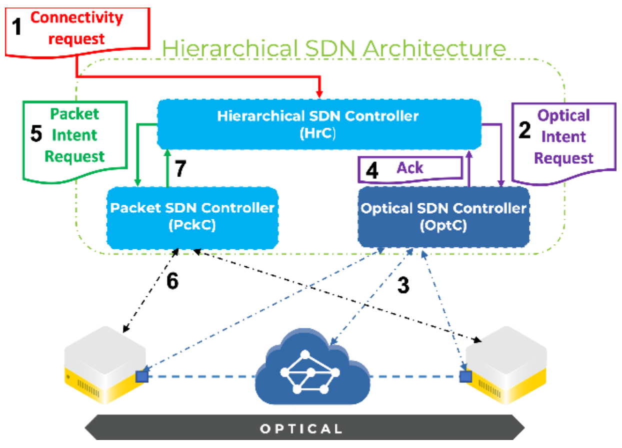

1 Connectivity request
Hierarchical SDN Architecture
Hierarchical SDN Controller (HrC)
5 Packet Intent Request
2 Optical Intent Request
7
4 Ack
Packet SDN Controller (PckC)
Optical SDN Controller (OptC)
6
3
OPTICAL

Figure 9-5 Concurrent end-to-end intent setup workflow.

9.1.2 Packet-optical node

For comparing the two approaches an experimental testbed has been setup including a preliminary implementation of a packet-optical node where the optical pluggable is emulated using an external transceiver. This solution has been preliminary adopted because nodes supporting the utilization of pluggables are not yet available in the partner labs. Other solution will be used during the project when more advanced hardware will be available, specifically two options are under evaluation: (i) using a single device with coherent pluggables and P4/P4 Runtime support; (ii) using two separate devices connected through a direct packet link: one device with P4/P4Runtime support and a second device with basic packet processing features and coherent pluggables support.

The internal software architecture of the considered packet-optical node is shown in Figure 9-6. A Mellanox/NVIDIA SN2010 Ethernet switch, running the SONiC NOS has been used. In addition to the default SONiC applications, it includes additional containerized functionalities, i.e., the NETCONF agent container and the P4/P4Runtime container, depending on the considered scenario (i.e., Excl or Cunc).

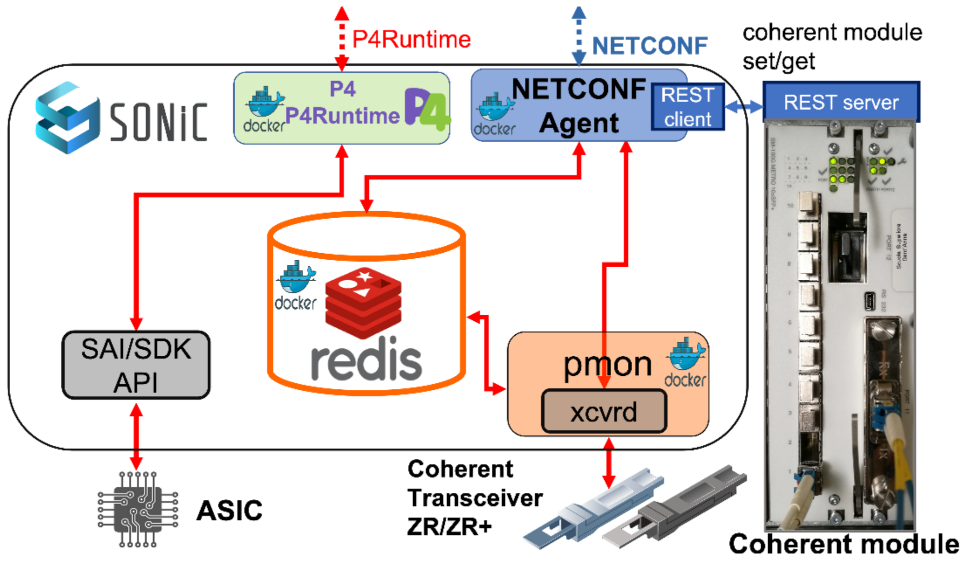


Figure 9-6 Packet-Optical node architecture: the packet-optical node exposes P4 Runtime and NETCONF interfaces toward the control plane.

The NETCONF agent container is used to configure and monitor optical pluggables. The agent uses the OpenConfig model for hardware representation, including ports and pluggables. To avoid misconfiguration issues when multiple controllers access the node, ownership segregation has been implemented using the Network Configuration Access Control Model (NCACM) solution, as detailed in RFC 8341. More specifically, for each configured user (i.e., PckC and OptC), a set of rules is configured in the NETCONF agent, permitting or denying operations (e.g., write, read-only) over specific prefix-based configuration parts.

The NETCONF container is able to direct access the C-CMIS driver of the physically-connected pluggable modules, managing the optical parameters. As we do not have yet coherent pluggable modules in our laboratories, an external coherent transceiver (e.g., Ericsson SPO xPonder with coherent 100G line ports) acts as pluggable. That is, its configuration is not provided by the SDN controller directly, instead, it is provided by the packet-optical node as for locally equipped transceivers. More in detail, when the controller interacts via NETCONF with the packet-optical node, the agent leverages a REST interface toward the coherent transceiver to configure/read the optical parameters, as it is a pluggable module attached to the box.

The P4/P4Runtime is the interface leveraged by the PckC to configure the packet layer. Such interface has been implemented using a containerized Bmv2. This container-based solution provides all the P4 features while processing all traffic at software level in the switch CPU.

Thus, switching performance is limited but perfectly usable for therefore usable validation and demonstration purposes.

9.1.3 SDN controller applications

The implementation of SDN controllers is based on ONOS. Specifically, the parent controller and the two child controllers has been implemented developing a dedicated ONOS NetApp (i.e., the HrC app, the OptC app and the PckC app).

- The HrC app retrieves and maintains the information on the status of entire network, communicating with child SDN controllers through REST APIs. For each connection request, it computes the end-to-end path and splits the computed path between the child controllers, relying on both packet and optical domains, i.e., where needed a new lightpath is installed in the optical domain.
- The PckC app allows the communication between PckC and HrC. Two versions of the application have been developed, implementing the Excl and Conc approaches. More in detail, the PckC app may be deployed with (or without) a Pluggable Manager module that is not required when con Conc approach is exploited because plugabbles are fully configured by the OptC.
- The OptC app enables the communication between OptC and the HrC, managing and retrieving the pluggables optical parameters. The OptC app is developed with or without the Pluggable Manager module according to the two considered scenarios, i.e., using the Excl approach such module is not required because pluggables are configured by PckC.

In addition, for supporting the Excl approach where the OptC only configures the ROADMs (while the configuration of the pluggables is performed by the PckC), the ONOS intent service has been extended to support intent requests whose end-points are ROADMs interfaces.

9.1.4 Experimental evaluation

The packet-optical testbed topology used for the experimental evaluation of the two considered approaches is illustrated in Figure 9-7 and includes two packet-optical nodes equipped with pluggable transceivers, three emulated ROADMs (e.g., OpenROADM NETCONF agents running in dedicated docker containers), and four P4-based emulated switches (e.g., running BMv2 software switch). Each P4 switch is emulated on a dedicated bare metal server (Intel Xeon E5-2643 v3 6-core 3.40 GHz clock, 32 GB RAM) and links are implemented through real physical interfaces (i.e., Mellanox ConnectX3 Network Interface Cards).

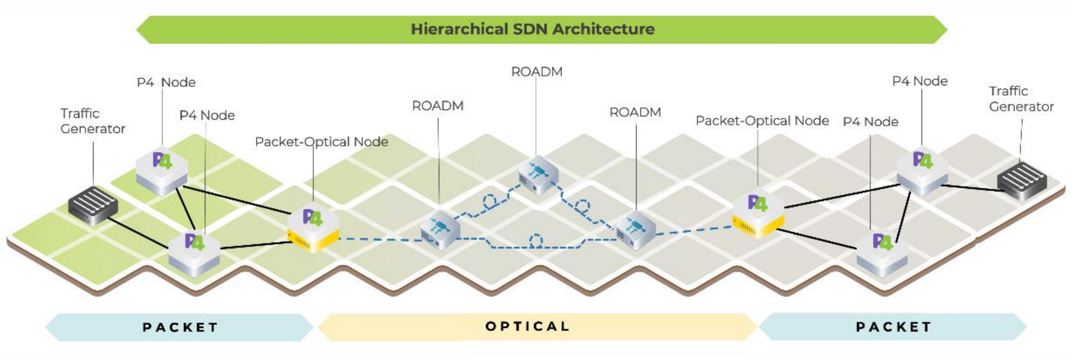


Figure 9-7 Testbed topology.

Each packet-optical node consists of a Mellanox/NVIDIA SN2010 Ethernet switch, which runs the SONiC operating system over ONIE. In addition to the basic SONiC components, the P4/P4Runtime and NETCONF docker containers have been added, as shown in Figure 9-6. As presented before, the xPonder Ericsson SPO with coherent line ports at 100G acts as pluggable coherent module in packet-optical nodes.

The scenarios presented in the previous sections have been validated starting from an empty network where no lightpaths are configured in the optical network, thus the packet domain is composed by two islands. The first goal of the proposed experimental test is to validate the procedure to establish an end-to-end intent (i.e., spanning across the packet and optical domains). Secondly, we have compared the end-to-end connection setup time (i.e., Te2e) obtained considering the two approaches repeating the connection setup experiment for 30 times. Obtained results are illustrated in the following.

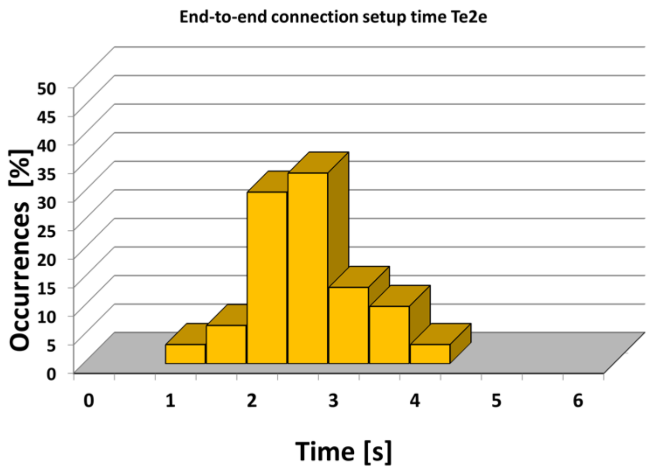


Figure 9-8 Distribution of Te2e over 30 experiments using Concurrent workflow, average value Te2e = 2.38 s.

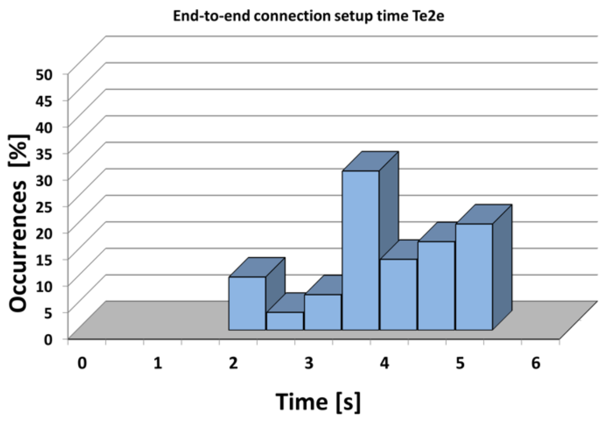


Figure 9-9 Distribution of Te2e over 30 experiments using Exclusive workflow, average value Te2e = 3.59 s.

The distributions obtained for the end-to-end connection setup time Te2e using the two considered workflows are respectively illustrated in Fig. H and Fig. I. The results prove that the Concurrent approach guarantees a faster Te2e, average value is faster of about 34% (3.59 seconds for Exclusive and 2.38 seconds for Concurrent). This happens because using the Exclusive approach, the configuration of the optical devices is performed by two different controllers (i.e., OptC configures the ROADMs, then PckC configures the pluggables). Thus, to allow the Exclusive workflow, additional information is required to be exchanged between the two child controllers (e.g., the lightpath central frequency decided by OptC) and therefore the configuration of pluggables is executed only when the configuration of ROADMs is fully completed. Conversely, with the Concurrent workflow the configuration parameters of ROADMs and pluggables are decided at OptC, and their actual configuration is triggered in parallel, thus leading to a faster end-to-end connection setup.

Within the B5G-OPEN project the Concurrent approach will be adopted for the control plane implementation. This choice that is mostly driven by practical implementation consideration is also supported by the experimental results described in this section, that indicate Concurrent option as the one potentially achieving faster end-to-end configuration.

9.2 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 containerized architecture allows reducing coupling between independent modules and between the functionalities implemented by the module and the platform-specific lower-layer details. Moreover, it allows easy extensibility because new features may be added as additional containers without any impact on existing ones.

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.

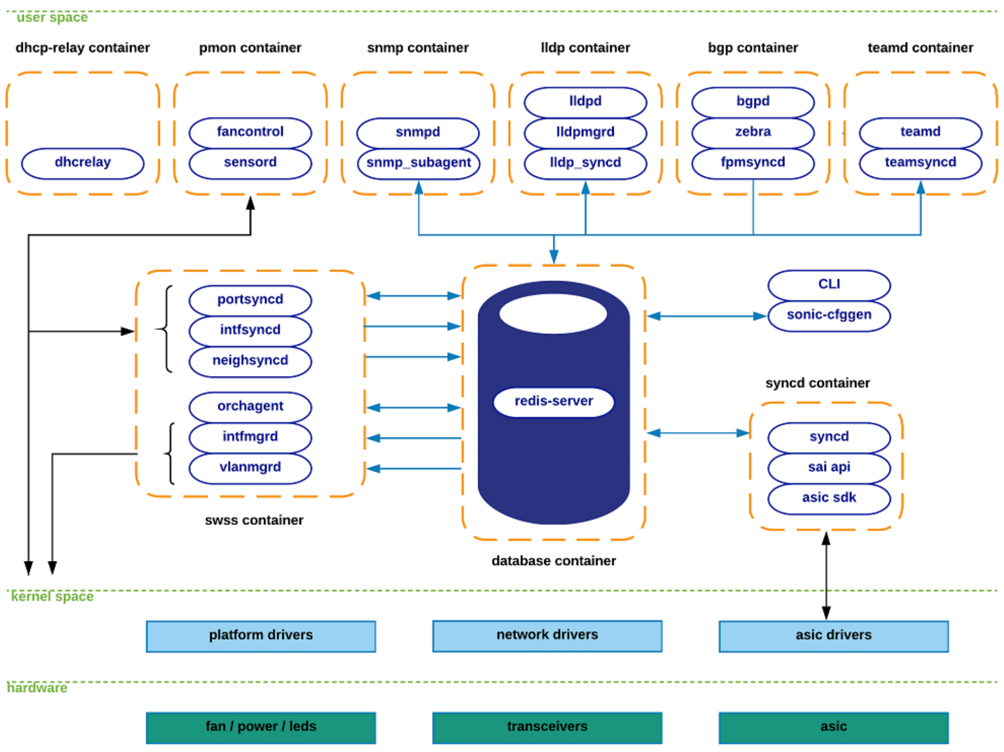


Figure 9-10 SONiC system architecture

Figure 9-10 shows a high-level view of the functionality enclosed within each docker-container, and how these containers interact among them. Some key modules, as the SONiC's configuration module (sonic-cfggen) and the SONiC's CLI, are not implemented as containers but run directly on the linux-host system itself.

Currently, SONiC breaks its main functional components into the following docker containers:

Dhcp-relay container: enables the relay of DHCP requests from a subnet with no DHCP server to one or more DHCP servers on other subnets.

Pmon container: in charge of running *sensord*, a daemon used to periodically log sensor readings from hardware components and to alert when an alarm is signaled. Pmon container also hosts *fancontrol* process to collect fan-related state from the corresponding platform drivers.

Snmp container: hosts snmp features. It includes: i) *Snmpd*: in charge of handling incoming SNMP polls from external network elements; and ii) *Snmp-agent*: feeds *snmpd* with information collected from SONiC databases in the centralized redis-engine.

Lldp container: hosts LLDP functionality. It inclues: i*) Lldp*: this is the process establishing LLDP connections with external peers; ii) *Lldp_syncd*: in charge of uploading LLDP's discovered state to the centralized database; and iii) *Lldpmgr*: provides configuration capabilities to lldp daemon by subscribing to STATE_DB within the Redis database.

Bgp container: runs one of the supported routing-stacks: Quagga or FRR. Even though the container is named after the routing-protocol being used (BGP), in reality, these routing-stacks can run various other protocols (such as OSPF, ISIS, RIP, LDP, etc.).

BGP container functionalities are broken down as follows:

- *bgpd*: regular BGP implementation. Routing state received by means of the protocol are pushed down to the forwarding-plane through the zebra/fpmsyncd interface.
- *zebra*: acts as a traditional IP routing-manager; that is, it provides kernel routing-table updates, interface-lookups and route-redistribution services across different protocols. Zebra also pushes the computed FIB down to both kernel (through netlink interface) and to south-bound components involved in the forwarding process, through Forwarding-Plane-Manager interface (FPM).
- *fpmsyncd*: small daemon in charge of collecting the FIB state generated by zebra and dumping its content into the Application-DB table (APPL_DB) of the redis-engine.

Teamd container: runs Link Aggregation functionality (LAG). It includes: i) *teamd* is a linux-based open-source implementation of LAG protocol; and ii) *teamsyncd* process allows the interaction between *teamd* and south-bound subsystems.

Swss container: the Switch State Service (SwSS) container comprises of a collection of tools to allow an effective communication among all SONiC modules. Swss also hosts the processes in charge of the north-bound interaction with the SONiC application layer with the exception of *fpmsyncd*, *teamsyncd* and *lldp_syncd* processes which run within other containers. The goal of all these processes is to provide the means to allow connectivity between SONiC applications and SONiC's centralized message infrastructure (redis-engine).

- *portsyncd*: pushes port-related state, collected from the hardware-profile config files and by listening to netlink events, into APPL_DB. Attributes such as port-speeds, lanes and MTU are transferred through this channel.
- *intfsyncd*: listens to interface-related netlink events and push collected state into APPL_DB. Attributes such as new/changed ip-addresses associated to an interface are handled by this process.
- *neighsyncd*: Listens to neighbor-related netlink events triggered by newly discovered neighbors as a result of ARP processing. Attributes such as the mac-address and neighbor's address-family are handled by this daemon and transferred to APPL_DB.
- *orchagent*: contains the logic to extract all the relevant state injected by *syncd daemons, processes this information accordingly, and pushes it into the ASIC_DB within the redis-engine.
- *intfmgrd*: reacts to state arriving from APPL_DB, CONFIG_DB and STATE_DB to configure interfaces in the linux kernel.
- *vlanmgrd*: reacts to state arriving from APPL_DB, CONFIG_DB and STATE_DB to configure VLAN-interfaces in the linux kernel.

Database container: hosts the Redis-database engine. Databases held within this engine are accessible to SONiC applications through a UNIX socket. These are the main databases hosted by the Redis engine:

- APPL_DB: stores the state generated by all application containers -- routes, next-hops, neighbors, etc.
- CONFIG_DB: stores the configuration state created by SONiC applications -- port configurations, interfaces, VLANs, etc.
- STATE_DB: stores "key" operational state for entities configured in the system. This state is used to resolve dependencies between different SONiC subsystems.

- ASIC_DB: stores the necessary state to drive ASIC's configuration and operation.
- COUNTERS_DB: stores counters/statistics associated to each port in the system.

Syncd container: its goal is to provide a mechanism to allow the synchronization of the switch's network state with the switch's actual hardware/ASIC. This includes the initialization, the configuration and the collection of the switch's ASIC current status. The main logical components are:

- *syncd*: process in charge of executing the synchronization logic mentioned above. At compilation time, syncd links with the ASIC SDK library provided by the hardware-vendor, and injects state to the ASICs by invoking the interfaces provided for such effect.
- *SAI API*: the Switch Abstraction Interface (SAI) defines the API to provide a vendor-independent way of controlling forwarding elements, such as a switching ASIC, an NPU or a software switch in a uniform manner.
- *ASIC SDK*: hardware vendors are expected to provide a SAI-friendly implementation of the SDK required to drive their ASICs. This implementation is typically provided in the form of a dynamic-linked-library.

9.3 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 9-11, 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.

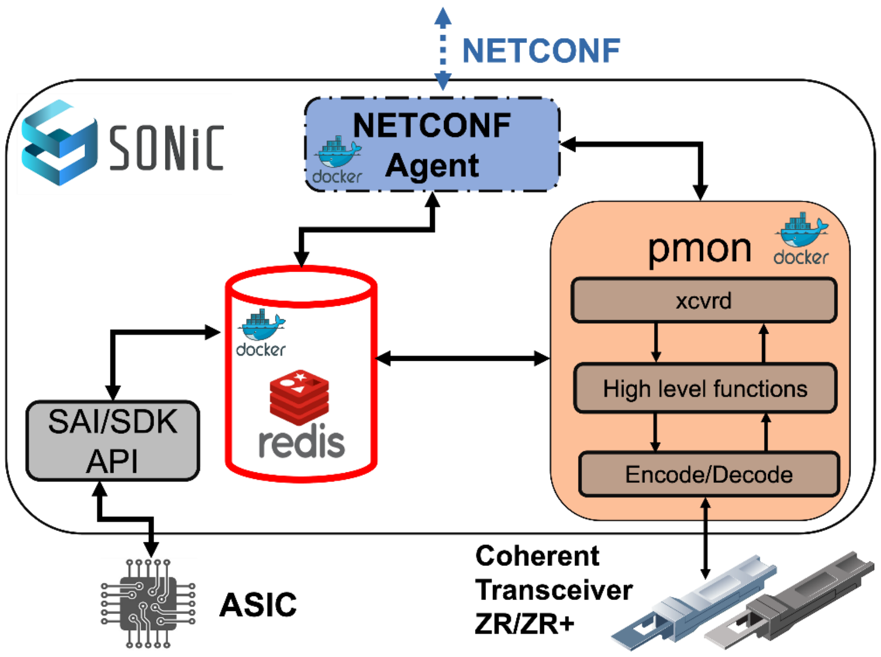


Figure 9-11 Pluggable management and control architecture

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 by the *xcvrd* daemon from the Redis database. 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 9-11 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.

9.4 P2MP Pluggable Management and Control

The management of P2MP pluggable modules proposed by Open XR [OpenXR] considers a dual management structure. The first path, as shown in Figure 9-12 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 9-12 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.

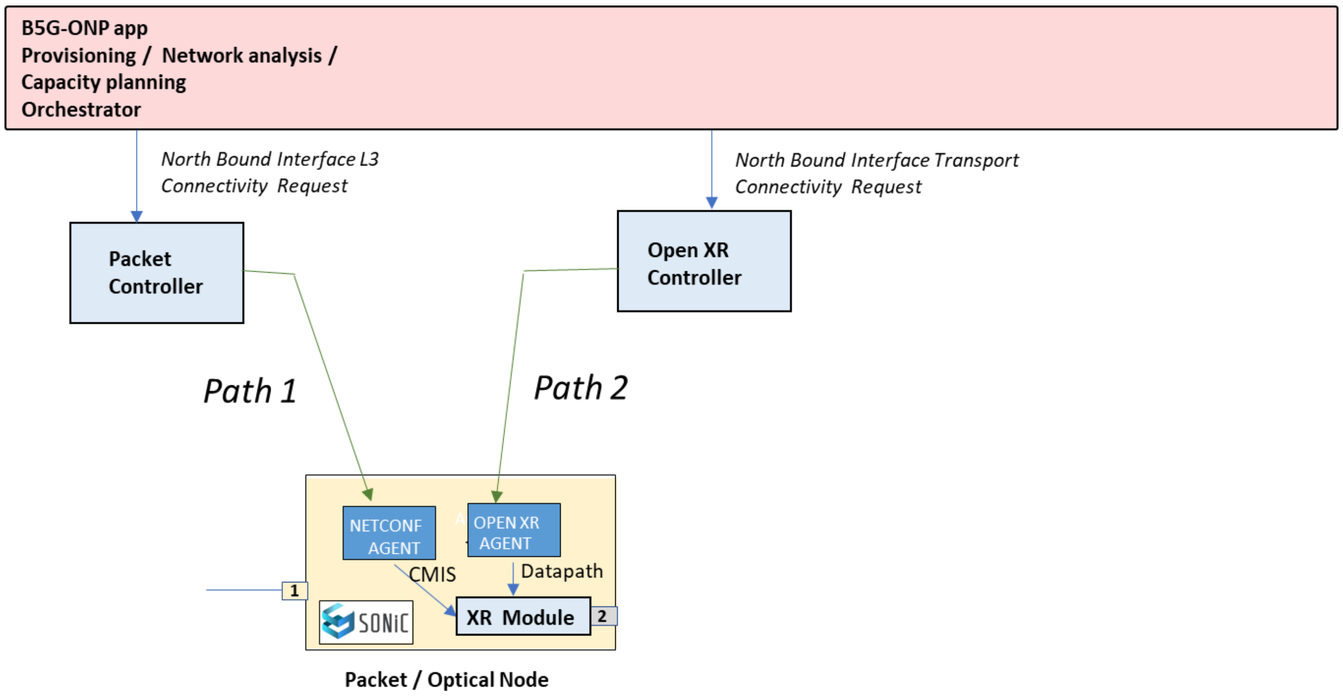


Figure 9-12 P2MP Control integration in B5G-OPEN

10 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.

In the measurements, standards protocols devised for telemetry such as gRPC/gNMI [GRPC22] are rapidly gaining attraction. Although such protocols can reduce the amount of data, there are huge volumes of measurement data to be collected. Additionally, the frequency of data collection leads to make those architectures not practical. In the case of events, they should be transported and distributed to other systems.

In addition, many devices already deployed in transport networks still rely on NETCONF/RESTCONF for their management and configuration. These protocols provide notification mechanisms [RFC 5277], but they were not originally designed to provide a constant flow of telemetry.

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.

10.1 B5G-OPEN Telemetry Architecture

Figure 10-1 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). Note that the SDN architecture might include a hierarchy of controllers, including optical line systems and parent SDN controllers (see Section 5) . 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).

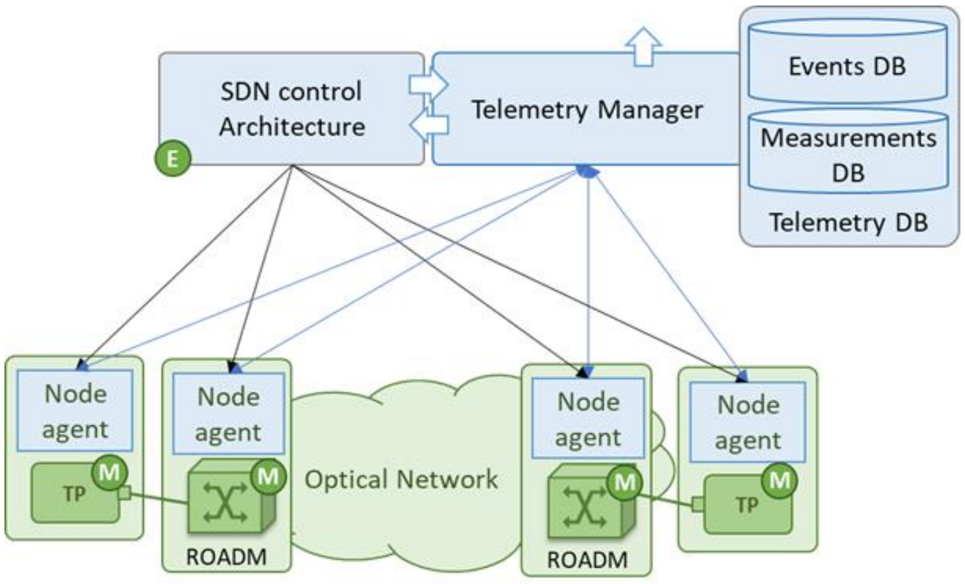


Figure 10-1 Overall network architecture

A detailed architecture of the proposed telemetry system is presented in Figure 10.2 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.

Let's describe a typical measurements telemetry workflow valid for a variety of use cases. The node agent includes modules (denoted as data sources) that gather measurements from the 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 Figure 10.2). The periodicity of the 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 a 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 Figure 10-2) (2), which conveys the data to the telemetry manager. Since gRPC requires a previous definition of the data to be carried, 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 transported data, 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 Measurements 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.

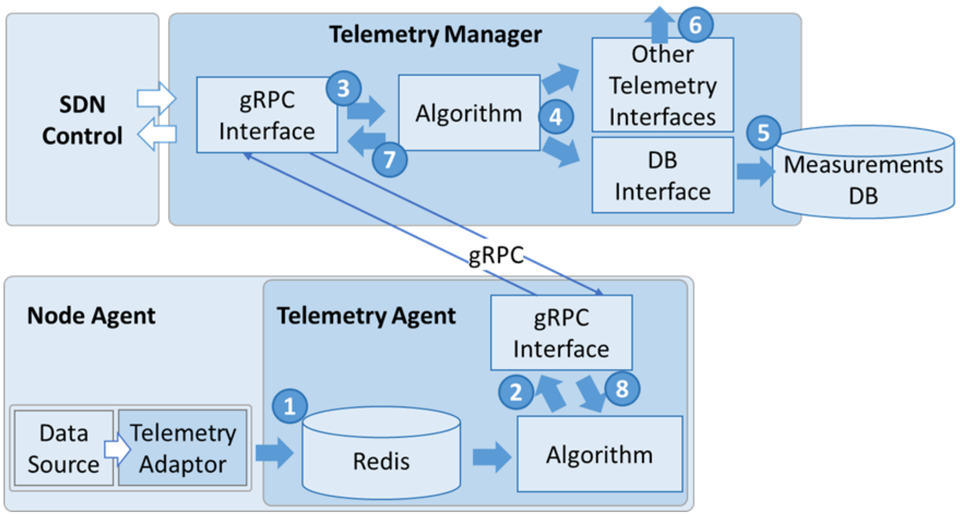


Figure 10-2 Measurements telemetry architecture and workflow

Fig. 10.2. Measurements telemetry architecture and workflow

The architecture for the case of events telemetry is presented in Figure 10-3Events 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.

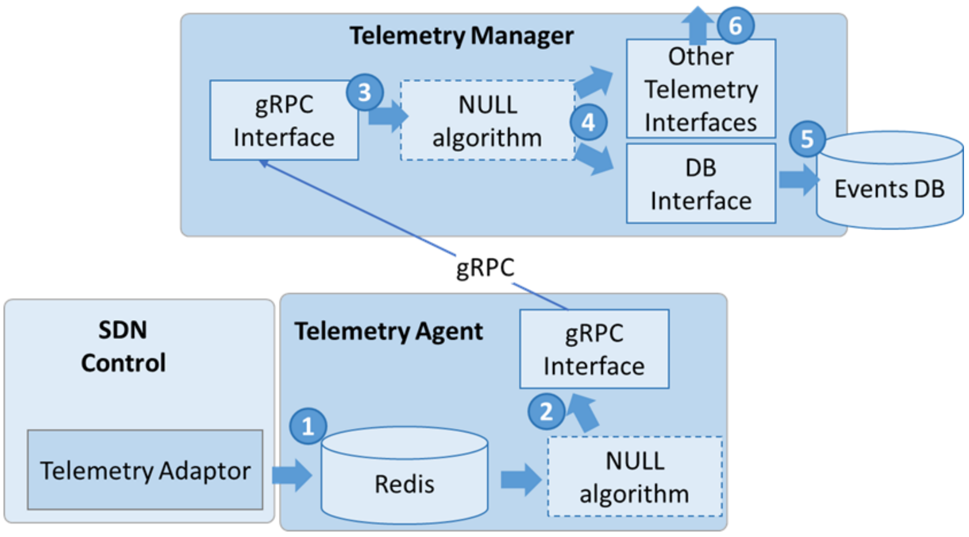


Figure 10-3 Events telemetry architecture and workflow

10.2 Telemetry Data Sources

10.2.1 TAPI Optical Network Orchestrator / SDN controller

In this case, the objective is to use streaming (mechanism that handles the providing of data from one system to another in some form of steady and continuous data flow) for the reporting (notification) of ongoing change of state of the controlled system from one Management-Control entity (TAPI Optical Network Orchestrator) to another (usually superior) management-control entity. Since a significant part of the information is derived from instrumentation the data flow is often called telemetry. A streaming approach is defined that focuses on conveying TAPI entities, i.e., yang sub-trees and allow a client to achieve and maintain eventual consistency with the state of the controlled system.

In this setting, an Event source/server streaming mechanism is made available as an alternative to traditional notifications. The streaming capability is distinct from TAPI Notification and is designed to better deal with scale and to provide an improved operational approach. In this context, any component of the SDN control plane may act as a source of streaming telemetry.

In particular, the TAPI Optical Network Orchestrator SDN controller will act as a data source. For this, the internal architecture of the software will be modified to report asynchronous events that happen in the network Macroscopically, the component will implement a REDIS client following the B5G-OPEN network streaming telemetry architecture and will generate asynchronous events related to topology and connection management. The events that will be notified cover network events, related to:

- Topology (new link, new node, updated node edge point…)
- Connectivity (new service, new connection)

The encoding of such events follows TAPI streaming and Telemetry yang model. For example, the next snippet shows a specific event:

```
{
   "metadata":{
      "measurement":"EventTelemetry",
      "index":"sdn_index"
   },
   "data":{
      "tapi-streaming:log-record":{
         "log-record-body":{
            "event-time-stamp":{
               "primary-time-stamp":"2022-11-02 11:05:25.080535465 UTC"
            },
            "link":{
               "cost-characteristic":[
                  {
                     "cost-name":"te-metric",
                     "cost-value":"1.000000"
                  }
               ],
               "direction":"UNIDIRECTIONAL",
               "layer-protocol-name":[
                  "PHOTONIC_MEDIA"
               ],
               "node-edge-point":[
                    {
                       "node-edge-point-uuid":"89937add-3380-58ab-94ff-9b2fb4efbeeb",
                       "node-uuid":"589df6c1-90e1-51f5-bda4-b4cd6b2d01e4",
                       "topology-uuid":"d8013ae5-12d1-54c0-b653-5d3b5080989f"
                    }
                 ],
                 "uuid":"71505848-d2b3-57dd-8069-295ce111ec61"
              },
              "record-content":"LINK"
           },
           "log-record-header":{
              "entity-key":"71505848-d2b3-57dd-8069-295ce111ec61",
              "log-append-time-stamp":"2022-11-02 11:05:25.080519123 UTC",
              "record-type":"RECORD_TYPE_CREATE_UPDATE",
              "tapi-context":"1d2ba340-41c3-53a9-a615-88380211e6fc",
              "token":"0"
           }
        }
     }
}
```

10.2.2 LiFi Access Points

The LiFi telemetry data will be collected by LiFi access points as illustrated in Figure 11.5-. On the physical layer, the LiFi AP contains photodiode receivers which could capture the uplink optical signals and convert them into electrical signals. Such signals contain information of the link quality and network performance which are the key telemetry data of interest. This information such as received signal strength (RSS) and achieved throughput, can be collected periodically whenever a user is connected. To accommodate with the telemetry manager, a telemetry adaptor will be implemented within the LiFi AP to send telemetry data to the telemetry manager.

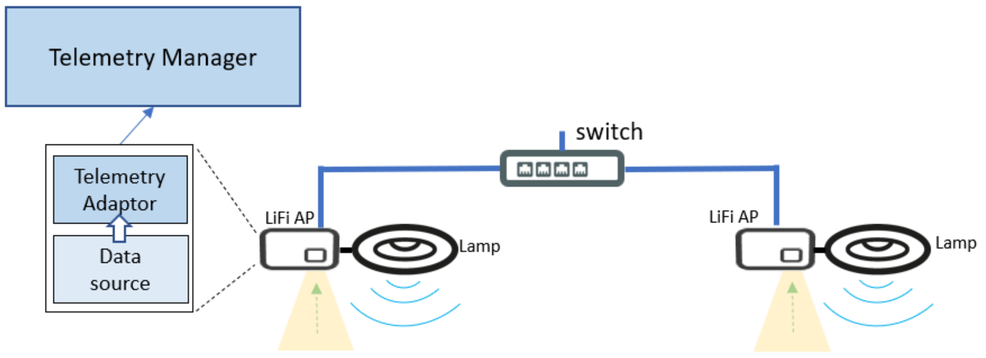


Fig. 11.5 LiFi Telemetry.

10.2.3 Data Collection

“Flex-Telemetry” (see Figure 10-4) is a program that performs periodically requests to collect performance measurements from ADVA devices, using NETCONF and a combination of open (OpenConfig) and proprietary data models. Meanwhile, a modular plugin system provides a NBI interface capable of providing a stable source of stream telemetry to different mediums, such as time-series and in-memory DBs, International Data Spaces (IDS)

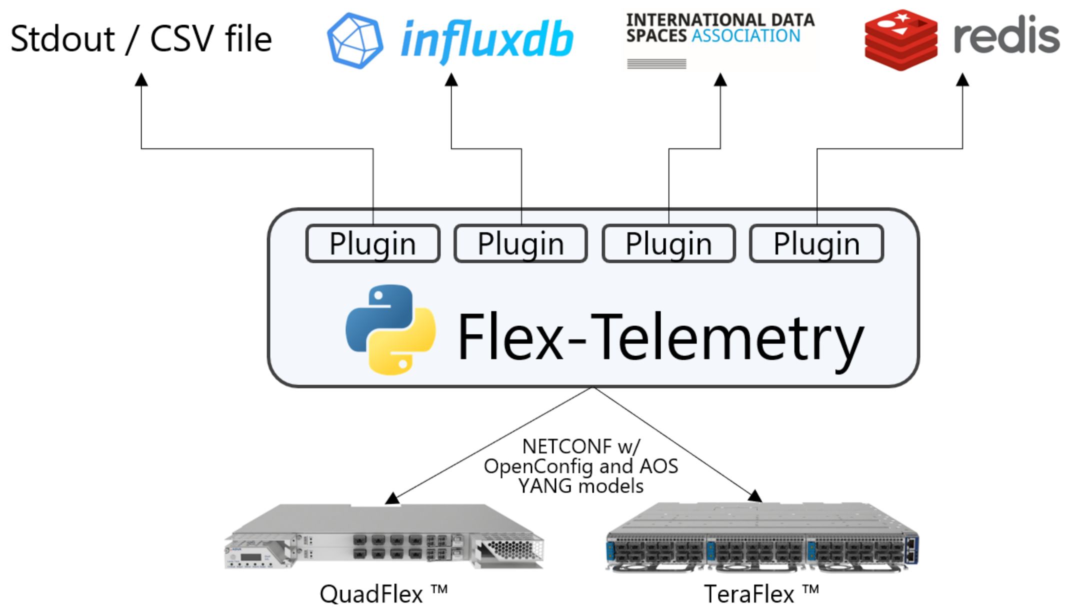


Figure 10-4 ADVA Flex-Telemetry agent

10.2.4 Spectrum monitoring

The Nokia Optical Network testbed combine more than 400 km optical fibre with 7 Nodes (Figure 10-5). Hardware come from 3 different vendors and also integrates offline lab measurement. We have a fully characterized component: fibre length, loss and chromatic dispersion; ROADM losses. Transponder consists in 12 commercial elastic TRX Line and offline TRX. Other channels based on ASE noise are available to load the testbed. The testbed is continuously monitored by our agent.

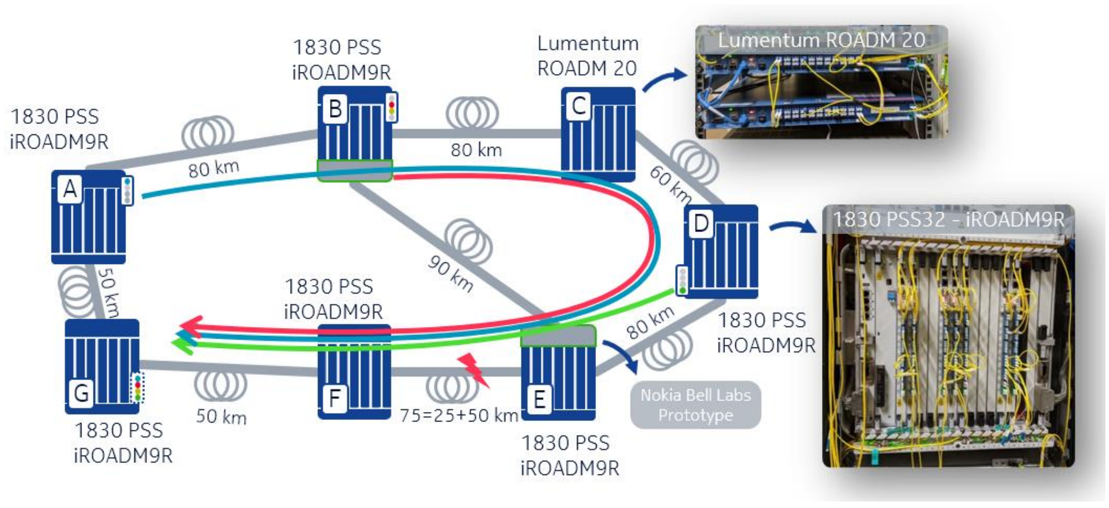


Figure 10-5 Nokia Optical Network testbed

We plan to integrate the spectrum monitoring at each node ingress and egress port to the telemetry system by implementing a telemetry adaptor. The message will be sent to REDIS instance via JSON with the following structure:

```
{
freq : [<array of frequency value in MHz>]
power_x : [<array of power value in mB>],
power_y : [<array of power value in mB>],
channel: [<array of [<central channel frequency MHz>,< channel width in MHz
>]>]
location:<string containing location identifier>
}
```

10.3 Packet flow monitoring using hashing techniques

Monitoring packet streams at line rates equal or above 100 Gb/s is extremely challenging. Sampling techniques are not recommended since a large number of packet flows (above 40%) contain only one packet [Jurkiewicz20]. Also, inspecting all packets and storing all packet headers do not scale as well since FlowIDs require 104 bits in IPv4 or 288 bits in IPv6. For this reason, monitoring streams of packets have to be done using hash-based techniques on all the packets.

There exist different probabilistic data structures that provide accurate summaries (not exact) efficiently in terms of time and memory requirements; these can be used to query streams of packets, for example:

- To test if a packet belongs to a group of flows or not, for instance, a black list (using Bloom Filters)
- To obtain cardinality of flows, i.e., how many different flows are traversing the port (using HyperLogLog algorithm)
- To identify the top-K heaviest flows/heaters, i.e., top 10 heavy-hitters or top-20, etc (using Count-Min Sketch)

These summaries allow to perform specific operations on all flows at the data plane (in a programmable data plane like P4) with reduced memory requirements. Packet-optical boxes will include a telemetry agent which will coordinate with a packet-based telemetry management system for the setup, configuration and retrieving of flow-monitoring information, by means of BF, HLL and CMS data structures. These structures, implmented in the P4 pipeline of the packet-optical boxes will be available through gRPC to the telemetry manager.

10.3.1 Bloom Filters

Bloom Filters allow to check whether a FlowID belongs to a list or not [Bloom70]. A Bloom Filter is an array of M bits initially set to zero. Adding an element to the BF set requires to hash the element using K hash functions and setting the resulting bits to one in the BF. This operation of adding (i.e. setting the results of the K hash functions to one) is repeated for every element in a list S (I.e. S={x,y,z}). BF may have false positives (never false negatives) if, by chance, all k hash functions of w point to bits set to 1 by other elements.

Next, when a packet w arrives at the port, the way to check whether it belongs to the set S stored in the BF is by taking the K hash functions on w and see which bits they point in the BF structure.

- If $h_k(w)=1$ for all k hash functions, then YES
- If $h_k(w)=0$ for some k hash functions, then NO

In general, using 15-20 bits per flow lead to compact BF structures with reduced false positive rates and feasible implementations on P4.

10.3.2 HyperLogLog

HyperLogLog (HLL) allows to estimate the cardinality of a set, that is, the number of different flow IDs traversing a given port [Flajolet07]. In HLL, only one hash function (e.g., murmur32) is needed to be performed to every packet traversing the port. The hash function produces a result with n bits; of these n bits, the first b bits identify a register (there are $m = 2^b$ registers), while the remaining bits are processed in the following way: the number of consecutive zeroes is inspected from bit b+1 onwards. The result is stored in the register pointed by the first b bits, only if the number of consecutive zeroes is larger than the existing value already stored in the register.

In general, using 32-64 registers with 16 bits per register allow to obtain accurate estimates of flow cardinality with compact memory use (about 1 Kbit).

10.3.3 Count-Min Sketch

The Count-Min Sketch (CMS) is a data-structure that allows to store the frequencies of each flowID in a compact manner [Cormode09]. It is very similar to having multiple Counting Bloom Filters. The challenge is again to store the frequencies of flows (especially the heavy hitters) where most of them are unique (around 60% of the flows only have one or two packets).

To do this, the CMS comprises a matrix with d rows (one per hash function) and w columns. When an element arrives, we compute all d hash functions (one per row) and increase by one each position in the appropriate column. Of course, there will be collisions in some positions of the structure. However, taking the minimum value of the d positions will reduce the probability of overestimating the flow.

After all elements are introduced in the CMS, we can query it to get an approximate of frequency for a particular element. This is performed by inspecting all the counters pointed by the hash functions of the flowID and taking the minimum value. In general, with some tens of Kbit of memory, we can accurately estimate cardinalities in the order of thousands of flows.

11 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.

11.1 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. Since the two last decade, the introduction of the digital signal processing at the transponder level leads to low cost and massive monitoring of the physical layer while the software defined network paradigm takes advantage of the NE programmability through standardized interface (via the OpenAgent) to exploit dynamic reconfiguration towards automation.

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]. Additionally, the recent introduction of the P2PT [Pao22] also enables the feedback and the decision of a transponder or a node locally and will be considered as a local control loop even if the Central Telemetry Manager can be a client -a consumer- of the P2PT.
- 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.

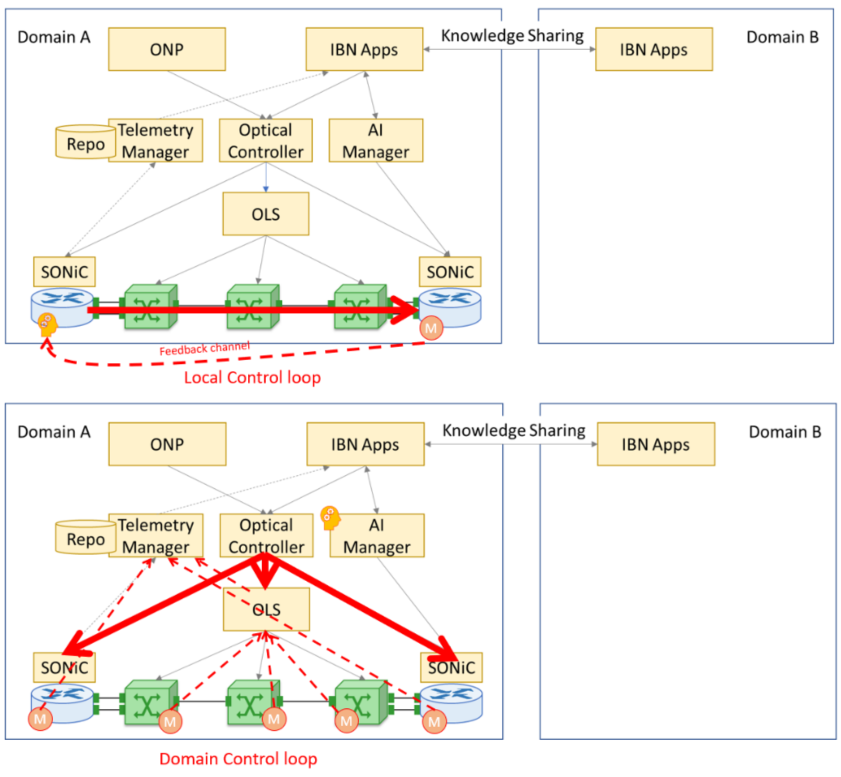


Figure 11-1 Intra domain Control loop architecture

11.2 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 11-2). In this section, we rely on a deep learning-based IBN application for the optical time domain, named OCATA [OCATA], 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 (labelled 1 and 2 Figure 11-2). Such models are made available to IBN applications (3), which use them to generate expected signals that can be compared with those obtained from the network (4). 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.

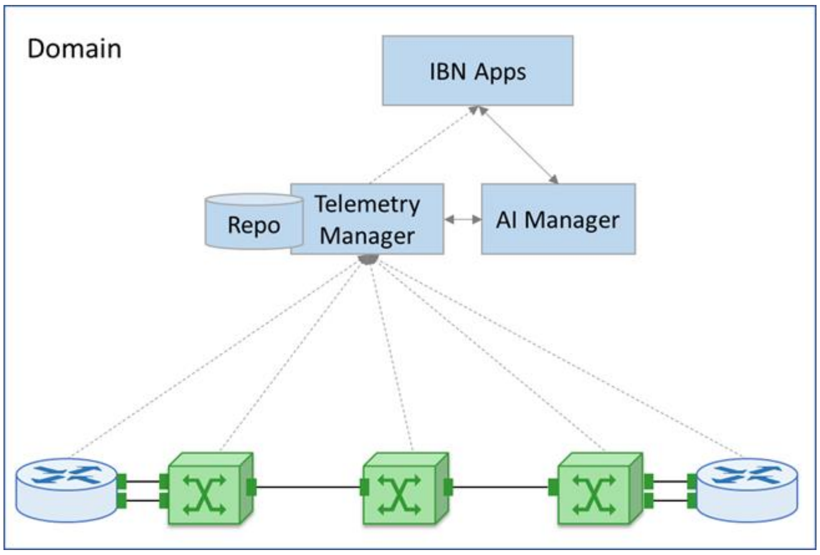


Figure 11-2 Intent-based networking in the intra-domain

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 11-3). 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.

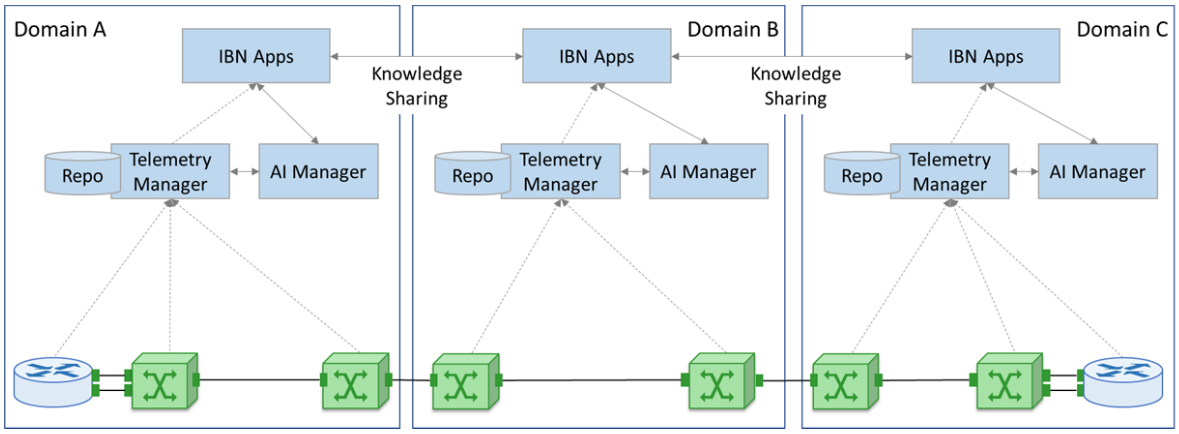


Figure 11-3 Intent-based networking in multi-domain scenarios

12 Initial Considerations on Interfaces and Protocols

This section provides an overview of the main interfaces and protocols that are considered for the interactions between B5G-OPEN control plane components and towards external systems (such as Operational Support Systems) and Network devices (including the prototypes developed by WP3).

12.1 OpenConfig for Packet and Optical

The OpenConfig project [OpenConfig] is a collaborative effort by network operators to develop programmatic interfaces and tools for managing networks in a dynamic, vendor-neutral way. Thus, its models are periodically updated. All OpenConfig models are available on github [OpenConfig]. The OpenConfig information model is composed by a set of abstract modules. Each one is composed by a set of YANG models and represents a specific capability and features of a network device, such as HW components hierarchy, interfaces, OSPF configuration, QoS, among others.

The main modules used for Packet and Optical SDN operations are the following:

Platform:

- Platform - openconfig-platform.yang – It constitutes the main model to define the **hardware components** of a network device.
- CPU - openconfig-platform-cpu.yang – It augments the platform model to add specific parameters of a CPU component.
- FAN - openconfig-platform-fan.yang – It augments the platform model to add specific parameters of a FAN component.
- LINECARD - openconfig-platform-linecard.yang – It augments the platform model to add specific parameters of a Linecard component.
- PORT - openconfig-platform-port.yang – It augments the platform model to add specific parameters of a port component.
- PSU - openconfig-platform-psu.yang – It augments the platform model to add specific parameters of a PSU (Power Suply Unit) component.
- TRANSCEIVER - openconfig-platform-transceiver.yang – It augments the platform model to add specific parameters of a Transceiver component.
- Platform Types - openconfig-platform-types.yang – It defines the types used to define the parameters in the platform module parameters

Optical-Transport:

- Terminal Device - openconfig-terminal-device.yang – It defines the main model to define a terminal optics device.
- Optical Transport Types - openconfig-transport-types.yang – It defines the types used to define the parameters in the optical transport module parameters.

Terminal device manifest

Openconfig has defined the manifest files, a special type of model which is not configuration nor operation. A remote controller requires some data from the transceiver in order to perform optical planning and impairment validation of the end-to-end transmission across an Optical Line System (OLS). When a pluggable module is recognized by a terminal device (which can be a transponder or a packet-optical box), the operational mode datastore is updated.

- operational-mode-capabilities this set of attributes contains all characteristic information of the signal (modulation format, FEC, bit rate...), relevant information for the physical impairment validation (OSNR Rx sensitivity, CD/PMD tolerance and penalties).
- optical-channel-config-value-constrains: Contains the transmission configuration constrains/ranges of the optical-channel's attributes characterized by the operational-mode, i.e., the central frequency range, the frequency grid and the configurable transmitted power.

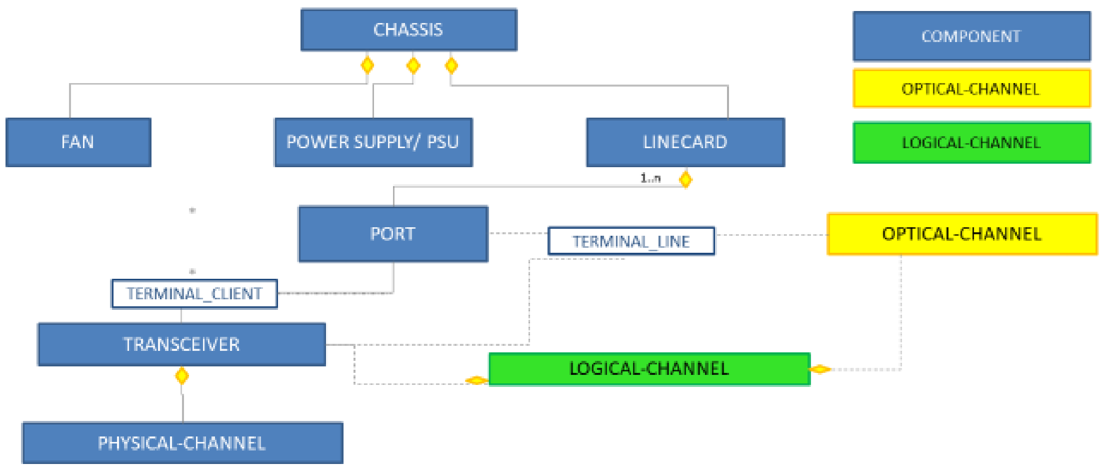


Figure 12-1 Hierarchy of components of an open terminal device

The main modules that are required for packet (IP/MPLS) control are:

bgp: This set of modules describe the BGP protocol configuration. They are used in the service related use cases to handle the BGP protocol and to support IP Connetivity.

interfaces: Model for managing network interfaces and subinterfaces. For the use cases that are currently defined is used to configure the line side interfaces after setting up the optical connectivity.

local-routing: This module describes configuration and operational state data for routes that are locally generated, i.e., not created by dynamic routing protocols. It can be used with network-instances to configure the static routes.

network-instance: The network instance is an abstraction of a packet fowardign device. It may be a Layer 3 forwarding construct such as a virtual routing and forwarding (VRF) instance or the Global routing instance. A Layer 2 instance such as a virtual switch instance (VSI) and Mixed Layer 2 and Layer 3 instances are also supported. The network instance works in conjunction with other modules such as:

o Interfaces

o VLANs

o Potocols

12.2 P4 AND P4 RUNTIME

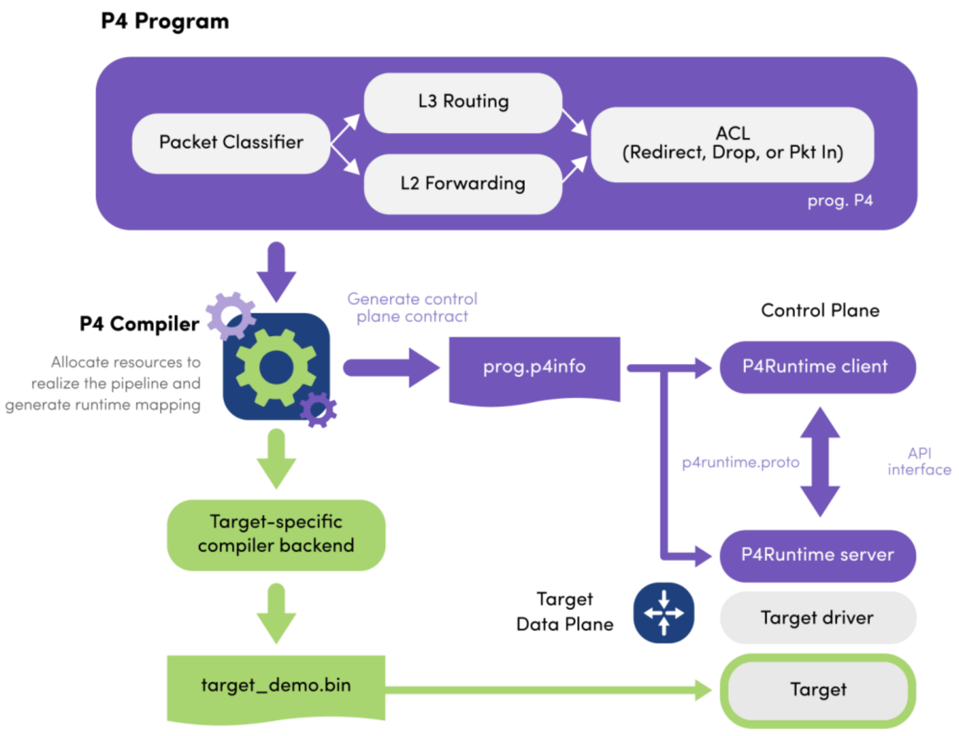


Figure 12-2 P4 development workflow [https://p4.org/]

Programming Protocol-independent Packet Processors (P4) is a domain-specific language for network devices that defines how data plane devices (switches, NICs, routers, filters, etc.) process packets. P4 [P4] is an open-source project whose goal is to develop and define the tools needed to work with P4 (e.g., specifications, compiler, interfaces, etc.) in order to enable next-generation SDN. The tools/applications developed in the project are maintained in [P4lang] GitHub repository. Figure 12-2 shows the P4 development workflow required to program and install a P4 pipeline and control it via an SDN controller. More in detail, a P4 program allows to implement a custom pipeline supporting: configurable match-action tables and packet headers, metadata extraction, programmable actions, and stateful data structures. The P4 compiler generates an executable file for the target data plane and the runtime mapping metadata to allow the communication among control and data planes. The P4Runtime API is an RPC interface used by the control plane for managing a P4 device where a custom pipeline is installed. After a detailed analysis, P4 language and P4Runtime perfectly fit the requirements for managing and control the packet optical node in the BG5-OPEN project. Indeed, the possibility to perform in-network operations opens the way to new applications and functionalities to be operated at wire-speed. The B5G-OPEN deliverable D3.1 reports an example of optical monitoring parameters included within telemetry packets that are processed by the P4 ASIC for fast recovery.

12.3 Transport API (TAPI)

The TAPI Optical Network Orchestrator, as an SDN controller, provides Network Topology and Connectivity Request services to a parent SDN Controller or another T-API-able user. It is mainly responsible for the offering of DSR connectivity services between optical transponders that are connected to the ROADMs. The transport protocol used for all operations on the NBI is RESTCONF [RFC8040]. It is an HTTP-based protocol that provides a programmatic interface for accessing data defined in YANG, which is the language T-API is defined in. The key YANG models composing the T-API information models are to be based either in the current version 2.1.3 [TR-547] or in the upcoming TAPI 2.4 [TAPI2.4], including the following modules

- *tapi-common.yang ,*
- *tapi-connectivity.yang ,*
- *tapi-dsr.yang*
- *tapi-topology.yang*
- *tapi-connectivity.yang*
- *tapi-path-computation.yang*

12.3.1 Generic Aspects

T-API is based on a context relationship between a server and a client. A Context is an abstraction that allows for logical isolation and grouping of network resource abstractions for specific purposes/applications and/or information exchange with its users/clients over an interface. It is understood that the APIs are executed within a shared Context between the API provider and its client application.

A shared Context models everything that exists in an API provider to support a given API client. The T-API server *tapi-common:context* includes the following information: The set of Service Interface Points (SIP) exposed to the TAPI client applications representing the available customer-facing access points for requesting network services.

This set must allow Connectivity Service (CS) creation at the DSR Layer, a topology-context which includes one or more top-level Topology objects which are dynamic representations of the network, and connectivity-context which includes the list of Connectivity-Service and Connection objects created within the TAPI Context.

Adopting TAPI allows a standard and mature way to interact with SDN controllers for optical networks, as specified in OOPT MUST [MUST]. In particular, the figure below (Figure 12-3) shows a common representation of an optical network using TAPI terminology and convention.

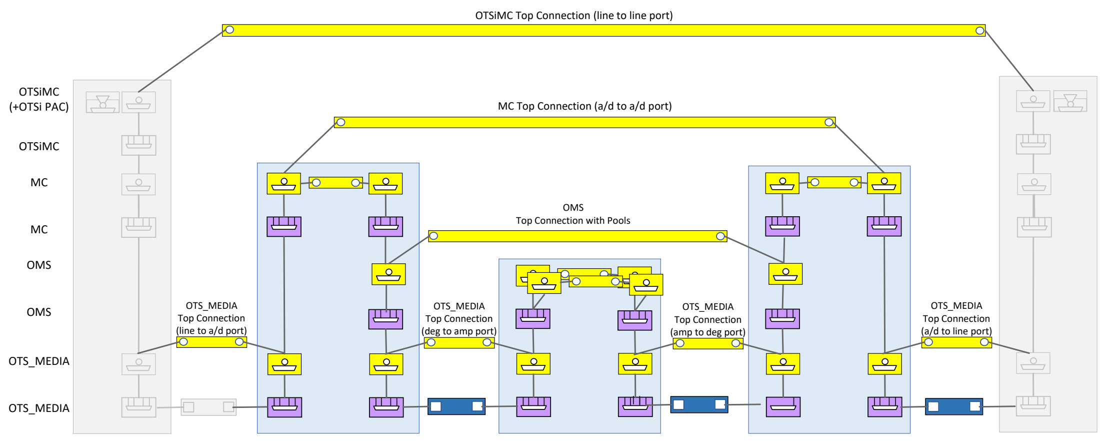
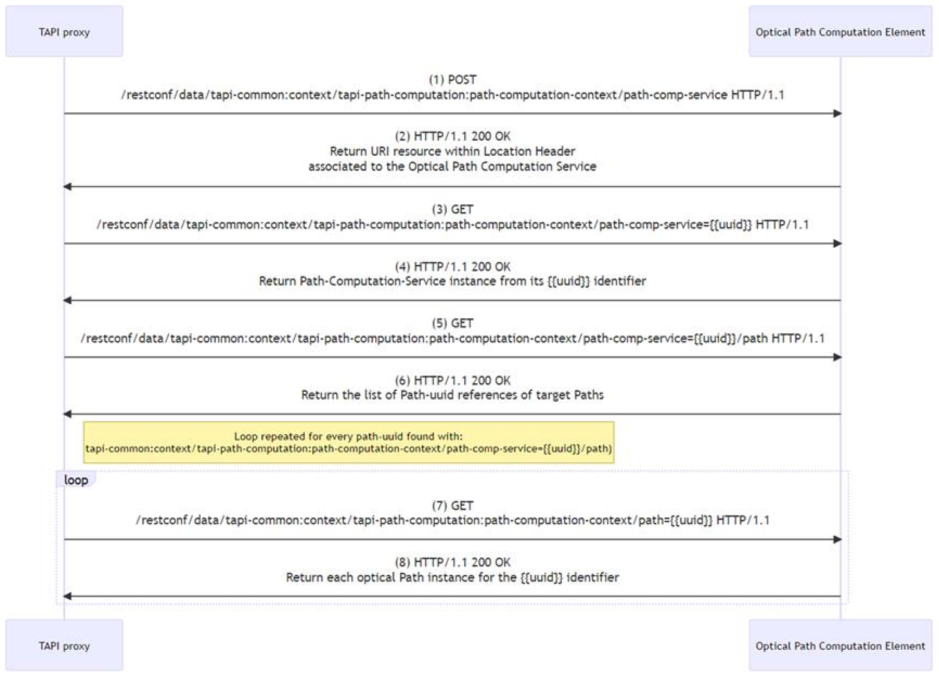


Figure 12-3 TAPI representation of a digital service between pluggables across an optical network

12.4 PATH COMPUTATION

B5G OPEN will adopt the TAPI (Transport API) architecture [TR-547], and as such, the OPCE element is a module that can assist the TAPI Optical Network Orchestrator for computing the optical path e.g., in the provisioning process. More than one OPCE implementations can be used, by different partners, e.g., implementations included inside the B5G-ONP, or external to it, with different capabilities.

The interaction between the OPCE and the TAPI Optical Network Orchestrator will be engineered according to the TAPI standards (TAPI version 2) in section 12.3. Figure 12-4 shows an exemplified sequence of messages in a typical interaction.


TAPI proxy
Optical Path Computation Element
loop
TAPI proxy
Optical Path Computation Element

(1) POST
/restconf/data/tapi-common:context/tapi-path-computation:path-computation-context/path-comp-service HTTP/1.1

(2) HTTP/1.1 200 OK
Return URI resource within Location Header
associated to the Optical Path Computation Service

(3) GET
/restconf/data/tapi-common:context/tapi-path-computation:path-computation-context/path-comp-service={{uuid}} HTTP/1.1

(4) HTTP/1.1 200 OK
Return Path-Computation-Service instance from its {{uuid}} identifier

(5) GET
/restconf/data/tapi-common:context/tapi-path-computation:path-computation-context/path-comp-service={{uuid}}/path HTTP/1.1

(6) HTTP/1.1 200 OK
Return the list of Path-uuid references of target Paths

Loop repeated for every path-uuid found with:
tapi-common:context/tapi-path-computation:path-computation-context/path-comp-service={{uuid}}/path)

(7) GET
/restconf/data/tapi-common:context/tapi-path-computation:path-computation-context/path={{uuid}} HTTP/1.1

(8) HTTP/1.1 200 OK
Return each optical Path instance for the {{uuid}} identifier

Figure 12-4 Exemplified TAPI Optical Network Orchestrator -.OPCE interaction.

The project may require extensions of these interactions for accommodating two key novel aspects:

- Multiband operation
- Optical impairment computations in this context.

The details of such variations are being discussed along the WP3/WP4 activities in the project, and will be reported in the appropriate deliverables. Additionally, they will be contributed to the optical scientific community and standards as one of the project outcomes.

12.5 ONOS NATIVE

The ONOS controller includes a wide set of northbound REST APIs providing GET/POST/DELETE methods towards the network [ONOSREST]. For example, GET methods can be used to retrieve information about the network topology or about the current configuration of ONOS applications (NetApps) running on the controller. Similarly, POST and DELETE methods can be used to interact with the network devices and the network applications, e.g., sending new configuration to the network devices or modifying actual values of NetApps parameters.

These interfaces will be used to integrate the optical controller within the B5G-OPEN control plane, specifically the interfaces will be consumed by the Orchestrator and the TAPI tools.

Current implementation of such interfaces is not complete for the B5G-OPEN purposes, especially interfaces should be extended for enabling retrieval of optical resource utilization (e.g., supported and available frequency slots) and optical physical parameters to be used by upper control plane components. Such APIs will be therefore extended during the project to provide all data required by the upper B5G-OPEN components.

12.6 OPENROADM

[OpenROADM] is a Multi-Source Agreement initiative, active since 2015 and comprising several network operators and optical system and component vendors. From the control plane perspective, OpenROADM defines data models for device, network, and service modelling, targeting the fully disaggregated network model. The device model covers detailed configuration information, alarm, and performance monitoring and, as such, was chosen by the METRO-HAUL project as the reference interface for ROADM devices. Recently, device models have been extended opening to the “partial disaggregated” solution covering also trans-, mux- and switch-ponders.

The METRO-HAUL project developed an OpenROADM driver for the ONOS SDN controller to control ROADM devices. The driver is currently downloadable from the official ONOS repository and available under the ODTN-driver section [ONOS]. During device discovery, ONOS retrieves the number and type of ports together with their capabilities to feed its internal device database. More specifically, the current driver collects the spectral feature of the ports reading the <mc-capabilities> branch from the device datastore, available both for degrees and Shared Risk Group (SRG, i.e., add/drop modules). However, with such spectral information is possible to model only single band devices. Recent updates of the device model (starting from v.7.0.0) address multi-band devices by a new top-level branch named <mc-capabilities-profile>, very similar to the old <mc-capability> but that can be instantiated several times to describe the different bands and can be referenced by ports, degree and SRG, as can be seen in the following tree, extracted from the OpenROADM device model.

```
+--rw circuit-packs* [circuit-pack-name]
|   +--rw circuit-pack-type
|   +--
....
|   +--rw ports* [port-name]
|      +--rw port-name
|      +--
|      +--ro mc-capability-profile-name*
....
+--rw degree* [degree-number]
|   +--rw degree-number
|   +--
|   +--ro mc-capability-profile-name*
....
+--rw shared-risk-group* [srg-number]
|   +--rw srg-number
|   +--
|   +--ro mc-capability-profile-name*
....
+--ro mc-capability-profile* [profile-name]
    |   +--ro profile-name
    |   +--ro center-freq-granularity?
    |   +--ro min-edge-freq?
    |   +--ro max-edge-freq?
    |   +--ro slot-width-granularity?
    |   +--ro min-slots?
    |   +--ro max-slots?
```

Figure 12-5 Extract of OpenROADM tree for multiband support

On the other hand, the device model covers only the case of point-to-point connections. Coverage for point-to-multipoint optical connections, as those needed by XR-optics, is not yet in the scope of the MSA. Currently, the <roadm-connection> container allows creation of both express and add-drop connections uniquely between one source and one destination Network Media Channel (NMC) interface. Even if nothing prevents, from a generic perspective, to have more than one roadm-connection referencing the same NMC, this is not a shared solution and may create interoperability issues.

12.7 Interfaces for the Telemetry Platform

We rely on Redis as a demarcation point between data sources and the telemetry system. Telemetry data generated by data sources are encapsulated as JSON objects and published in a Redis database. In particular, the RedisJSON module that provides JSON support for Redis is used. RedisJSON lets store, update, and retrieve JSON values in a Redis database, similar to any other Redis data type. The publish/subscribe paradigm is used to decouple data sources (publishers) from subscribers aimed at reaching good scalability and achieving dynamic messaging routing.

A gRPC interface is used to transport telemetry data, both measurements and events, between telemetry agents and the telemetry manager. GRPC uses Protocol Buffers as the interface description language to serialize structured data. Protocol buffers have a strict specification, which needs to be defined for every type of data being transported. Then, to avoid the proliferation of protocol buffers schema, a single one is defined that encoded in base64 JSON objects encapsulating whatever telemetry data to be transported. Note that B5G-OPEN relies on intelligent data aggregation techniques to reduce the amount of telemetry data that is actually conveyed to the centralized telemetry manager. Therefore, even though gRPC potentials are not fully exploited in this approach, preliminary results show no significant increment of data being transported.

12.8 Control of pluggable modules

12.8.1 Coherent pluggable modules

Common Management Interface Specification (CMIS)[CMIS], defines a generic management communication interface and protocol among the host (e.g., network switch) and modules (e.g., optical transceivers). This interface has been defined in order to provide a standard across a variety of module capabilities and form factors (QSFP-DD, OSFP, COBO) to foster the vendor agnostic management. The Coherent CMIS (C-CMIS) extends the CMIS interface to handle coherent optics pluggables, e.g., 400ZR modules, that require additional calls to perform interface-related data processing, such as Forward Error Correction (FEC). The CMIS and C-CMIS specifications are defined within the Optical Internetworking Forum (OIF)[OIF] thanks to the joint collaboration of network devices vendors. These standards fulfil the needs of B5G-OPEN project for managing ZR and OpenXr pluggable modules within a packet-optical node.

12.8.2 PON pluggable modules

The integration of the PON access networks (e.g., PON pluggables) with the B5G-OPEN software platform can be realized in three different levels (from higher to lower layers):

a) B5G-OPEN integration with PON Manager;

b) B5G-OPEN integration with PON Controller;

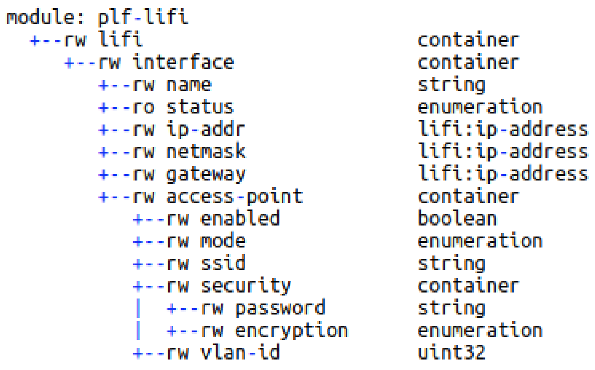
c) Direct communication using OLT PON SDK or CLI.

The aforementioned options generate a set of four architectural alternatives for the PON control and therefore for the actual integration with the B5G-OPEN platform, which are described in detail in Section 8.1.

The third alternative (direct communication with the pluggables through the B5G-ONP app) seems more straightforward and seems to be the dominant option at the time of the writing of this deliverable.

12.9 LiFi Integration

The LiFi AP architecture has been described and shown in Figure 7-5. The interface for the LiFi integration would be via Ethernet ports. The support for NETCONF will be provided as well as a YANG model for LiFi. A basic model example for LiFi has been implemented as shown in Figure 12-6. It will also be further considered using Openconfig.



```
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
```

Figure 12-6: Basic model for LiFi

It will be enhanced for more control operations such as:

- Enable *lifictl* operations from Netconf i.e., add options in the model to set lamp power, set brightness, display help message, etc.
- Enable system reboot from Yang model
- Add DHCP IP configuration option

12.10 B5G-OPEN Slicing North Bound Interface

The B5G-OPEN project will provide ad-hoc developed APIs using REST/APIs paradigm, following best practices (e.g., Open-API documentation), as the northest API of the ecosystem. The target user of these APIs in an industrial deployment would be e.g., the operators OSS systems, or directly operator personnel in charge on provisioning of the different services.

These APIs are designed to provide an open and programmatic access to the different use cases to be demonstrated. The details of these APIs will be defined later in the project, appropriately reported.

General policies have been already defined for those APIs:

- REST-based APIs, exposed via open documentation frameworks, preferably OpenAPI.
    - Making use of the IETF YANG model for Network Slices [NSv16].
- Adoption of the Optimization-as-a-Service (OaaS) paradigm [Gar19][Pav15]:
    - This relies in the concept of *algorithm repository,* as a set of algorithms exposed, browsable in a catalogue, and runnable via the open APIs. A subset of the algorithms developed along B5G-OPEN will be integrated using this form.
    - Utilization of container models for shipping the algorithm implementations. This means that the different algorithms will be packaged into separated containers, that can be ran independently in the B5G-ONP system. This is an enabler for integrating algorithms developed in different languages and platforms (a practical aspect, that becomes actually a booster for the OaaS concept), and possibly by third parties, into the B5G-ONP system.

12.11 Kubernetes

The integration of the Kubernetes cluster with the B5G-OPEN software platform will be realised through the APIs [K8sAPI] exposed by the Kubernetes API Server (kube-apiserver) deployed in the Kubernetes master. An API client to be deployed in the B5G-ONP app will be used in order to instantiate/delete new services, to retrieve basic resource related statistics and to influence the placement decisions.

A preliminary implementation of the B5G-ONP interaction with a set of (two) K8S clusters is already in place in ELIG facilities. This is being used to prototype the APIs that interact with the K8s for the following use cases, at this moment:

- Registration of new K8s clusters in the B5G-OPEN system. By doing that, the clusters are eligible for dimensioning/provisioning use cases.
- Retrieval of macroscopic occupation information on registered clusters.

It is expected that along year two, the bulk of the control related use cases are also prototyped and demonstrated.

12.12 Open XR control interface

The integration of OpenXR IPM with B5G-OPEN system will be realized on an Open XR NBI. This interface enables:

- Retrieval of module inventory information: part numbers, serial numbers, line and client interfaces;
- Monitoring of fault conditions of the module resources: fault management, following RFC-8632;
- Module software management: inventory and upgrade of SW versions;
- Hub-Spoke topology management: discovery and configuration of optical P2MP network topologies;
- Line side transport capacity management: discovery and configuration of digital sub carriers into transport capacities;
- Client service management: discovery and configuration of client mapping to transport capacities;
- VTI (L2) service management: configuration of VLAN based end to end services. Symmetric or asymmetric traffic with dedicated or shared downlink transport capacity.

A first integration will make use of an OpenXR REST API. Openconfig shall later be extended to allow control of the novelties introduced by XR Optics allowing the integration to be done via Openconfig.

13 Conclusions and Next Steps

This deliverable has presented the set of requirements, use cases and initial architecture proposal for the B5G-OPEN control plane. The set of requirements include the **discovery** of the existing resources and topology, **planning** of the network resources and advanced use cases to support **fault management,** along with use cases for **autonomous network operations** are also described. Control-plane services, including **point-to-point** optical connectivity, **point-to-multipoint** XR connectivity, **IP link** provisioning, **B5G-OPEN Slice** and **telemetry** services, are also described in the document.

The state of art in existing frameworks has been surveyed and presented in Section 4, covering SDN controllers, Node Operating Systems, Telemetry tools and Network planning frameworks. The decisions to build every prototype component of the B5G-OPEN architecture is based on the outcome of the survey. The maturity and availability of open-source code is the main driver to select the framework to build upon.

Regarding the B5G-OPEN **control plane architecture**, this is described in Figure 5-1 showing a number of components that will trigger innovations in the following aspects of the control plane:

- Systems and devices to enable multi-band optical transmission, and their appropriate control, including the control of emerging pluggable interfaces together with existing transmission systems.
- Control tools for setting up multi-domain connections traversing different network segments, including the modelling of physical layer impairments in MB capable networks.
- Planning tools able to understand the packet-optical multi-band network as a whole, including access together with optical and packet layers.
- Telemetry modules for the collection of network state and continuous monitoring of both optical and packet parameters.
- Intent Based Networking, network automation and the design of AI/ML algorithms that enable network self-management and operations based on telemetry information and past historical data.

Sections 6 through 11 have gone in detail in the internal architecture of the components of each architectural Block. In the detailed architecture, the frameworks that have already been selected have been added in the required architectural component, for example SONIC as Node Operating System for packet/optical boxes or ONOS as Optical SDN Controller. It is relevant to highlight that B5G-OPEN architecture is generic and defines Open interfaces among the components.

In terms of Control Plane architecture definition, the work for the second year will be focused on extending the work to multi-controller / segments in a domain-less manner. B5G-OPEN envisions that the delivered services are not constrained to source/destination being in the same segment.

During the first year of the project, a set of interfaces, summarized in Section 12, has been identified to fulfil the control plane requirements for packet, optical and access technologies, as well as communication within B5G-OPEN components and towards external users/applications. The plan for the second year of the project is to specify in detail those interfaces. When required, the available standards/ APIs will be extended and the new specification will be taken back to the relevant body, with the collaboration of WP6. For example, WP4 members are active contributors to Transport API, Openconfig, OpenRoadm or OpenXR, among others. The selected interfaces will be implemented in the foreseen B5G-OPEN Prototypes developed in WP4.

The second year of the project will be focused, in addition, to completing the interface specification, on prototyping the identified WP4 components. These prototypes of software components in development by WP4 will be ready to be used in WP5 demonstrations and will control selected WP3 hardware. It is expected that WP3 will provide the final specification of devices and interfaces and final version of the physical layer impairment validation model to be used in the path computation function.

In terms of **detailed prototyping activities**, the following work is expected for **Y2:**

- Implementation of different B5G-ONP modules, starting with a first prototype of the interface with i) the TAPI Orchestrator and ii) the K8S clusters, according to the guidelines stated in this deliverable.
- Implementation of an interface between a SONIC whitebox and the B5G-ONP in different control use cases.
- Implementation and enhancement for the LiFi YANG model. Meanwhile, the telemetry adaptor is under development for LiFi access which delivers LiFi telemetry data that has been defined.
- Design and implementation of the TAPI enabled Network Orchestrator, which will interact with the B5G-ONP path computation module and with the optical controller.
- Implementation of network streaming module, by which the TAPI orchestrator shall be able to report the status of the network following the B5G-OPEN telemetry architecture as presented in this document.
- **E**xtending and testing of the SONiC NOS to enable the control of coherent pluggables and implement the interfaces toward other B5G control plane blocks. In particular, a demonstration for OFC 2023 (accepted) is in preparation implementing the SONiC box interface toward the B5G-ONP. Moreover, the SONiC box interface toward the ONOS optical controller is in phase of development and will be tested during Y2.
- Deployment and testing the ONOS optical controller. After the first development, during Y2 several developments are needed in ONOS: support of multi-band; support of multi-domain intents; interfaces toward T-API proxy on north-bound; drivers on south-bound toward packet-optical nodes, OpenROADM devices, OpenConfig devices, T-API OLSs, other devices; retrievment of physical impairments from devices and OLSs.
- **I**ntegration of data sources from devices and controllers provided by several partners into the telemetry system and preparing a demonstration for OFC 2023. Intelligent data aggregation algorithms will be developed to process telemetry data distributely.
- Regarding IBN, development of strategies for sharing models between network domains, so as to allow end-to-end modelling.

- Integration and demonstrate the Flex-Telemetry agent in the B5G-OPEN telemetry framework.
- Extension and integration of the OLS control for the control and monitoring of multi-band amplifiers. A current topic is to evaluate the upgrade of the B6G-OPEN Optical Controller and OLS Controller to the recently released TAPI 2.4.0.
- Implementation of packet flow monitoring based on hashing techniques using P4. The monitoring algorithms will be evaluated both in simulation environments, along with a proof-of-concept testbed based on P4 whitebox.
- Integration of the P2MP transceivers with the OpenXR REST API. Openconfig shall later be extended to allow control of the novelties introduced by XR Optics allowing the integration to be done via Openconfig.

14 REFERENCES

[5GCH] 5G-Crosshaul: The 5G Integrated fronthaul/backhaul | 5G-Crosshaul Project | Grant agreement ID: 671598 | European Commission. Retrieved October 5, 2022, from https://cordis.europa.eu/project/id/671598

[BBF-GIT1] Broadband Forum YANG models on GitHub: https://github.com/BroadbandForum/yang

[BBF-TR385] Broadband Forum TR-385, "ITU-T PON YANG Modules", October 2020

[Blo70] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", Communications of the ACM, 13 (7): 422–426, 1970

[Chi17] A. Chifor [Online]. Retrieved October 5, 2022. "Container Orchestration with Kubernetes: An Overview". https://medium.com/onfido-tech/container-orchestration-with-kubernetes-an-overview-da1d39ff2f91

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

[Cor09] G. Cormode, "Count-min sketch" (PDF). Encyclopedia of Database Systems. Springer. pp. 511–516, 2009

[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)

[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

[ELASTICSEARCH] [Online] https://www.elastic.co/es/

[ETSI19] ETSI. GR NFV-IFA 029 - V3.3.1 - Network Functions Virtualisation (NFV) Release 3; Architecture; Report on the Enhancements of the NFV architecture towards "Cloud-native" and "PaaS.", 2019 https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[Fer20b] 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

[Fla07] P. Flajolet et al, "Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm", Discrete Mathematics and Theoretical Computer Science Proceedings, pp. 137–156, 2007

[Gar19] M. Garrich, C. S. N. Martinez, F. J. M. Muro, M. V. B. Delgado, and P. P. Marino, "Network Optimization as a Service with Net2Plan," 2019 European Conference on Networks and Communications, EuCNC 2019, pp. 443–447, Jun. 2019, doi: 10.1109/EUCNC.2019.8802041.

[Gar20] M. Garrich et al., “IT and Multi-layer Online Resource Allocation and Offline Planning in Metropolitan Networks”. Journal of Lightwave Technology, 38(12), 2020, pp. 3190–3199. https://doi.org/10.1109/JLT.2020.2990066

[Gio20] A. Giorgetti et al., "Control of open and disaggregated transport networks using the Open Network Operating System (ONOS) [Invited]", JOCN 2020

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

[Gon22] O. Gonzalez de Dios et al, "MANTRA Whitepaper. IPoWDM convergent SDN architecture - Motivation, technical definition & challenges", Telecom Infra Project, August 2022, [Online] https://cdn.brandfolder.io/D8DI15S7/at/n85t9h48bqtkhm9k7tqbs9fv/TIP_OOPT_MANTRA_IP_over_DWDM_Whitepaper_-_Final_Version3.pdf

[Gor20] J. Gordon et al., “Summary: Workshop on Machine Learning for Optical Communication Systems,” NIST Special Publication 2100-04, Mar. 2020 [available online: https://doi.org/10.6028/NIST.SP.2100-04]

[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

[GRAFANA] [Online] https://grafana.com/

[GRPC22] 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.

[INFLUXDB] [Online] https://www.influxdata.com/

[Ish20] K. Ishii et al., “Two-Level Abstraction Approach for SDN-based Service Provisioning in Open Line Systems Featuring TAPI Externalized Path Computation”. 2020 European Conference on Optical Communications, ECOC 2020, 2020 https://doi.org/10.1109/ECOC48923.2020.9333136

[Jur20] P. Jurkiewicz et al. “Flow length and size distributions in Campus Internet Traffic”, 2020 (available arXiv:1809.03486)

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

[K8sAPI] Kubernetes API [Online]. Retrieved October 13, 2022, from https://kubernetes.io/docs/reference/kubernetes-api/

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

[Mam19] L. Mamushiane et al., “Overview of 9 Open-Source Resource Orchestrating ETSI MANO Compliant Implementations: A Brief Survey”. 2019 IEEE 2nd Wireless Africa Conference, WAC 2019 – Proceedings, 2019. https://doi.org/10.1109/AFRICA.2019.8843421

[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

[MCN] FUTURE COMMUNICATION ARCHITECTURE FOR MOBILE CLOUD SERVICES | Grant agreement ID: 318109 | European Commission [Online]. Retrieved October 5, 2022, from https://cordis.europa.eu/project/id/318109

[MHEC] METRO High bandwidth, 5G Application-aware optical network, with edge storage, compUte and low Latency | METRO-HAUL Project | Grant agreement ID: 761727 | European Commission [Online]. Retrieved October 5, 2022, from https://cordis.europa.eu/project/id/761727/es

[Moh19] S. Mohanty et al., "An evaluation of open source serverless computing frameworks". Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom, 2018, pp. 115–120. https://doi.org/10.1109/CLOUDCOM2018.2018.00033

[Muq21] A. S. Muqaddas et al., "NFV orchestration over disaggregated metro optical networks with end-to-end multi-layer slicing enabling crowdsourced live video streaming". Journal of Optical Communications and Networking, Vol. 13, Issue 8, Pp. D68-D79, 13(8), 2021, D68–D79. https://doi.org/10.1364/JOCN.423501

[MUST] MUST Optical SDN Controller NBI Technical Requirements Document TIP OOPT PG - Version: 1.1 [Online] Retrieved October 5, 2022 from https://cdn.brandfolder.io/D8DI15S7/at/sp6tgqcpjp8rgsshf8pvmwpg/TIP_OOPT_MUST-Optical-SDN-Controller-NBI-Technical-Requirements-v11_FINAL_GREEN_ACCESS.pdf

[NSv16] "Framework for IETF Network Slices - draft-ietf-teas-ietf-network-slices" [Online]. Retrieved December 16, 2022, from https://datatracker.ietf.org/doc/draft-ietf-teas-ietf-network-slices/

[Nubomedia] NUBOMEDIA: an elastic Platform as a Service (PaaS) cloud for interactive social multimedia | Grant agreement ID: 610576 | European Commission [Online]. Retrieved October 5, 2022, from https://cordis.europa.eu/project/id/610576

[OCATA] 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.

[OCP} Open Compute Project [Online] https://www.opencompute.org/

[ODL] "OpenDayLight Transport PCE" [Online] https://docs.opendaylight.org/projects/transportpce/en/latest/developer-guide.html

[OIF] "OIF" [Online] https://www.oiforum.com/

[ONAP] "ONAP" [Online]. Retrieved October 5, 2022, from https://www.onap.org/

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

[ONL] "Open Network Linux" [Online] http://opennetlinux.org/

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

[ONOSREST] "ONOS REST APIs" [Online] https://wiki.onosproject.org/display/ONOS/Appendix+B%3A+REST+API

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

[OSM] Open Source Mano, OSM [Online]. Retrieved October 5, 2022, from https://osm.etsi.org/

[OSMRel4] A. Hoban et al., OSM Release FOUR Technical Overview OSM Release FOUR-A Technical Overview, 2018. www.etsi.org

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

[P4Lang] "P4Lang" [Online]. Retrieved October 5, 2022, https://github.com/p4lang

[Pao13] F. Paolucci et al., "A survey on the path computation element (pce) architecture". IEEE Communications Surveys and Tutorials, 15(4), 2013, pp. 1819–1841. https://doi.org/10.1109/SURV.2013.011413.00087

[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.

[Ped18] Pedreno-Manresa, J.J. et al. "On the Need of Joint Bandwidth and NFV Resource Orchestration: A Realistic 5G Access Network Use Case". IEEE Communications Letters, 22(1), 2018, pp. 145–148. https://doi.org/10.1109/LCOMM.2017.2760826

[Red22] Red Hat [Online], What is container orchestration?, 2022. Retrieved October 5, 2022, from https://www.redhat.com/en/topics/containers/what-is-container-orchestration

[RFC3917] J. Quittek, et al. (Eds.), "Requirements for IP Flow Information Export (IPFIX)," IETF RFC-3917, 2004.

[RFC5103] B. Trammell, et al. (Eds.), "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)," IETF RFC-5103, 2008.

[RFC7011] B. Claise, et al. (Eds.), "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," IETF RFC-7011, 2013.

[RFC7015] B. Trammell, et al. (Eds.), "Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol," IETF RFC-7015, 2013.

[RFC8040] A. Bierman, M. Björklund and K. Watsen, RFC 8040 "RESTCONF Protocol", January 2017

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

[Sca21] D. Scano, A. Giorgetti, A. Sgambelluri, E. Riccardi, R. Morro, F. Paolucci, P. Castoldi, and F. Cugini, "Hierarchical control of SONiC based packet-optical nodes encompassing coherent pluggable modules," in ECOC 2021

[Sga20] A. Sgambelluri, et al., "OpenROADM-controlled white box encompassing silicon photonics integrated reconfigurable switch matrix" in OFC 2020

[Sga21] A. Sgambelluri, D. Scano, A. Giorgetti, F. Paolucci, E. Riccardi, R. Morro, P. Castoldi, and F. Cugini, "Coordinating pluggable transceiver control in SONiC-based disaggregated packet-optical networks," in OFC 2021.

[Sli17] F. Slim et al., "Towards a dynamic adaptive placement of virtual network functions under onap". IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2017, 2017, pp. 210–215. https://doi.org/10.1109/NFV-SDN.2017.8169880

[SoftFIRE] Software Defined Networks and Network Function Virtualization Testbed within FIRE+ | Grant agreement ID: 687860 | European Commission [Online]. Retrieved October 5, 2022, from https://cordis.europa.eu/project/id/687860

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

[Stra] ONF Stratum [Online] https://opennetworking.org/stratum/

[TAI] "Transponder Abstraction Interface"[Online] https://github.com/Telecominfraproject/oopt-tai

[TAPI2.4]	Transport API 2.4.0 [Online], Retrieved 15 december at https://github.com/OpenNetworkingFoundation/TAPI/releases/tag/v2.4.0
[TELEGRAF]	[Online] https://www.influxdata.com/time-series-platform/telegraf/
[TFS]	ETSI Teraflow SDN Controller, [Online]. Retrieved October 5, 2022, from https://tfs.etsi.org/
[TIP]	Telecom Infra Project. [On-line] https://telecominfraproject.com/
[TR-547]	TAPI Reference Implementation Agreement TR-547 [Online] https://opennetworking.org/wp-content/uploads/2021/12/TR-547-TAPI_ReferenceImplementationAgreement_v1.1.pdf
[Uzu21]	D. Uzunidis, E. Kosmatos, C. Matrakidis, A. Stavdas and A. Lord, "Strategies for Upgrading an Operator's Backbone Network Beyond the C-Band: Towards Multi-Band Optical Networks," in IEEE Photonics Journal, vol. 13, no. 2, pp. 1-18, April 2021.
[Vil21]	R. Vilalta et al "Teraflow: Secured autonomic traffic management for a tera of SDN flows", in Proc of European Conference on Networks and Communications & 6G Summit, 2021
[Wel21]	D. Welch et al., ”Point-to-Multipoint Optical Networks Using Coherent Digital Subcarriers”, IEEE Journal of Lightwave Technology Vol 39, Issue 16, August 2021