D5.2 DESIGN, SPECIFICATION AND INITIAL IMPLEMENTATION OF THE CONTROL PLANE

Project title	**Photonics technologies for ProgrAmmable transmission and switching modular systems based on Scalable Spectrum/space aggregation for future aglle high capacity metrO Networks**
Project acronym	**PASSION**
Grant number	**780326**
Funding scheme	**Research and Innovation Action - RIA**
Project call	**H2020-ICT-30-2017 Photonics KET 2017 Scope i. Application driven core photonic technology developments**
Work Package	**WP05**
Lead Partner	**CTTC**
Contributing Partner(s)	**CTTC, SMO**
Nature	**R(report)**
Dissemination level	**PU (Public)**
Contractual delivery date	**30/10/2020**
Actual delivery date	**30/10/2020**
Version	**1.0**

History of changes

Version	Date	Comments	Main Authors
0.0	9/10/2020	Definition of the ToC based on the contents provided in MS16. Note: The delivery of this Deliverable was delayed according to the requested Project Extension. The initial delivery data was scheduled by the end of May 2020 and the new delivery data was postponed to the end of October 2020	Ricardo Martínez, Michela Svaluto Moreolo

0.1	14/10/2020	Completing the Experimental Evaluation Section with latest results based on both devised RSA algorithms (RSA-CR and RSA-IM).	Ricardo Martínez, Michela Svaluto Moreolo
0.2	16/10/2020	First integrated version relying on MS16 and adding new control plane validations / evaluations	Ricardo Martínez, Michela Svaluto Moreolo
0.3	19/10/2020	Comment and general review	Germano Gasparini, Giorgio Parladori
0.4	21/10/2020	Compile new version with SMO inputs	Ricardo Martínez, Michela Svaluto Moreolo, Javier Vilchez, Laia Nadal, Josep Maria Fabrega, Paolo Dini
0.5	27/10/2020	Comments from Quality Check Process	Ben Puttnam
0.6	27/10/2020	Addressing comments from Quality Check Process	Ricardo Martínez
1.0	28/10/2020	Final Document	Ricardo Martínez, Michela Svaluto Moreolo, Javier Vilchez, Laia Nadal, Josep Maria Fabrega, Paolo Dini, Germano Gasparini, Giorgio Parladori

Disclaimer

This document contains confidential information in the form of the PASSION project findings, work and products and its use is strictly regulated by the PASSION Consortium Agreement and by Contract no. 780326.

Neither the PASSION Consortium nor any of its officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

The contents of this document are the sole responsibility of the PASSION consortium and can in no way be taken to reflect the views of the European Union.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780326.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This deliverable aims at providing the validation of the implemented SDN controller designed in the context of WP2 and reported in [D2.2]. This encompasses validating the different control interfaces enabling the communication between the controller and the network element and device agents to handle the management and programmability of optical connections within the adopted PASSION metro optical network. To this end, it is considered the derived workflows coordinated by the SDN controller are validated. This means that the automatic functions and operations commanded by the controller are completely validated upon receiving an optical connection request until the connection is configured. Such a validation is exclusively made at the control plane level. That is, no data plane assessment is conducted and the full integration of control and data plane operations (i.e., connection establishment and configuration) is tackled in other activities within the context of WP5.

One of the key control plane functions covered in this deliverable regards the validation of the workflows based on the decisions made by a devised Routing and Spectrum Assignment (RSA). In a nutshell, the RSA algorithm takes over of the computation of the route as well as the resources to automatically accommodate incoming point-to-point optical connection requests. In this context, two RSA algorithms are experimentally evaluated, namely RSA Co-Routed optical flows (RSA-CR) and RSA Inverse Multiplexed optical flows (RSA-IM). The goal of benchmarking both RSA algorithms is twofold: i) validating the entire control plane operations and interfaces designed and implemented; ii) performing an exhaustive evaluation of the RSA algorithms tailored to the specific peculiarities and features of the underling PASSION solutions. For the latter, it is considered dynamic optical connection request arrival requiring heterogenous data rates.

1 INTRODUCTION

In [D2.2] it was reported the design of the SDN controller to enable the automatic computation, resource selection and programmability of optical connections within a Metropolitan Network (MAN) supporting flexible, elastic, and huge transport capacity investigated by the PASSION project (such as S-BVT Tx based on VCSELs, S-BVT Rx using coherent detection and different optical switch architectures addressing defined Hierarchical Levels). To this end, the required control functions to be handled within the SDN controller (e.g., on-line path computation, connection lifecycle management, etc.) were defined. Additionally, it was designed the control interfaces tailored to the features and characteristics of the underlying transport infrastructure. Bearing in mind the above, the macroscopic objectives covered by the designed SDN controller are:

1) Receiving optical connection demands from an upper-layer system / application
2) Supporting the automatic configuration / programmability of the underlying network devices and elements such as optical switches and S-BVT transceivers

The organization of this deliverable is as follows: first the implementation of the SDN controller is described, outlining the key building blocks and the interactions among them. Next, it is also tackled the defined and implemented control interfaces / APIs used for the communication between the higher-layer application requesting optical connection services and the SDN controller. Moreover, the defined APIs targeting the communication between the SDN controller and the underlying network elements and device agents to support optical connection lifecycle management (i.e., the establishment and deletion) are detailed. This is provided by means of describing the defined and implemented workflows encompassing the interactions within not only the SDN controller functions but also the interworking among the controller and the network element agents.

This deliverable thus reports the validation of the SDN controller functions, defined APIs, and complete workflows. To this end, we especially focus on the WP2 use case referred to as "*pay-as-you-grow*". To deal with this use case, different Routing and Spectrum Assignment (RSA) algorithms. As introduced in [D2.2]. are considered. The SDN control functions, APIs and RSA algorithms are experimentally evaluated at the control plane level within the CTTC ADRENALINE testbed. For the sake of completeness, it is worth outlining that the adopted model for generating the connection requests considers heterogenous data rates arriving dynamically and served/processed by the controller. The devised RSAs (running within the SDN controller path computation function) are triggered. Such RSA algorithms aim at seeking for each incoming connection request a feasible spatial and spectral path that fulfills the connection demands (in terms of bandwidth) as well as attaining an efficient use of the network resources (e.g., optical spectrum, S-BVT Transmitters and Receivers). Besides evaluating the performance of devised RSA algorithms for the targeted use case, it is also validated the subsequent operations, functions, and interfaces. The latter entails: i) support for executing of the selected RSA algorithm with updated resource availability; ii) actual configuration of the selected resources in the underlying network devices for accommodating the successfully computed connections.

2 IMPLEMENTED SDN CONTROLLER

[D2.2] provided a detailed description of the main interfaces and the SDN controller building blocks to handle the automatic programmability of the network devices and elements forming the underlying optical transport infrastructures. The implemented SDN controller architecture is shown in Figure 1. It is based on a proprietary CTTC implementation using C++ programming language. The SDN controller takes over of coordinating defined workflows entailing the functions, operations and interactions with an on-demand bandwidth application and the network element agents to either set up or release end-to-end optical connections fulfilling the requirements in terms of connection endpoints and bandwidth. To this end, the SDN controller enables:

a. Processing requests from an external upper-bound application for setting up or removing an optical connection. As said in [D2.2], such an application is referred to as "*on-demand bandwidth application*".

b. Computing end-to-end paths and selecting network resources to be eventually configured on the transport network. This entails triggering an on-line path computation process.

c. Interacting with the underlying agents and locally handling the selected resources to be programmed.

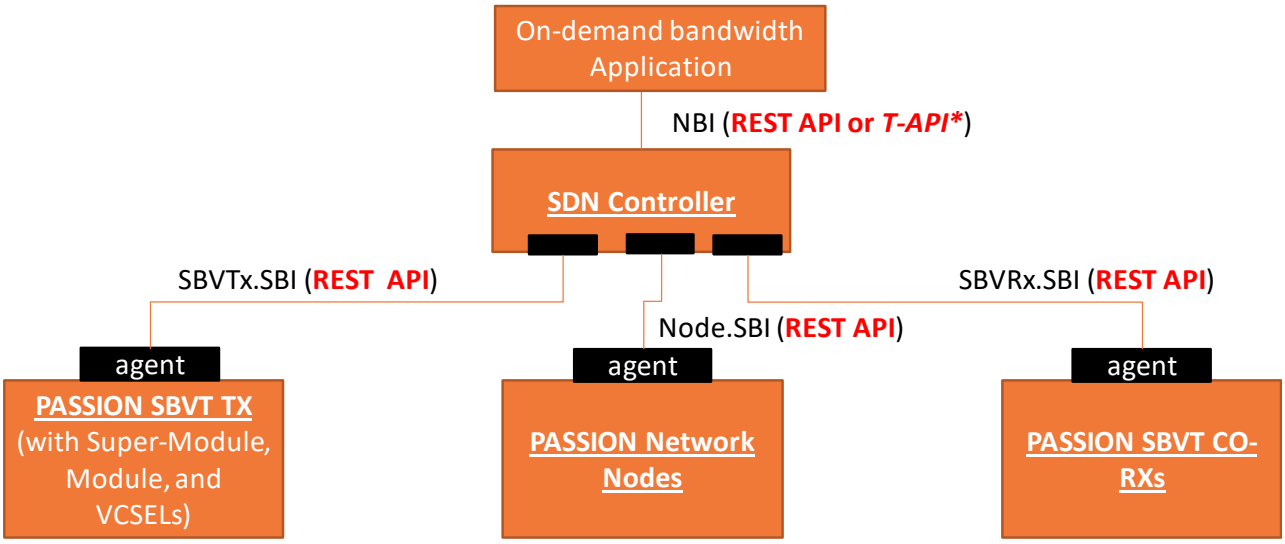

Figure 1. Schematic view of the PASSION SDN control architecture.

As shown in Figure 1, the SDN controller handles control interfaces with the on-demand bandwidth application as well as the control agents managing the programmability of the SBVT Tx, Rx and network nodes. The interface with the on-demand bandwidth Application is referred to as the Northbound Interface (NBI). The design of this client-server relationship interface (based on REST API) was tackled and described in [D2.2]. In section 2.1, the validation of such an NBI in terms of exchanged messages and carried contents is presented. Likewise, in [D2.2] a description of the devised control interfaces (i.e., the information model, messages, encoding) supporting the communication between the SDN controller and the network and device agents was provided. Such set of interfaces also create client-server relationships and are called as the Southbound Interfaces (SBI). In section 2.2, it is presented the validation of the set of designed SBIs enabling the SDN controller to actually handling the configuration of the network resources (i.e., SBVT Tx

VCSELs, SBVT CO-Rxs, optical spectrum and the optical switches) when managing the lifecycle of optical connection requests.

Prior to detail the conducted validations of both NBI and SBI, it is convenient to present the architectural aspects and key building blocks forming the implemented SDN controller. A logical representation of this controller is depicted in Figure 2. It is worth mentioning that a description of main functions and entities encompassed by the SDN controller was provided also in [D2.2].

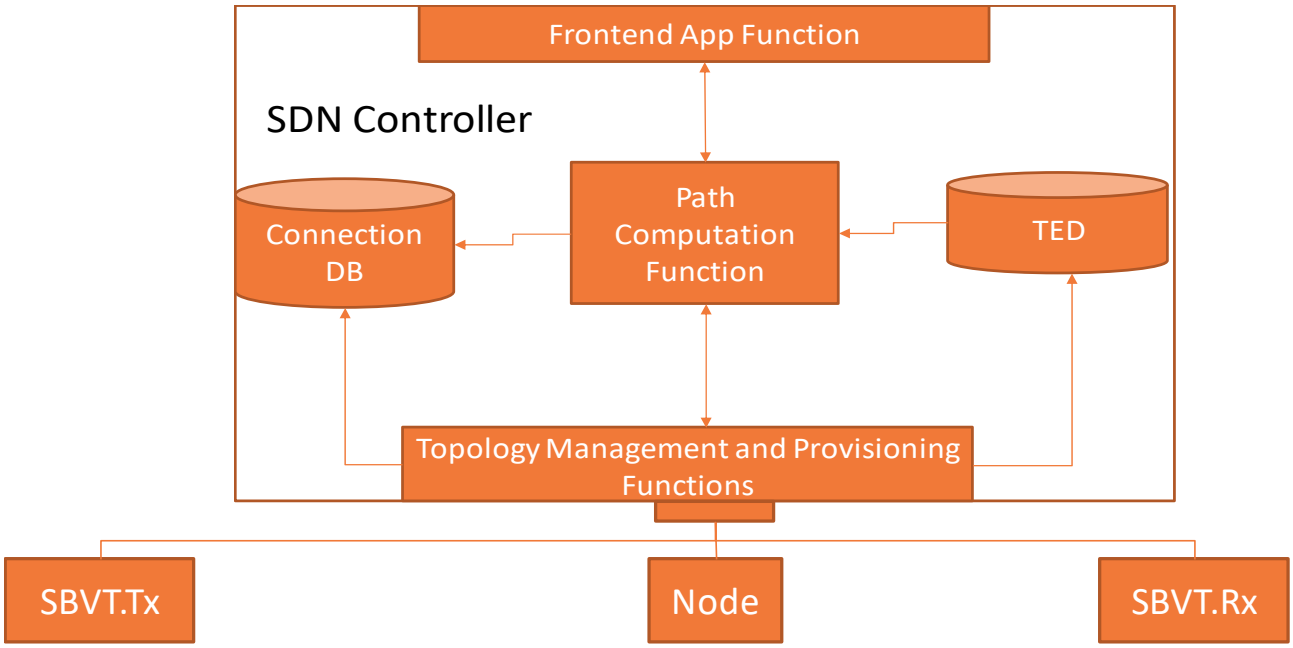

Figure 2. SDN Controller key building blocks and functions.

The key building blocks of the SDN controller are:

- *Frontend App Function*: this function provides the server-based operations to handle the NBI with the on-demand bandwidth Application. Thus, it processes the incoming requests (e.g., create and delete an end-to-end optical connection) arriving from an application.
- *Path Computation Function*: this constitutes the core function of the SDN controller coordinating most of the actions (e.g., triggering the end-to-end path computation) and operations (i.e., instantiating a new connection to be set up, etc.). This element also integrates a pool of RSA algorithms. Upon a new connection request, a specific RSA algorithm is triggered to compute a route and select the available resources fulfilling not only the demand requirements but also fostering the specific network objective function (e.g., to attain an efficient use of network resources).
- *Traffic Engineering Database* (TED). This repository is used to store information about the network connectivity (i.e., topology) along with the status of the available resources (e.g., optical link spectrum, available VCSELs at the SBVT Tx and CO-Rxs at the S-BVT Rx, etc.). From herein, the model where the topology and the connectivity among the network devices and network nodes is statically provided beforehand is adopted. Indeed, when triggering the SDN controller, there is a configuration XML-based file which is read by the SDN controller to create the TED contents. By doing so, the TED stores information about the connected remote ports at different network nodes, the supported attributes on the physical links (e.g., number of center frequencies, center frequency granularity, slot width granularity, link distance, etc.). Then, the availability of these resources is modified and

handled within the TED as connections are established and removed. Therefore, the TED information is considered as main input information to the RSA algorithms that are triggered.

- *Connection Database* (DB). This repository keeps track of the optical connections existing within the network. This DB stores information relevant to the connections such as its name (to identify the connection), the endpoints as well as the set of optical flows. In this regard, as it will be addressed in section 3, a connection request may demand larger bandwidth than the supported data rate by a single SBVT Tx VCSEL (set up to 50 Gb/s). Thus, if an optical connection requests a bandwidth > 50Gb/s, multiple optical flows related to the same connection need to be set up. In other words, a relationship 1:N (being N ≥ 1) between the connection request arriving from the NBI and the optical flows rolled out through the SBI is established. When a connection needs to be released, all the optical flows bound to that connection are removed (i.e., de-allocating their occupied network resources in terms of SBVT Tx VCSELS, RX CO-Rx, frequency slot on the traversed links) and then removed from the Connection DB.
- *Topology Management and Provisioning Function*: This function provides the client functions to handle the SBI operations with the underlying network elements and device agents. According to the required connection operations (i.e., establishment or deletion) it allows creating the message and encapsulating the contents to manage retrieval of resource information as well as the (de-)allocation of the network resources. Thus, it provides programmability of the end-to-end optical flows associated to every connection request.

As mentioned above, the operations and functions carried out within the SDN controller to manage the connection lifecycle are determined by well-defined workflows. These workflows determine the sequence of steps to be executed by the SDN controller involving the key functions defined above as well as the interactions through the NBI and SBI. The considered workflows were introduced in [D2.2], but herein they are also considered to better expose the validation framework of the implemented SDN controller interfaces covered in the subsequent sections.

Figure 3 shows the workflow to set up a new connection request. The connection request is sent by the *on-demand bandwidth application* to the SDN controller using the NBI POST (see section 2.1). The connection request includes the Identifier of the connection identifier (*id* in Figure 3), the source and destination nodes (*src* and *dst* in Figure 3) and the requested bandwidth (*bw* in Figure 3). It is important to recall that the source and destination nodes are bound to both SBVT devices, i.e. transmitter (Tx) and receiver (Rx).

The connection request (NBI) message is then received by the SDN controller at the Front-End App Function. Internally within the controller, the Path Computation Function is triggered via the Connection Request process. At this point, the Path Computation updates its network resource view within the TED repository. This is supported by the Topology Management and Provisioning Function relying on the defined SBIs:

i) the resource availability of the SBVT Tx devices (VCSELs) at the source node using the GET /…/sbvtTx message.
ii) the resource status of the SBVT Rx elements (CO-Rxs) at the destination node using the GET /…/sbvtRx) message

As mentioned above, the management of the network links (i.e., allocation and removal of optical spectrum resources) within the controller's TED is locally done. That is, once a connection is being

established, the controller updates the status of the available resources through those links accommodating the connection.

All the HTTP responses related to updating SBVT Tx and Rx availability through the SBIs are processed by the SDN controller and then conveniently reflected in the TED contents.

Once the TED is updated, the Path Computation Function triggers the routing algorithm (i.e. RSA). As said, the RSA uses as inputs the request's attributes (Conn. Req) and the updated TED. The output of the routing algorithm should provide:

i) the set of ordered network nodes to be traversed (detailing the input and output physical ports). This is referred to as spatial path;
ii) the selected frequency slot – FS - (i.e., optical spectrum) to accommodate the optical flow;
iii) the configurable parameters of both S-BVT Tx and S-BVT Rx such as the selected set of VCSELs and the CO-Rxs and their central frequencies (i.e. local oscillator, LO).

The programmability of each selected network element and device represented in the computed spatial and spectral paths is carried out separately using the defined SBIs. Specifically, the controller's Provisioning Function communicates with the specific network element and device agents. In the S-BVT Tx device, the VCSEL associated to the central frequency *centralFreq_n* is occupied and assigned to the specified *optFlowID*. This is done using the POST /…/sbvtTx/freqSlot message. Similarly, for the S-BVT Rx, the POST /…/sbvtRx/freqSlot message indicates to the corresponding agent to first select an available CO-Rx and then program its LO to be tuned at the specified central frequency *freqLocalOscillator_n*. Finally, for each traversed optical switch in the spatial path, an individual POST /…/opticalSwitch message is sent indicating the required cross-connection via *portIn*, *portOut*, *centerFreq_n*, etc.

Once the configuration of all the network elements and devices in the computed path is successfully done, the connection is established and thus added to the controller's Connection DB. Additionally, the successful connection establishment is notified to the on-Demand Bandwidth Application via the corresponding NBI-defined HTTP response message.

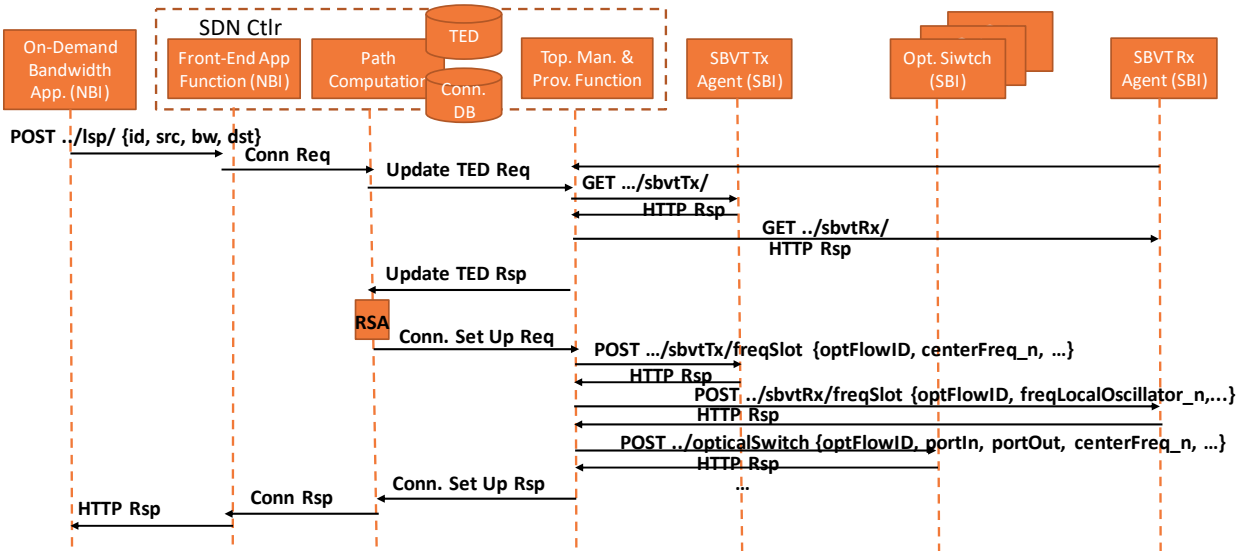

Figure 3. Workflow for setting up a new optical flow using defined NBI and SBI APIs

Figure 4 shows the workflow for removing an existing connection. The request is sent through the NBI from the on-demand bandwidth Application uniquely carrying the connection identifier (i.e., *id*). Upon receiving such connection removal request, the core function of the Path Computation

function firstly seeks for that identifier in the Connection DB. Since a connection may encompass several optical flows, it is retrieved by the identifier (i.e., *optFlowId*) for all the associated optical flows bound to the connection being removed.

Each of the optical flows needs to be removed independently via the defined SBIs. To this end, the controller's Provisioning Function communicates with the specific network element and device agents. In the S-BVT Tx device, the VCSEL/s occupied by the specified *optFlowID* are released (i.e., set occupied status to false). This is done using the DELETE /…/sbvtTx/freqSlot message. Similarly, for the S-BVT Rx, the DELETE /…/sbvtRx/freqSlot message indicates to the corresponding agent to release the occupied CO-Rx and set the LO null. Finally, for each traversed optical switch in the spatial path, an individual DELETE /…/opticalSwitch message is sent indicating the remove the associated cross-connection.

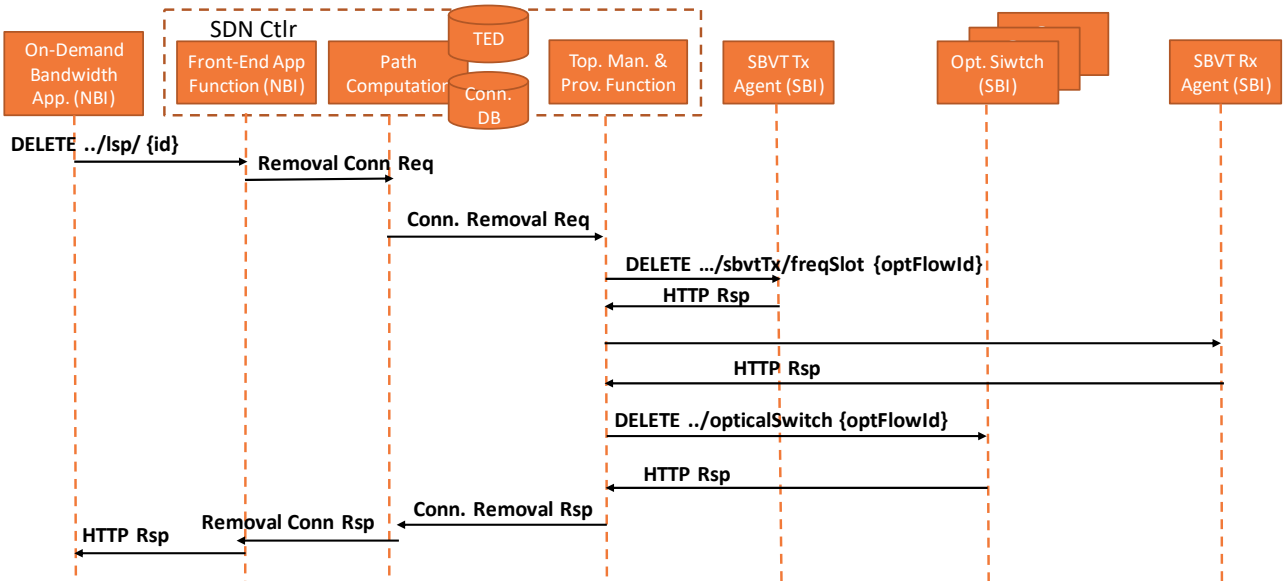

Figure 4. Workflow removing a connection using defined NBI and SBI APIs

Once all the resources in the network elements and devices related to the optical flows are de-allocated, the associated entry and all the information to the connection being released are eliminated from the Connection DB. Finally, the successful connection removal is notified to the on-Demand Bandwidth Application via the corresponding NBI-defined HTTP response message.

2.1 NORTHBOUND INTERFACE (NBI): VALIDATION

The validation of the NBI design [D2.2] is provided in this section. It is presented the captured messages exchanged between the *on-demand bandwidth application* and the SDN controller. Besides showing the exchanged RESTful API messages for requesting the establishment and removal of connections, the contents of those messages are also presented.

Figure 5 shows the pair of NBI RESTful request and response messages for requesting the establishment of a new connection. The contents carried in the request message (i.e. via POST method) are JSON-encoded. These specify:

- id: providing the connection id
- src: describing the ingress endpoint of the connection. Observe that this is specified using node IP address and port id. Such a port is physically connected to a SBVT Tx device.

- dst: describing the egress endpoint of the connection. Observe that this is specified using node IP address and port id. Such a port is physically connected to a SBVT Tx device.
- bw: specifying the amount of bandwidth demanded by the connection.
- bw_unit: specifying the units of bw (e.g., Gb/s)
- of: specifying the objective function which is used to determine the RSA algorithm and its capabilities (e.g., enabling protection, targeting specific optimization purposes, etc.) within the SDN controller.

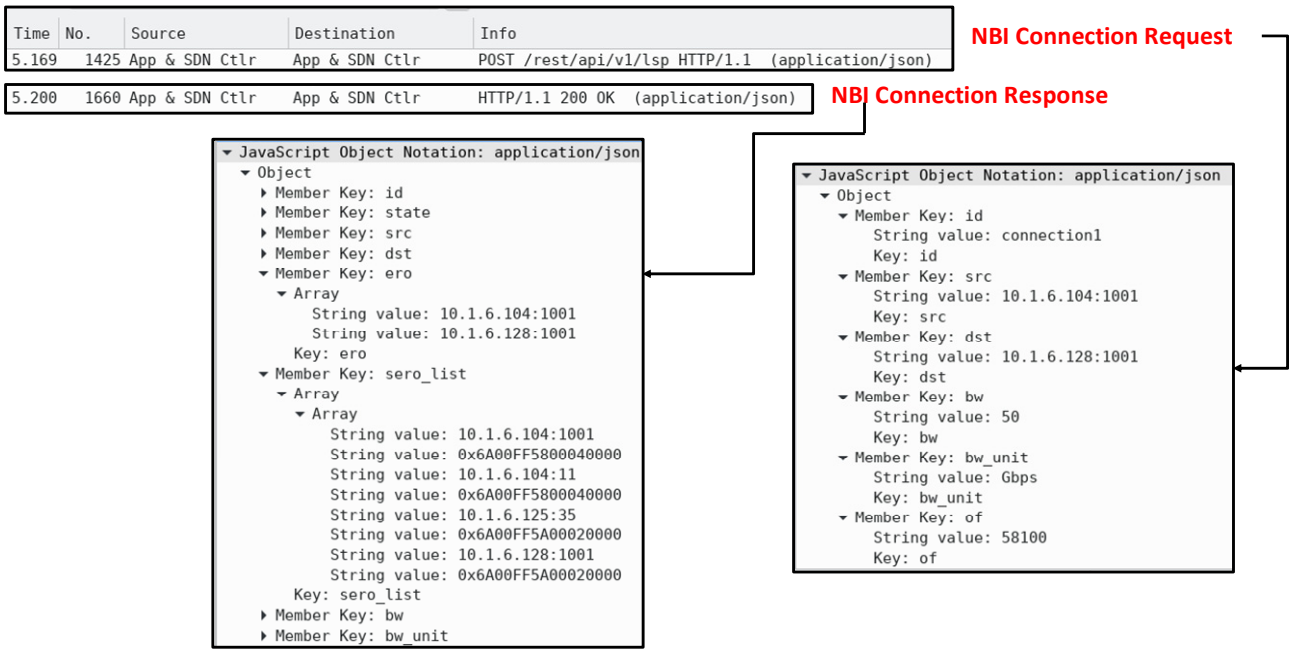

Figure 5. Validation of the on-demand bandwidth application and SDN Controller NBI: Connection Establishment

The successful establishment of a connection is notified to the *on-demand bandwidth application*. This is done via HTTP message with response code 200 OK. In the response body message, the following JSON-encoded contents are carried:

- id: providing the connection id
- state: informing that the connection is currently active
- src: ingress endpoint of the connection.
- dst: egress endpoint of the connection.
- ero: specifying the Explicit Route Object (ERO) to detail both the ingress and egress connection endpoints (i.e., src and dst).
- sero: secondary ERO which describes the path, links, and optical spectrum (i.e., FS) being allocated for each of the optical flows required to fulfill the connection request. Recall that a connection may entail setting up different number of optical flows within the underlying optical network infrastructure. The optical spectrum is represented as a FS determining the center frequency and the slot width encoded in 64 bits according to [ITU-T_G694.1]. In [D2.2], it is described how the FS for a specific center frequency and slot width is computed.
- bw: specifying the amount of bandwidth demanded by the connection.

- bw_unit: specifying the units of bw (e.g., Gb/s)

Figure 6 shows the pair of NBI RESTful request and response messages for removing an existing connection. To this end, in the URI of the DELETE method is included the id of the connection to be removed. The successful removal of the connection (i.e., de-allocating all the associated optical transport resources) is notified using HTTP message with response code 200 OK.

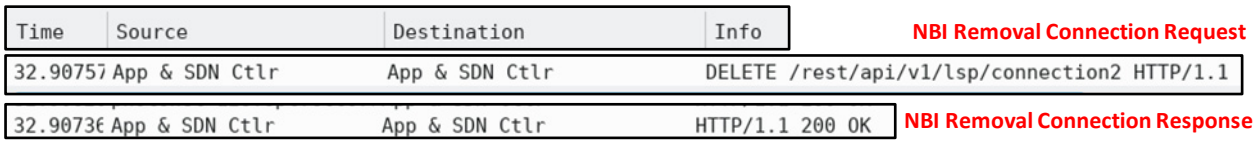

Figure 6. Validation of the on-demand bandwidth application and SDN Controller NBI: Connection Removal

2.1.1 NBI: Extended Capabilities

The above NBI validation basically focuses on enabling an upper-layer application over the SDN controller to simply handle the establishment and deletion of connectivity services within a single domain. This entails that a single SDN controller takes over of the computation and selection of the resources and their eventual configuration. However, this SDN controller's NBI can be extended to enable the application having more advanced functionalities supporting multiple domains governed by individual SDN controllers. To this end, a hierarchical control approach may be adopted where an orchestrator communicates with a pool of SDN controllers using their NBIs. The orchestrator is indeed a control entity which provides the coordination among the different SDN controllers to set up end-to-end multi-domains connections. Specifically, the overall optimal path across multiple SDN controller domains is computed by the NBI controller (i.e., orchestrator) where the individual SDN controller contributes with their local evaluation of the path segments forming the entire connection. To support this, a plausible approach relies on:

1. The orchestrator requests the SDN controller for a connection indicating the terminating points, the bandwidth needs and optional constraints
2. The SDN controller evaluates a list of possible (potential) implementations for the connection based on the available resources and returns the list to the NBI orchestrator
3. The NBI orchestrator decides based on its logic which element of the potential connection list is the suitable one and then sends a setup request towards the SDN controller
4. The SDN controller implements the connection, returning to the NBI orchestrator the details of the allocated resources (if needed)

This hierarchical approach also enables the evaluation of several alternative paths for a given connection request. This results in very interesting and useful to support recovery solutions (e.g., on-line restoration) where a pre-computed set of routes are available and upon a failure, one of these routes is chosen.

In any moment, the NBI orchestrator may require the actual status of the spectrum resources to the SDN controller that could be provided in terms of actually used frequencies (associated to the details of the connections that are up and running) and of the available/potential frequencies (associated to the potential connection end points).

For the connections, the following information could be provided in the NBI orchestrator request for allocation:

1. *Option A*: the NBI orchestrator indicates the end points and requests the overall bandwidth allocation for the connection request, detailing the overall requested spectrum by explicitly

providing the boundary central frequencies (i.e. the lowest and the highest). In this case, it is assumed that, if the requested bandwidth implies using multiple frequency slots, these latter are allocated contiguously.

2. *Option B*: the NBI orchestrator indicates the end points and requests only the overall bandwidth and spectrum needs of the connection, indicating only the granularity of the frequency slots. It is completely up to the SDN controller to identify the individual SBVT Tx VCSELs to be allocated for the service. In this case, there is not the constraint that the frequency slots used for the connection need to be contiguous. This may be more complex for the spectrum allocation algorithm but improve flexibility in the bandwidth and spectrum control and allocation. This option B may be more suitable for handling dynamic variation of the allocated bandwidth demand when compared to option A. In option B, the only blocking condition would be the exhausting of the available bandwidth, while option A fails also in case there is bandwidth, but it is not possible to allocate it on a contiguous set of frequency slots.

With both options, the information of all the allocated/identified FSs (in both SBVT TX and RX) shall be part of the NBI messages (responses) returning the allocated and implemented flows as response for the connection request. This provides the NBI orchestrator to be reported with topology details at the spectrum layer. To support this hierarchical orchestration/control architecture, the designed NBI needs to be enhanced to support the following set of advanced functionalities:

- *Retrieval of topological information*: this allows reporting through the NBI the actual status of the implemented connection as well as information about the available resources and possible pre-defined/existing connectivity. The topology view could be disaggregated, i.e. the potential end points of a connection could be encompassed by a logical node (termination node) while the pure (intermediate) switching capability is represented by separated logical nodes (switching node).

- *Path Computation*: this enables the pre-booking of resources (but not actual configuration/allocation) of a connection which may be interesting for restoration purposes.

- *Connectivity Management*: this entails the actual (de-)allocation of resources to accommodate a connection.

For the rest of this deliverable, this hierarchical orchestration/control model is not considered/addressed. Indeed, this deliverable exclusively focuses on a scenario where a centralized SDN controller handles the connection management within a single metro optical network domain.

2.2 SOUTHBOUND INTERFACE (SBI): VALIDATION

This section addresses the validation of the SDN controller SBI defined in WP2 [D2.2]. The SBI as commented above targets the communication with the agents controlling the programmability of the network elements and devices, namely: optical switches (also referred to as network nodes), the SBVT Transmitter and the SBVT Receiver. The SBI for all these network elements / devices covers: i) retrieval of updated information used for supporting on-line path computations (i.e., RSA); ii) allocation and de-allocation of the involved resources for managing the connection lifecycle. In the following the validations of the designed SBIs are presented individually on per network element and device basis. For all of them, the SBI creates client-server relationships relying on a RESTful API where the message payload is encoded using JSON format

2.2.1 SBI for Optical Switches

The objective of this SBI is to command the cross-connection between add/drop and express ports within an optical switch to set up the end-to-end optical flows between the SBVT Tx and SBVT RX at the connection endpoints. The set of operations being validated for this SBI (as described in [D2.2]) are:

i) creating the FS cross-connection for a specific optical connection request.
ii) removing the cross-connection associated to an active optical connection.

Figure 7 shows the pair of SBI RESTful request and response messages exchanged between the SDN controller and the optical switch agent for creating a cross-connection. To clarify, a cross-connection is entirely defined by the tuple: *portIn*, *portOut* and *FS* (i.e., center frequency and slot width). Further details are provided in [D2.2]. The successful cross-connection deployment is notified to the SDN controller using HTTP message with response code 201 CREATED.

Figure 7. Validation of the SBI between SDN Controller and Optical Switch Agent: cross-connection creation

The contents of the cross-connection creation request are JSON-encoded and carried into a POST method. These contents provide:

- connectionId: it specifies the optical flow identifier using the requested optical cross-connection
- portIn: in the optical switch it identifies the ingress port
- portOut: in the optical switch it identifies the egress port.
- centerFreq_n: it determines the center frequency (*n*) of the frequency slot being occupied by the optical flow according to [ITU-T_G694.1].
- slotWidth_m: it determines the slot width (*m*) of the frequency slot being used by the optical flow according to [ITU-T_G694.1].

Figure 8 shows the pair of SBI RESTful request and response messages exchanged between the SDN controller and the optical switch agent for removing an existing cross-connection bound to specific optical flow identifier. The successful cross-connection removal is notified to the SDN controller using HTTP message with response code 200 OK.

The optical switch removal request is done via a DELETE method message whose body contains (JSON-encoded) the following information:

- connectionId: it specifies the optical flow identifier whose active / existing optical cross-connection needs to be removed.

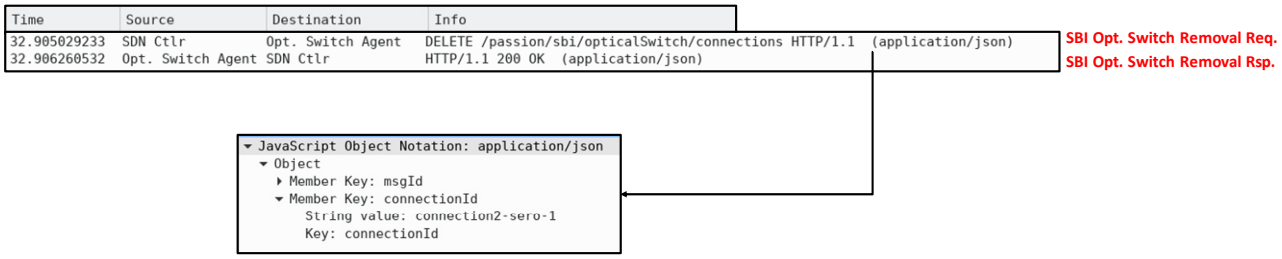

Figure 8. Validation of the SBI between SDN Controller and Optical Switch Agent: cross-connection removal

2.2.2 SBI for SBVT Transmitter

This SBI targets the specific programmability of the PASSION-designed SBVT Tx [D2.2]. Three specific SBI operations are supported for the interactions between the SDN controller and the agent governing an SBVT Tx:

i) retrieving the status of the Modules/SubModules/VCSELs.
ii) allocating a selected set of VCSELs to accommodate a specific connection request.
iii) releasing occupied VCSELs by a given active optical connection.

Figure 9 depicts the pair of SBI RESTful request and response messages to retrieve the status of the SBVT Tx device. To this end, the SDN controller queries via a GET method to the SBVT Tx Agent. The latter responds with the status of all the components forming the SBVT Tx device. Recall, that the SBVT Tx design within PASSION follows a modular implementation in terms of Module/SubModule/VCSELs. Further information about this design and how it is handled at the SDN controller is detailed in [D2.2]. The response message (with response code 200 OK) has the following contents:

- numModulesTx: it determines the number of Modules deployed within the SBVT Tx (ranging from 1 to 4)
- For those numModulesTx, an array is provided referred to as modulesTx. For each element of such an array, it is specified:
    - moduleTxId: it specifies the identifier of a specific modulesTx element
    - For a given moduleTx, there are 4-element SubModulesTx array. For each subModulesTx element, there is:
        - subModuleTxId: it specifies the identifier of a specific subModuleTx element
        - A single subModuleTx element has an array (up to 10) VCSEL devices (referred to as VCSELs). For each VCSELs array element, the following attributes are passed:
            - vcselId: it determines the identifier for the VSCEL element to be programmed.
            - usedState: it is Boolean value determining whether the VCSEL is occupied
            - bandwidth: it provides the optical spectrum occupied by each VCSEL. In PASSION project, this is set to 20 GHz

- central-frequency: it is a float value determining the central frequency (optical carrier) associated to a given VCSEL within the PASSION spectral range.
- modulation-format: it is an integer specifying the modulation format.
- fec: it is an integer detailing the FEC type being considered for configuring.

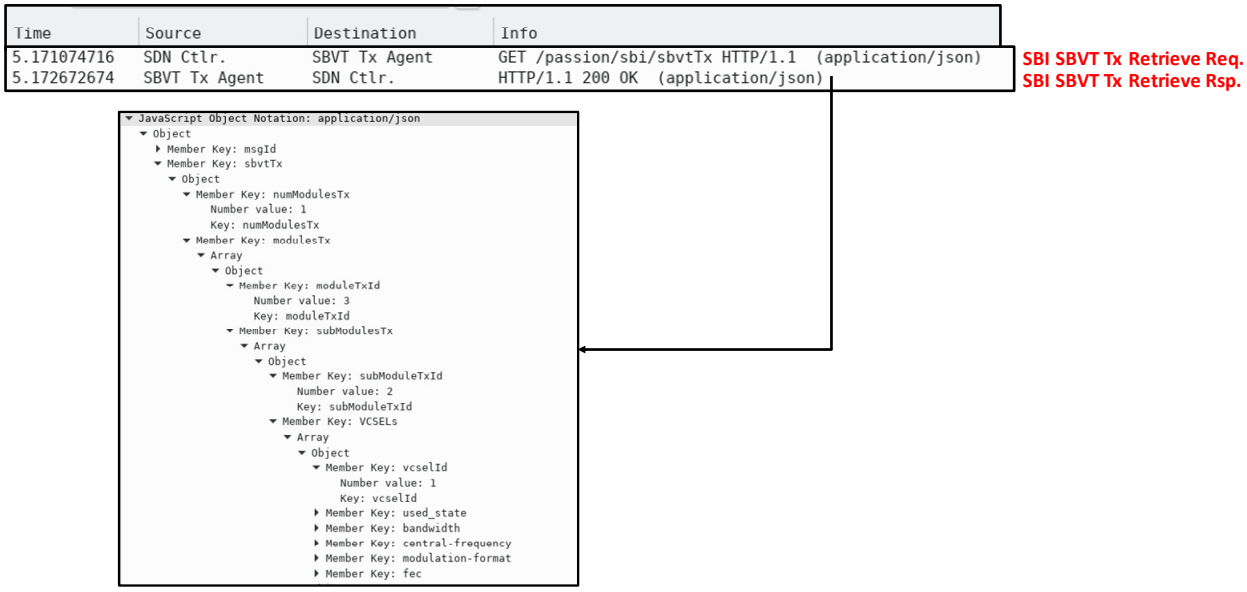

Figure 9. Validation of the SBI between SDN Controller and SBVT Tx Agent: SBVT Tx status retrieving

Figure 10 depicts the pair of SBI RESTful request and response messages to allocate (i.e., occupy) selected SBVT Tx VCSEL resources. To this end, the SDN controller sends a POST method to the SBVT Tx Agent specifying the VCSELs to be used for a given optical flow. A successful allocation of the selected resources is replied by the SBVT agent via an HTTP message with response code set to 201 CREATED.

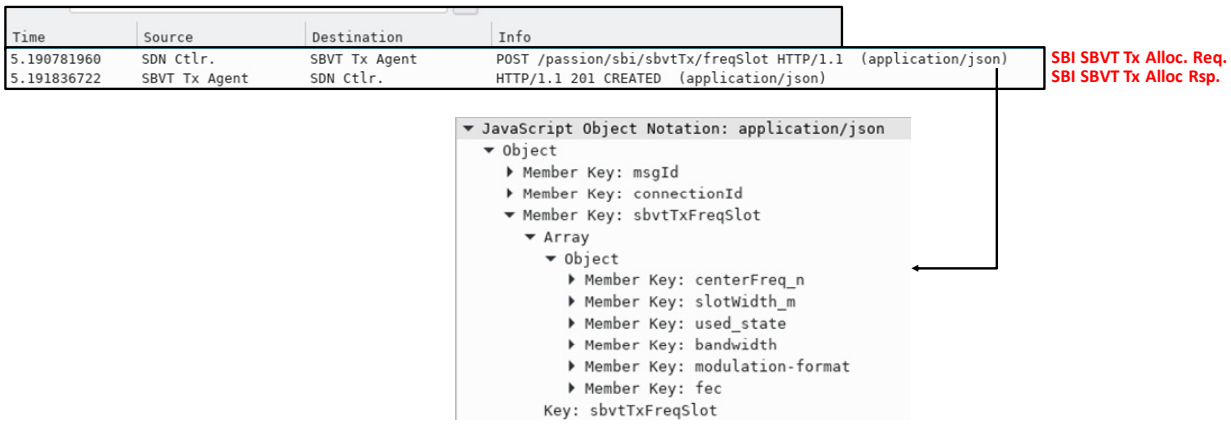

Figure 10. Validation of the SBI between SDN Controller and SBVT Tx Agent: SBVT Tx allocation

The SBVT Tx allocation request (i.e., POST method) carries the following JSON-encoded contents:

- connectionId: it specifies the optical flow identifier using the requested optical cross-connection

- sbvtTxFreqSlot array describing the pool of SBVT Tx VCSELs to be occupied. Indeed, the allocation of the VCSELs can be done either: i) describing the ModuleTxId, SubModuleTxId and VCSELId; or ii) describing the center frequency which is unique for each VCSEL (bound to a specific optical carrier) within an SBVT Tx device. For this second option, the contents are:
    - centerFreq_n: it determines the center frequency (*n*) of the frequency slot being occupied by the optical flow according to [ITU-T_G694.1]. Again, each VCSEL is associate to a dedicated optical carrier. Thus, determining the central frequency allows implicitly determining a specific VCSEL in the SBVT Tx device
    - slotWidth_m: it determines the slot width (*m*) of the frequency slot being allocated by the optical flow according to [ITU-T_G694.1]. In PASSION project, VCSEL bandwidth is 20 GHz; thus, in a slot width granularity of flexi-grid networks (i.e., 6.25 GHz), slotWidth_m is set to 4.
    - used_state: it is a Boolean parameter which is set to True when allocating resources
    - bandwidth: Optical spectrum occupied by each VCSEL (i.e., 20 GHz)
    - modulation-format: it is an integer specifying the modulation format.
    - fec: it is an integer detailing the FEC type being considered for configuring.

Figure 11 shows the pair of SBI RESTful request and response messages exchanged between the SDN controller and the optical switch agent for releasing the allocated SBVT Tx resources (i.e., VCSELS) bound to specific optical flow identifier. The successful SBVT Tx resource removal is notified to the SDN controller using HTTP message with response code 200 OK.

The SBVT Tx resource removal request is done via a DELETE method message whose body contains the following (JSON-encoded) information:

- connectionId: it specifies the optical flow identifier whose SBVT Rx resources are released.

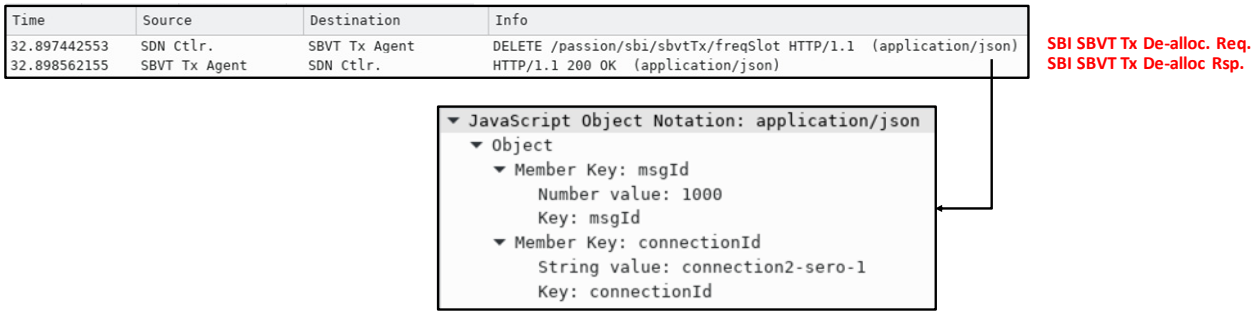

Figure 11. Validation of the SBI between SDN Controller and SBVT Tx Agent: SBVT Tx de-allocation

2.2.3 SBI for SBVT Receiver

This SBI supports the configuration of the SBVT Rx adopted in PASSION [D2.2]. Three specific SBI operations for the interactions between the SDN controller and the agent governing a SBVT Rx are implemented:

i) Retrieving the status of the Modules and their optical receivers (CO-Rxs).
ii) Allocating and configuring the selected CO-Rxs to accommodate a connection demand.
iii) Releasing occupied CO-Rxs by a given active optical connection.

Figure 12 depicts the pair of SBI RESTful request and response messages to retrieve the status of the SBVT Rx device. To this end, the SDN controller sends a GET method to the SBVT Rx Agent, and the latter responds with the status of all the components forming the SBVT Rx device. The SBVT Tx design within the project follows a modular implementation in terms of Module/Optical Receiver. Further information about this design and how it is handled at the SDN controller is provided in [D2.2]. Therefore, the response message (with response code 200 OK) has the following contents:

- numModulesRx: it determines the number of Modules deployed within the SBVT Rx (ranging from 1 to 4)
- For those numModulesRx, an array is provided referred to as modulesRx. For each element of such an array, it is specified:
    - moduleRxId: it specifies the identifier of a specific modulesRx element
    - numOpticalReceivers: it specifies the number of optical receivers embedded in each moduleRx
    - opticalReceivers is an array containing the set of optical receivers. For each opticalReceivers element, there is:
        - opticalReceiverId: it specifies the identifier of an optical receiver
        - usedState: it is Boolean value determining whether the optical receiver is occupied
        - bandwidth: it provides the optical spectrum occupied by each VCSEL. In PASSION project, this is set to 20 GHz
        - freqLocalOscillator: it specifies the frequency being tuned by a local oscillator in that occupied (i.e., used) receiver.

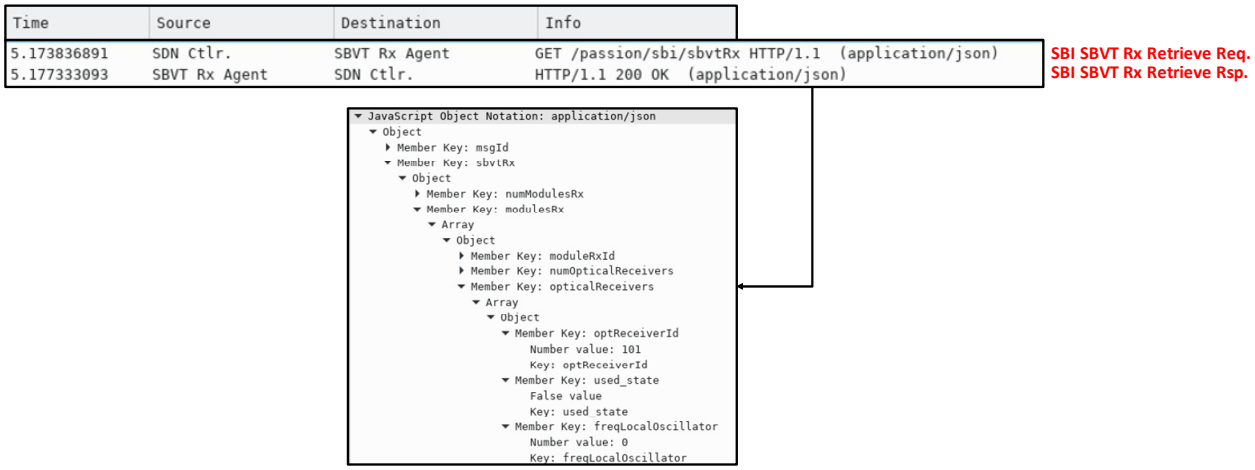

Figure 12. Validation of the SBI between SDN Controller and SBVT Rx Agent: SBVT Rx status retrieving

Figure 13 depicts the pair of SBI RESTful request and response messages to allocate (i.e., occupy) specific SBVT Rx coherent receivers. To do so, the SDN controller sends a POST method to the SBVT Rx Agent specifying the coherent receivers to be allocated for a given optical flow. In this message, it is not specified the Module where the coherent receiver needs to be allocated. This is delegated to the specific agent to find the first optical receiver being available. A successful

allocation of the selected resources is responded by the SBVT agent via an HTTP message with response code set to 201 CREATED.

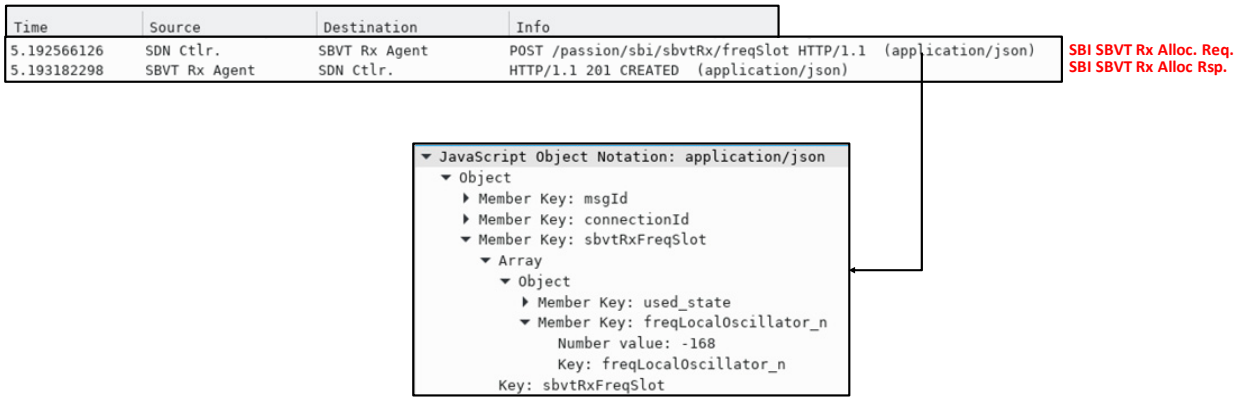

Figure 13. Validation of the SBI between SDN Controller and SBVT Rx Agent: SBVT Rx allocation

The SBVT Rx allocation request (i.e., POST method) carries the following JSON-encoded contents:

- connectionId: it specifies the optical flow identifier using the requested optical cross-connection
- sbvtRxFreqSlot array describing the pool of SBVT Rx coherent receivers to be occupied. For each coherent receiver it is notified the central frequency to be tuned by the local oscillator. Thus, the contents to do so are:
    - used_state: it is a Boolean parameter which is set to True when allocating resources
    - freqLocalOscillator_n: using the ITU-T flexi-grid nomenclature [ITU-T_G694.1] it is specified the center frequency (*n*) that the local oscillator of the coherent receiver needs to be tuned.

Figure 14 shows the pair of SBI RESTful request and response messages exchanged between the SDN controller and the optical switch agent for releasing the allocated SBVT Rx resources (i.e., coherent receivers) bound to specific optical flow identifier. The successful SBVT Rx resource removal is notified to the SDN controller using HTTP message with response code 200 OK.

The SBVT Rx resource removal request is done via a DELETE method message whose body contains (JSON-encoded) the following information:

- connectionId: it specifies the optical flow identifier whose SBVT Rx resource are released.

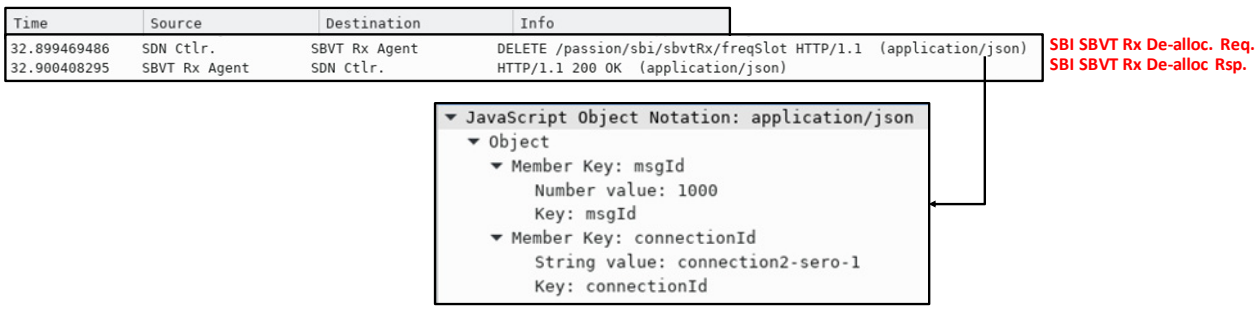

Figure 14. Validation of the SBI between SDN Controller and SBVT Rx Agent: SBVT Rx allocation

3 VALIDATION OF THE IMPLEMENTED SDN CONTROL PLANE

This section is devoted on validating the SDN controlller functions and interfaces (i.e., NBI and SBI) when dynamically setting up (and removing) optical connections demanding heterogeous data rates. This allows dealing with the defined PASSION project Use Case #1 "*pay-as-you_grow*". To do so, it is considered two path computation algorithms named as RSA Co-Routed Optical Flows (RSA-CR) and RSA Inverse Multiplexed Optical Flows (RSA-IM). Part of this work has been presented in [ONDM2020_Rmartinez] [BROADNETS2020_RMartinez]. Prior to discussion of the obtained performance of the devised RSA algortihms in a dynamc traffic scenario, the adopted metro network scenario is first presented.

3.1 DEPLOYED OPTICAL METRO NETWORK SCENARIO

The considered optical flexi-grid metro network is shown in Figure 15. It follows a star-ring topology formed by three identical clusters: A, B and C. Each cluster has a pair of linear network branches interconnected through a ring network. The deployed optical switches through the network infrastructure are mapped into the defined HL node types. HL4 nodes (represented in Figure 15 as black squares) provide traffic aggregation at the access network segment. This is done via an attached S-BVT (represented by a circle in Figure 15). These S-BVT devices has only 20 licensed VCSELs providing a maximum aggregated capacity of 1Tb/s. Additionally, 20 CO-Rxs are deployed at every HL4 S-BVT Rx. S-BVT VCSELs operate at a specific optical carrier within the targeted spectral range (i.e., between 191.900 and 195.875 THz). To support all the frequencies over that range within a cluster, 4 S-BVTs are needed per branch whose optical carriers are spaced 50 GHz. As shown in Table 1, clusters' S-BVTs are labeled by 1A, 1B, 2A, ..., 4B. Thus, optical carriers within a single cluster are spectrally spaced to 25GHz. The S-BVT's VCSELs optical carriers for both branches within a single cluster are deployed to avoid spectral collisions considering the filtering restrictions of the HL4 nodes. These nodes are built using Arrayed waveguide Gratings (AWG) with a channel spacing of 50GHz. The LO of every S-BVT's CO-Rx is fully tunable.

Table 1 Deployed S-BVT VCSELs (supported central frequencies) and CO-Rxs

S-BVT	#VCSELs & CO-Rxs	Supported Frequencies (THz)
1A	20	191.900, 192.100, 192.300, …, 195.700
1B	20	192.000, 192.200, 192.400, …, 195.800
2A	20	191.925, 192.125, 192.325, …, 195.725
2B	20	192.025, 192.225, 192.425, …, 195.825
3A	20	191.950, 192.150, 193.350, …, 195.750
3B	20	192.050, 192.250, 192.450, …, 195.850
4A	20	191.975, 192.175, 192.375, …, 195.775
4B	20	192.075, 192.275, 192.475, …, 195.875
Full (F)	160	191.900, 191.925, 191.950, …, 195.875

HL3 nodes, represented by red squares in Figure 15, are the transit optical switches with 25GHz filters based on Wavelength Selective Switch (WSS) technology enabling all-optical routing between HL4 and HL2/1 nodes. No S-BVTs are placed at HL3 transit nodes. Finally, HL2/1 nodes (blue squares in Figure 15) also use a 25GHz filtering optical switch. This node has 3-fully equipped S-BVTs with 160 VCSELs and Co-Rxs per each S-BVT (see Table 1). Hence, those S-

BVTs enable all the 25GHz-spaced carriers within the spectral range providing up to 8 Tb/s transport capacity towards the HL4 nodes. Finally, physical links between neighboring HLx nodes have a pair of fibers.

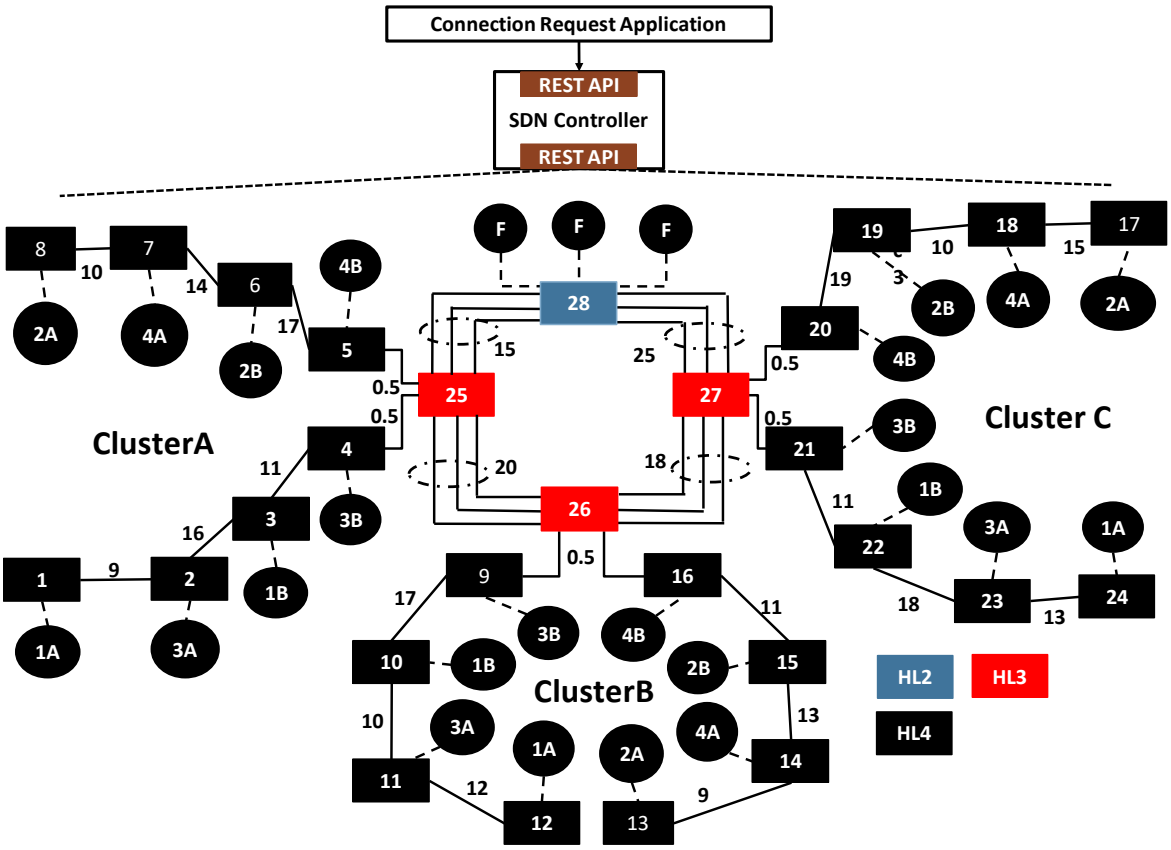

Figure 15. SDN-controller Optical Metro Network Scenario.

3.2 Devised Routing and Spectrum Assignment (RSA) Algorithms

The experimental evaluation of the devised RSA algorithms allows validating the designed and deployed SDN controller functionalities to dynamically and automatically serve optical connections with different data rates in terms of Gb/s. The conducted evaluation is exclusively realized at the control plane level. In other words, the network elements and devices forming the considered metro transport infrastructure are emulated.

The optical connections requests *req*s arriving to the SDN controller determine 3 pieces of information: the source node (src), the destination node (dst) and the bandwidth (bw) in Gb/s. Upon receiving a *req*, the selected RSA algorithm looks for a feasible route and resource to fulfill the *req* demands. This entails selecting: i) the src's S-BVT Tx resources (i.e., VCSELs); ii) the dst's S-BVT Rx resources (i.e., CO-Rxs with the central frequency at each LO); iii) the spatial path; and iv) the spectral path (i.e., frequency slots, FSs).

It is assumed that a *req* may demand a *bw* higher than the maximum capacity supported by a single S-BVT Tx VCSEL (i.e., 50 Gb/s). To deal with that, the establishment of a *req* may require using a *Num* (>= 1) different optical flows. Each of these optical flows is associated to an optical signal transported between a specific pair of S-BVT Tx and Rx devices. Every optical flow allocates an exclusive VCSEL and CO-Rx at the S-BVT endpoints which are not shared among other optical flows even if they belong to the same *req*.

Depending on the spatial path (distance and number of hops), the "net" VCSEL data rate at the dst node could be lower than the maximum transmission rate due to the accumulated physical impairments degrading the optical signal quality. To handle this, it is assumed that the S-BVT VCSELs can operate into three *operational modes* (OM), namely: *high*, *medium,* and *low*. Each OM provides different data rates (as shown in Table 2) according to the maximum path distance and number of crossed hops. At the time being, the accurate figures supporting each data rate per OM for both maximum path distance and number of hops is being experimentally evaluated in the context of other project WPs. The outcomes of these experiments will be eventually used to refine the values shown in Table 2.

Without loss of generality, when processing a *req* at the SDN controller, the RSA algorithm explores iteratively the defined OMs with the objective to select the one attaining the higher VCSEL data rate as long as both the distance and the number of hops restrictions are not exceeded. The *Num* of optical flows for a *req* to be allocated are associated to an individual S-BVT VCSEL, S-BVT CO-Rx and FS (i.e., ITU-T flexi-grid *n* and *m* parameters). Observe that the optical flow's FS is determined by the selected S-BVT Tx VCSEL (optical carrier), the HL node filtering capabilities, and the spectrum continuity and contiguity constraints to be ensured at every link forming the end-to-end path.

Table 2 VCSEL Operational Modes (OM)

VCSEL Operation Mode (OM)	Data Rate (Gb/s)	Maximum Path Distance (km)	Maximum Number of Hops
High	50	-	-
Medium	40	-	-
Low	25	-	-

In addition to the *req* parameters, the inputs of any devised RSA algorithm are provided by the information stored in the TED repository, which are: the src's S-BVT Tx (i.e., VCSELs) availability and their assigned optical carriers; ii) the dst's S-BVT Rx (CO-Rx) availability; iii) the network topology and the link attributes such as the available optical spectrum (i.e., unused nominal central frequencies, NCFs) and distance. The following notation (Table 3) is used to describe the devised RSA algorithms:

Table 3 Notation for describing devised RSA algorithms

`req`	`Optical connection request`
`src, dst`	`Source and destination nodes of req`
`bw`	`Data rate (in Gb/s) of req`
`OM`	`Set of VCSEL's OM sorted by data rate`
`OM[i].r`	`Data rate (in Gb/s) for OM[i]`
`OM[i].d`	`Maximum distance (in km) for OM[i]`
`OM[i].h`	`Maximum number of hops for OM[i]`
`Num`	`Number of VCSELs and CO-Rxs using OM[i]; Computed as upper integer of bw / OM[i].r`
$vcsel_n$	`Number of unused VCSELs at node n`

$vcsel_n$.NCF	Unused NCFs on available VCSELs at node n
$coRx_n$	Number of unused CO-Rxs at node n
$coRx_n$.NCF	Used NCFs by the occupied CO-Rxs at node n.
A	K Shortest Paths (K-SPs) between src and dst nodes sorted by distance and number of hops
A[k].d	Distance (in km) of computed k^{th} SP
A[k].h	Number of hops of computed k^{th} SP
A[k].NCF	Unused and common NCFs (bitmap) over spatial k^{th} SP
A[k].setNCF	Unused and common NCFs (bitmap) considering S-BVT (VCSEL and CO-Rxs) endpoints. Computed intersecting A[k].setNCF & $vcsel_{src}$.NCF & $coRx_{dst}$.NCF
FS	ITU-T flexi-grid Frequency Slot specifying selected n and m
findFS	Function returning first feasible FS over A[k].setNCF; otherwise returns NULL
P	Set of Num paths (p) for each optical flow in req
p_l	Path l for the optical flow in req; $0 \leq l < Num$
P_l.route	Spatial path for p_l
P_l.FS	Frequency Slot (n and m) for pl

3.2.1 RSA-CR Algorithm

The objective function of the RSA-CR algorithm is to accommodate *reqs* attaining the more efficient use of all network resources (i.e., S-BVT Tx and Rx and optical spectrum), where the resulting *Num* optical flows are routed along the same spatial paths (i.e., nodes and links). Figure 16 depicts an example of the RSA-CR algorithm. A *req* of 100 Gb/s between S-BVT Tx at src node 9 and S-BVT Rx at dst node 28 arrives. The RSA-CR output determines that two VCSELs (at high operational mode, i.e., 50Gb/s) and two CO-Rxs to be allocated. Consequently, two optical flows (i.e., X and Y) need to be set up.

Optical flow X allocates S-BVT's VCSEL with the optical carrier at 192.050THz; whilst optical flow Y allocates the VCSEL with the optical carrier at 192.250THz. At the dst, the respective CO-Rx LOs are programmed to receive such optical carriers. Both X and Y optical flows have their own FS (i.e., n, m). Attribute n describes the VCSEL optical carrier for the optical flow. On the other hand, m describes the slot width resulting from both the VCSEL optical bandwidth (25GHz) and the filtering features at each path node. Since heterogenous HL nodes are traversed, this enforces that the FS' n and m parameters over the path vary depending on the HL node filtering capabilities. That is, at node 9 (HL4), the FS of the optical flow X occupies n = -168 (i.e., ITU-T channel for 192.050THz) and m = 4 (slot width of 50GHz). For the optical flow Y at the same node the FS is n = -136 and m = 4. However, at HL3 nodes (26 and 25) and HL2/1 node (28) having filters of 25GHz, the FS for optical flow X uses n = -166 and m = 2. Likewise, for the optical flow Y, the FS in nodes 26, 25 and 28 is determined by n = -134 and m = 2. Although an optical flow's FS parameters (n and m) may vary along a path, the same optical spectrum carrying effective data (i.e., within 25GHz) remains spectrally continuous and contiguous from end-to-end.

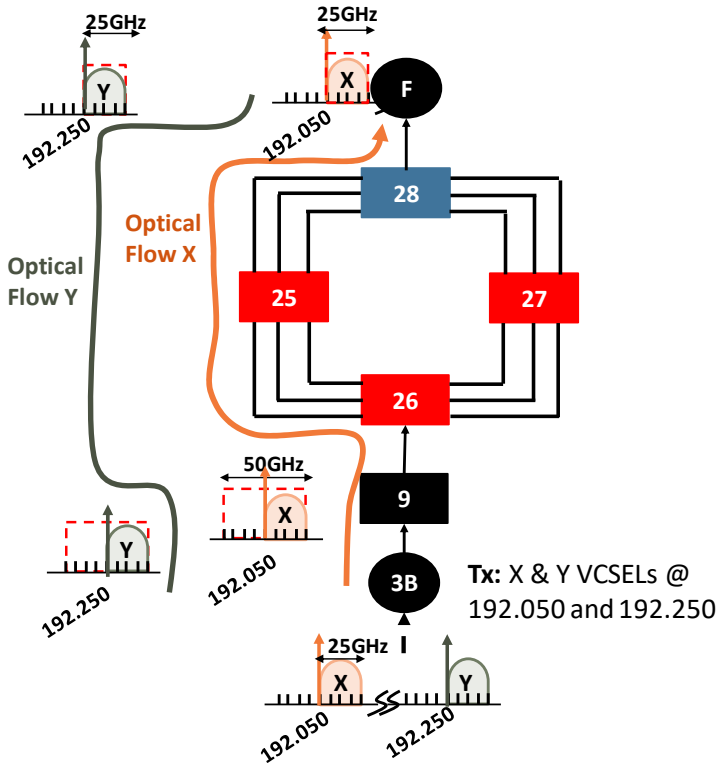

Figure 16. Example of RSA-CR algorithm.

The pseudocode of the RSA-CR is detailed in Algorithm 1 (see Figure 17). Upon receiving a *req*, the RSA computes the set *A* containing the K shortest paths (K-SPs) between the src and the dst nodes sorted by the distance and traversed number of hops. Then, the RSA iterates through the VCSEL *OM*s starting from the *high* mode until a feasible *A* path is found. If all *OM*s are checked but no solution is obtained, the *req* is blocked (i.e., *NoPath*). For an explored *OM[i]*, the *Num* of unused VCSELs and CO-Rxs at both S-BVT Tx and Rx is computed. *Num* is obtained as the higher integer above *bw / OM[i].r*. Recall that *Num* also determines the number of optical flows to be established. If either the available number of VCSELs at src node or the CO-Rxs at the dst node are lower than the required *Num*, the *req* is blocked. Otherwise, the computed *A* paths are checked. The first *A[k]* path attaining a feasible solution is selected and their computed resources (i.e., VCSELs, CO-Rxs, and FSs) are allocated. To do this, for the *A[k]* path, it is checked that both the distance and number of hops do not exceed the maximum permitted values determined by *OM[i]* (i.e., *OM[i].r* and *OM[i].h*). If this occurs, *A[k]* path is discarded. Otherwise, the set of unused and common NCFs (i.e., *A[k]:setNCF*) is computed. This is done by intersecting the available NCFs along the *A[k]* path links, the associated NCFs of each available VCSEL at the source node ($vcsel_{src}.NCF$), and those NCFs not being occupied by the used CO-Rxs at the dst node (i.e., $coRx_{dst}.NCF$).

The resulting *A[k].setNCF* becomes the input for the *findFS* function. This looks for a common and available FS from the S-BVT Tx (VCSELs), spatial path and S-BVT Rx (Co-Rxs). If it succeeds, the FS is associated to a path (i.e., *p*) for one out of the *Num* optical flows to be computed. Then, *p* is appended to the P set. Those spectral resources (i.e., NCFs) computed for the FS in the previous *p* are removed from *A[k].setNCF*. This does avoid double booking in the remaining optical flows to be computed. Finally, if *P* reaches the targeted *Num* of optical flows, the RSA-CR successes. Otherwise, *A[k]* path is discarded and next *A[k+1]* is explored. If none of the A paths provides a feasible solution, the next *OM[i+1]* (with lower VCSEL data rate) is considered.

Algorithm 1 RSA-CR; Input: $req\{src, dst, bw\}$

```
1: Compute A; i ← 0
2: while i < OM do
3:     Compute Num
4:     if Num > vcsel_src OR Num > coRx_dst then
5:         return NoPath
6:     end if
7:     while k < K do
8:         if A[k].d > OM[i].d OR A[k].h > OM[i].h then
9:             Continue                          ▷ next k^th SP
10:        end if
11:        P ← {}; l ← 0
12:        Compute A[k].setNCF
13:        while l < Num do
14:            p_l.route ← A[k]
15:            Compute FS ← findFS(A[k].setNCF)
16:            if !FS then                       ▷ No feasible FS
17:                Break                         ▷ next k^th SP
18:            end if
19:            p_l.FS ← FS; P ← P + p_l; l ← l + 1
20:            Update A[k].setNCF ← A[k].setNCF − FS
21:        end while
22:        if l == Num then
23:            return P                          ▷ Based on OM[i]
24:        else
25:            Break                             ▷ next k^th SP
26:        end if
27:    end while
28: end while
29: return NoPath
```

Figure 17. Pseudocode for RSA-CR algorithm.

For the sake of completeness, Figure 18 below, also shows the flowchart of the implemented RSA-CR algorithm within the SDN controller.

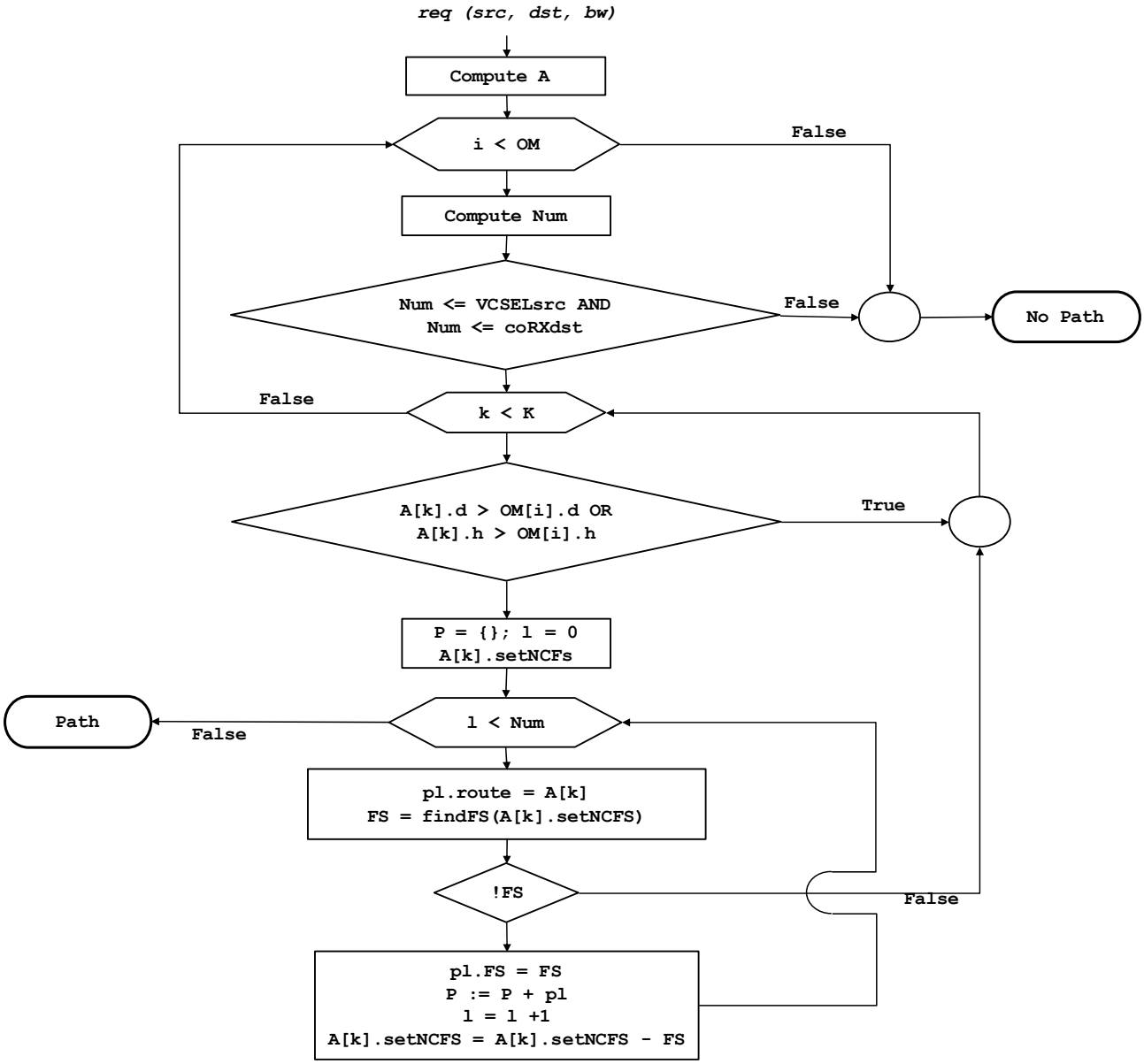

Figure 18. Flowchart of the implemented RSA-CR algorithm

3.2.2 RSA-IM Algorithm

The RSA-IM algorithm is like the RSA-CR mechanism but instead of forcing that all the optical flows for a given *req* are routed over the same spatial path. That is, the optical flows fulfilling a *req* demand not only occupy different FSs but also may be routed over different nodes and links from the src to the dst nodes. Analogously to the RSA-CR algorithm, first it is computed the K-SPs. Then, starting from the highest VCSEL's OM, it is determined the number of VCSELs and optical flows to be allocated. If for the considered VCSEL's OM, no path fulfills the OM's maximum distance and number of hops are discarded, a lower VCSEL's OM is explored. Otherwise, a subset of the K-SPs forming the pool of candidate paths satisfying the under-considered VCSEL's OM requirements is created. This candidate set of paths is then used to accommodate the required optical flows. To do this, for every optical flow, the algorithm iterates over the resulting candidate paths seeking for a feasible FS (i.e., subject to the available S-BVT VCSELs' optical wavelengths and the unused optical spectrum on each path link). If the candidate paths cannot accommodate all the optical flows for *req*, the connection is blocked.

Figure 19 shows the example for the RSA-IM algorithm. As in the previous example, a *req* of 100 Gb/s between S-BVT Tx at node 9 and S-BVT Rx at node 28 needs to be set up; assuming high VCSEL's OM can be used, X and Y optical flows are set up allocating the VCSEL optical carriers at 192.050 THz and 192.250 THz, respectively. X and Y flows are routed over different spatial paths, namely, X flow traverses the path formed by the node sequence 9-26-25-28, whilst Y flow is accommodated over the path 9-26-27-28. The aim of the RSA-IM algorithm when compared to the RSA-CR strategy is to favor fulfilling the spectrum contiguity and continuity constraint mostly within the subnetwork formed by the HL3 all-optical switches.

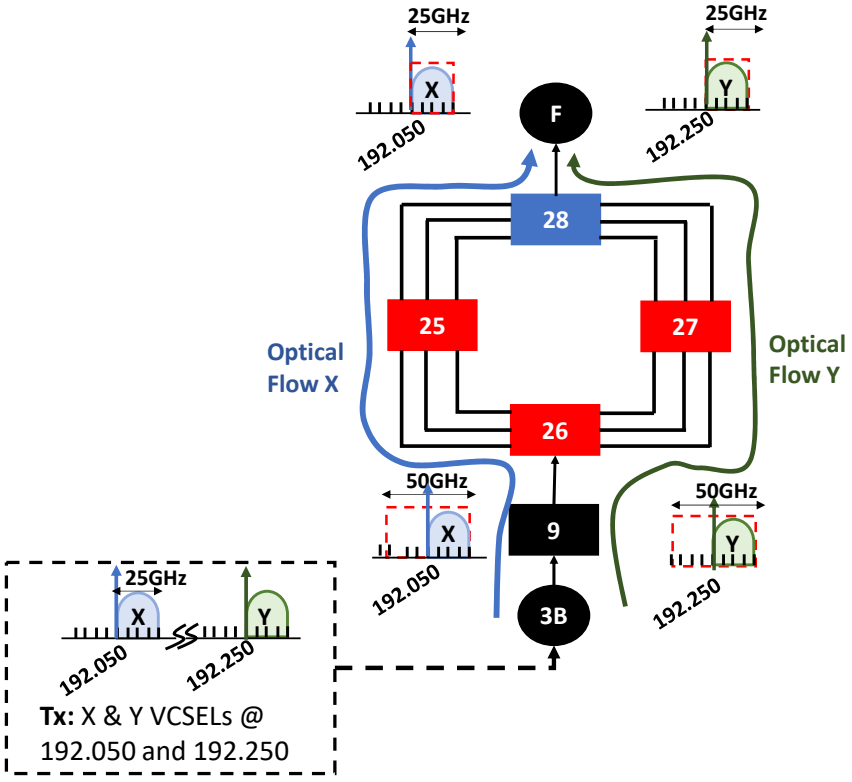

Figure 19. Example of RSA-IM algorithm

The flowchart depicted in Figure 20 describes the RSA-IM algorithm. Upon receiving a *req*, the *A* set with the K-SPs between the src and the dst nodes is computed. Then, the RSA iterates through the VCSEL OMs, i.e., *high, medium,* and low. Like in the RSA-CR, if all the OMs are explored but no solution for *req* is obtained, this is blocked (i.e., NoPath). For an explored *OM[i]*, the *Num* of required VCSELs and CO-Rxs at both S-BVT Tx and Rx is computed to fulfil the *req*'s *bw*. Recall that this *Num* is obtained as the higher integer value above *bw / OM[i].r*. The *Num* also specifies the number of optical flows to be established. If either the available number of VCSELs at src node or the CO-Rxs at the dst node are lower than *Num*, the *req* is blocked.

Conversely, for each optical flow to be computed, it is explored the paths in *A*. For each *A[k]* path, it is checked that both the distance and number of hops do not exceed the maximum permitted values determined by *OM[i]* (i.e., *OM[i].r* and *OM[i].h*). If this occurs, *A[k]* path is discarded, and the pre-computed path in *A* is explored. Otherwise, the unused and common NCFs of the *A[k]* path (i.e., *A[k].setNCF*) is resolved. This entails intersecting: i) the available NCFs along the *A[k]* path links, ii) the associated NCFs of the unused VCSELs at the source node ($vcsel_{src}.NCF$), iii) and those NCFs not being occupied by the used CO-Rxs at the dst node (i.e., $coRx_{dst}.NCF$).

The resulting *A[k].setNCF* becomes the input for the *findFS* function which seeks an available FS. If this succeeds, the FS is associated to the under-considered *A[k]* path (i.e., *p*) as one out of the *Num* optical flows to be computed. Thus, *p* is added to the targeted *P* set (which contains all the *Num* optical flows). Those spectral resources (i.e., NCFs) in FS for *p* are removed from *A[k].setNCF*. Observe that if the *findFS* fails on finding an FS, another *A* path is checked.

Once *P* reaches the targeted *Num* of optical flows, the RSA-IM is successful. If for an optical flow out of *Num*, all the *A* paths are explored but no solution is found, the RSA-IM tries to rely on a lower operational mode OM[i+1] (with lower VCSEL data rate).

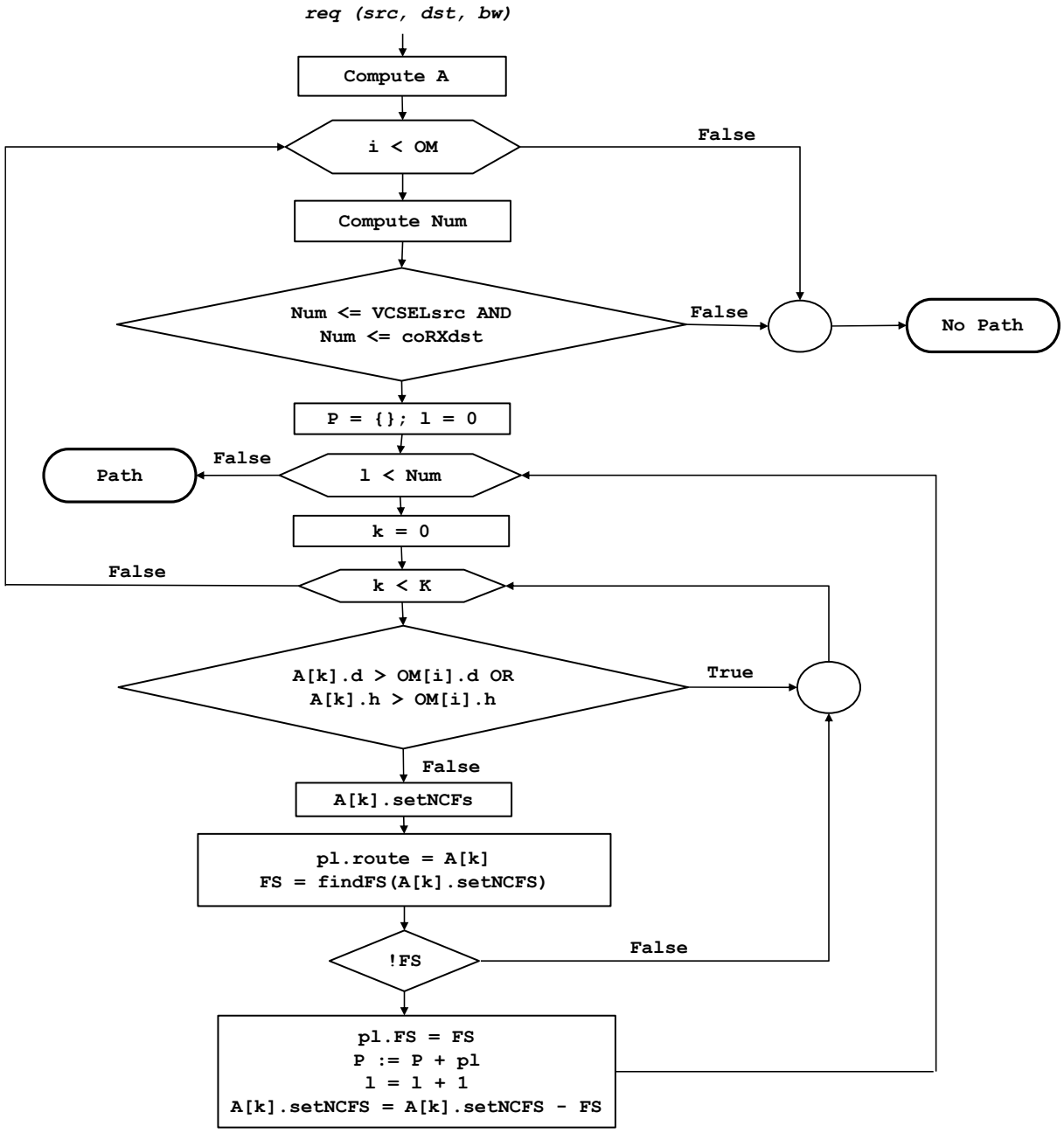

Figure 20. Flowchart of the implemented RSA-IM algorithm

3.3 EXPERIMENTAL VALIDATION AND EVALUATION

The validation and performance evaluation of the SDN controlled metro network and devised RSA-CR and RSA-IM algorithms are conducted at the control plane level within the CTTC ADRENALINE testbed [EUCNC2017_RMunoz]. The PASSION metro network shown in Figure 15 is deployed over 28 Linux-based servers. All the servers have an Intel Core 2 Duo (2.66 GHz) with a RAM of 2Gb. Each server hosts and runs an HLx node agent which logically emulates the (de-)allocation of optical flows. Servers running HL4 agents (i.e., nodes 1 to 24 in Figure 15) also host the agents for controlling S-BVT devices configured as described in Table 1. The server for the node 28 agent (i.e., HL2/HL1) co-locates three fully equipped S-BVT agents. Finally, both the Connection Request Application and SDN controllers run in separated servers.

3.3.1 Workflow Validation

The SDN controller operations (i.e., computation and programmability) triggered upon receiving a *req* are executed according to a defined workflow. Such a workflow (introduced in section 2) is supported through the defined NBI and SBI APIs.

The following provides the validation of the implemented control workflow and the defined RESTful APIs. At step 1 in Figure 21 a *req* arrives to the SDN controller via a `POST /lsp` message from the Connection Request Application. This specifies the *src*, the *dst* and the required *bw*. Next, the RSA (either RSA-CR or RSA-IM algorithms) is triggered. Prior to that, the SDN controller retrieves the status of both the VCSELs and the CO-Rxs at the S-BVT *src* and *dst* endpoints (step 2). This is done by sending to the corresponding S-BVTs' agents both the `GET /sbvtTx` and `GET /sbvtRx` messages. Next, once the RSA algorithm succeeds, the output formed by the spatial path and optical resources (FSs) for all the required optical flows are configured (steps 3 and 4). Specifically, for the S-BVT Tx configuration (step 3), the controller relies on the `POST /sbvtRx/freqSlot` command. The payload of this message carries a set of JSON-encoded parameters (see section 2.2.2) with the central frequency/carrier (*n*) of the selected VCSEL and the slot width (*m*). Analogously, the S-BVT Rx is also programmed via the `POST /sbvtRx/freqSlot`. The contents of this message determine the LO to be tuned (i.e., *n*) at the chosen available CO-Rx element (see section 2.2.3).

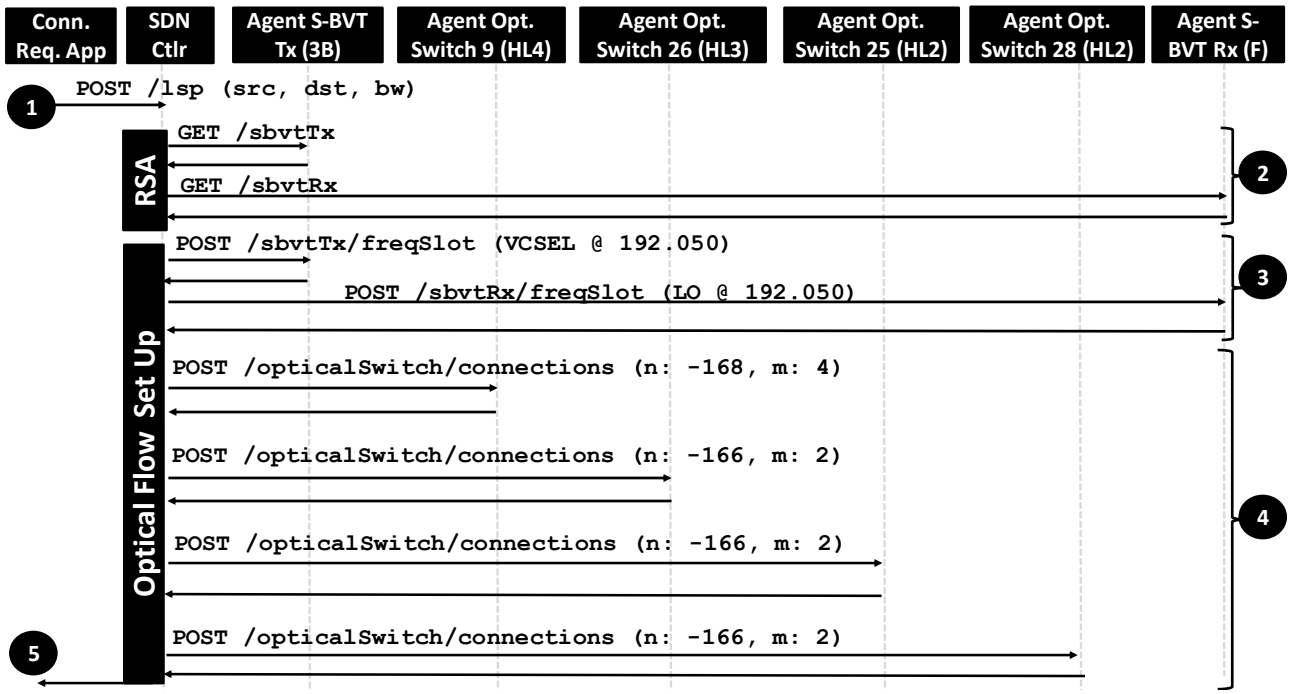

Figure 21. Validation of the workflow SDN controller – SBVT and HL Agents: logical view.

The HL node cross-connections (step 4) are programmed via the `POST /opticalSwitch/connections` command. The message contents provide the spatial and spectral switching information to be configured, namely: *portIn*, *portOut* and *FS (n, m)*. Once all the optical resources are configured, the SDN controller responds to the Connection Request Application (step 5). Figure 22 depicts the experimental validation of the described workflow and control APIs (i.e., NBI and SBI) showing the captured exchanged messages among the SDN controller and the involved network element and devices agents.

	Source	Destination	Info
1	App & SDN Ctlr	App & SDN Ctlr	POST /rest/api/v1/lsp HTTP/1.1 (applicatio
2	App & SDN Ctlr	S-BVT & Node9 Agents	GET /passion/sbi/sbvtTx HTTP/1.1 (applicat
	S-BVT & Node9 Agents	App & SDN Ctlr	HTTP/1.1 200 OK (application/json)
	App & SDN Ctlr	S-BVT Rx & Node128 Agents	GET /passion/sbi/sbvtRx HTTP/1.1 (applicat
	S-BVT Rx & Node128 Agents	App & SDN Ctlr	HTTP/1.1 200 OK (application/json)
3	App & SDN Ctlr	S-BVT & Node9 Agents	POST /passion/sbi/sbvtTx/freqSlot HTTP/1.1
	S-BVT & Node9 Agents	App & SDN Ctlr	HTTP/1.1 201 CREATED (application/json)
	App & SDN Ctlr	S-BVT Rx & Node128 Agents	POST /passion/sbi/sbvtRx/freqSlot HTTP/1.1
	S-BVT Rx & Node128 Agents	App & SDN Ctlr	HTTP/1.1 201 CREATED (application/json)
4	App & SDN Ctlr	S-BVT & Node9 Agents	POST /passion/sbi/opticalSwitch/connections
	S-BVT & Node9 Agents	App & SDN Ctlr	HTTP/1.1 201 CREATED (application/json)
	App & SDN Ctlr	Node126 Agent	POST /passion/sbi/opticalSwitch/connections
	Node126 Agent	App & SDN Ctlr	HTTP/1.1 201 CREATED (application/json)
	App & SDN Ctlr	Node125 Agent	POST /passion/sbi/opticalSwitch/connections
	Node125 Agent	App & SDN Ctlr	HTTP/1.1 201 CREATED (application/json)
	App & SDN Ctlr	S-BVT Rx & Node128 Agents	POST /passion/sbi/opticalSwitch/connections
	S-BVT Rx & Node128 Agents	App & SDN Ctlr	HTTP/1.1 201 CREATED (application/json)
5	App & SDN Ctlr	App & SDN Ctlr	HTTP/1.1 200 OK (application/json)

Figure 22. Validation of the workflow SDN controller – SBVT and HL Agents: RESTful APIs captured messages.

3.3.2 Performance Evaluation

The performance evaluation of the devised RSA-CR and RSA-IM algorithms is done under dynamic connection requests with different data rates (*bw*). Connection requests arrive according to a Poisson process with mean inter-arrival time (IAT) set to 5 s. The connection duration (i.e., holding time, HT) is exponentially modeled with the mean varied to obtain different traffic loads. The *src* and *dst* nodes of each *req* are randomly selected for unidirectional connections between the HL4 nodes and the core (HL2) optical switch (i.e., node 28), and vice-versa. The *bw* is uniformly distributed as multiple of 50 Gb/s up to 200 Gb/s. Each result data point is obtained generating 10k connection requests.

Network links are formed by bidirectional optical fibers. Each fiber supports 644 NCFs spaced 6.25 GHz to span the targeted 191.900 - 195.875 THz spectrum range. The link length (in km) is labeled on each edge shown in Figure 15. This allows computing the distance of candidate paths being essential to the RSA computation at the time of selecting the most suitable VCSEL's OM (Table 2).

The evaluation compares, for different traffic loads, how the RSA algorithms performs when varying the K value of the K-SPs as follows: 1, 3, 6 and 9. The considered figures of merit / metrics are: i) the bandwidth blocked ratio (BBR); ii) the average number of used S-BVTs VCSELs and CO-Rxs; and iii) the average connection setup time. The BBR provides the relationship between the blocked bandwidth and the total requested bandwidth. Thereby, the lower the BBR, the better an RSA algorithm performs.

Figure 23 plots the BBR performance vs HT for the different K values when adopting either the RSA-CR algorithm (Figure 23.a) or the RSA-IM algorithm (Figure 23.b). Regardless of the K value, we observe that as HT increases (i.e., traffic load grows) the BBR performance reduces for both

RSA algorithms. Obviously, more optical flows occupying resources (i.e., S-BVT VCSELs and CO-Rxs, links's NCFs) co-exist. This leads to the RSA algorithm encountering more problems to provide a feasible path and FS for each req's optical flow. Two reasons do cause the RSA algorithm to fail (i.e., NoPath): i) no sufficient available S-BVT resources at either *src* or *dst* for the *Num* optical flows; ii) inability to guarantee the end-to-end spectrum contiguity and continuity constraints for all the optical flows. For the latter, the fact that S-BVT Tx VCSELs are bound to a fixed wavelength, constrains/narrows the candidate set of feasible FS for every optical flow. In other words, the available S-BVT VCSELs at the *src* determine the set of usable optical carriers (i.e., NCFs) for the optical flows's FSs. Bearing this in mind, increasing the number of spatial paths to be considered by both RSA algorithms (i.e., higher K) allows better dealing with the spectrum continuity and contiguity constraints for each required optical flow's FS. Thus, as K increases, the attained BBR improvement (with respect to K=1) does not proportionally grow. The rationale behind this is that increasing K yields to compute longer (in terms of hops and distance) candidate spatial paths. Using such longer spatial paths makes the RSA algorithms rely on lower performance OMs for the S-BVT VCSELs. Consequently, in general the number of required optical flows tends to increase which entails not only requiring more resources to be allocated (at both S-BVT Tx and Rx) but also notably complicates the spectrum constraints for each optical flow.

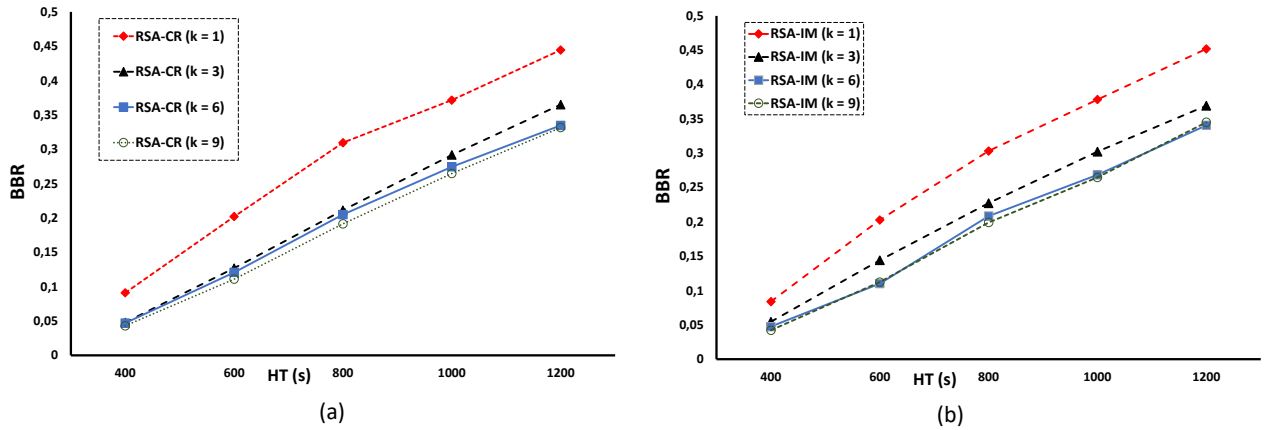

Figure 23. Blocked Bandwidth Ratio (BBR) vs HT (a) RSA-CR and (b) RSA-IM.

Figure 24 shows the comparison of the obtained BBR metric for both RSA algorithms using K set to 1 (Figure 24.a), 6 (Figure 24.b) and 9 (Figure 24.c). We observe that both RSA algorithms, regardless of the K value, perform similarly. Thus, the capability/benefit of the RSA-IM algorithm to route *req*'s optical flows over different paths with the purpose to better fulfil the spectrum constraints does not bring a significant improvement. As expected for K=1, both RSA algorithms attain the same BBR performance since the capability of the RSA-IM to accommodate the *Num* optical flows over diverse paths is indeed not made. On the other hand, for K=6, the RSA-IM algorithm (at low HT) does improve slightly the BBR performance compared to the RSA-CR strategy. Specifically, at such low HT values, the optical resources (i.e., SBVT's VCSELs and optical spectrum) are not heavily loaded. Thus, the RSA-IM algorithm encounters less problems to output feasible paths for the *Num* optical flows satisfying the spectrum constraints over different routes than the RSA-CR where all optical flows must be allocated over the same spatial path. Nevertheless, as HT increases, the optical resources become more occupied: Then, this improvement achieved by the RSA-IM algorithm tends to disappear. Finally, for K=9, again both algorithms provide very similar BBR performance. The amount of candidate paths is large. Therefore, both RSA-CR and RSA-IM tend to provide similar solutions when serving the incoming *reqs.*

A more thorough explanation as to why the RSA-IM does not provide a better BBR performance with respect to RSA-CR algorithm stems from the considered network topology. This network (see Figure 15) only enables route flexibility within the transit ring topology formed by nodes 25, 26, 27 and 28. This considerably narrows the route diversity that could be attained by the RSA-IM algorithm. An interesting follow-up of this work is to explore how the network topology and node connectivity impacts on the performance of both RSA algorithms.

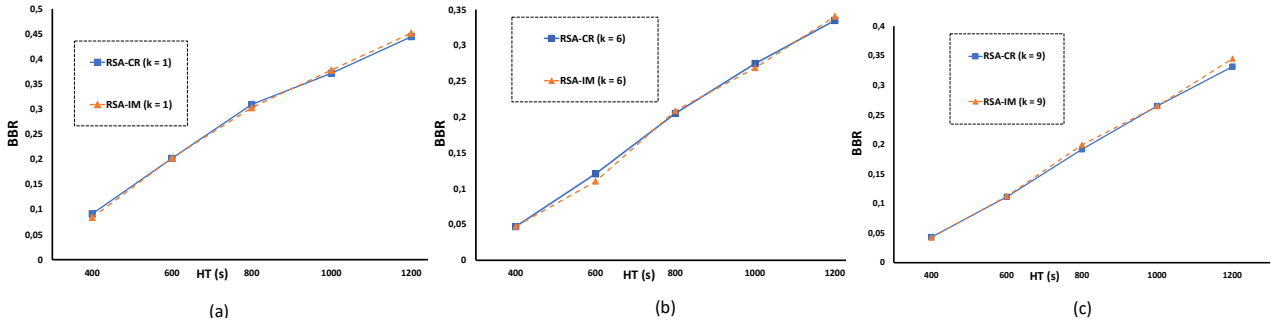

Figure 24. BBR vs HT for both RSA-CR and RSA-IM algorithms using K set to: (a) 1; (b) 6 and (c) 9.

Table 4 provides the average number of S-BVT VCSELs and CO-Rxs usedfor both different HT and K values attained by the considered RSA algorithms. As mentioned previously, a higher K value leads to enhanced BBR performance, i.e. larger *reqs* can be successfully accommodated. Consequently, this does increase the average number average S-BVT VCSELs and CO-Rxs used. As an example, for K=1 adopting the RSA-CR algorithm, the average S-BVT VCSELs and CO-Rx used are 10,2 for HT=400s. For a higher K (i.e., 9) and the same HT=400s, such an average utilization grows to 10,6. Moreover, as the traffic load becomes higher (HT=1200s), the average number ofS-BVT VCSELs and CO-Rxs used increases more significantly. Specifically, for the RSA-CR algorithm, the average utilization of the S-BVTs at HT=1200s ranges 16,1 – 16,2 at K=1; whilst at K=9, this average utilization metric reaches 19,2. Recall that as shown in Table 2, at the HL4 nodes the number of S-BVT VCSELs and CO-Rxs is 20. Thus, at K=9 and HT=1200s most of the S-BVT resources are averagely occupied. Similar trend is accomplished when the RSA-IM algorithm is applied. In general, increasing the K value, allows the algorithm to explore a larger number of candidate paths. Thereby, this provides higher spatial path flexibility which allows satisfying feasible *req*'s optical flows and fulfilling spectrum constraints. As a result, more concurrent connections co-exist within the network leading to increase the average used of S-BVT VCSEL and CO-Rx devices.

Table 4 Av. Results for the Average Used of VCSELs and CO-Rx and Average Setup Time

	RSA-CR		RSA-IM		RSA-CR		RSA-IM	
K	**1**				**9**			
HT	400	1200	400	1200	400	1200	400	1200
Av. Used VCSEL	10,2	16,1	10	16	10,6	19,2	10,6	19,2
Av. Used Co-Rx	10,2	16,3	10	16,1	10,6	19,2	10,6	19,2
Av. Setup Time (ms)	83,5	79,7	93,2	90,4	88,6	87,1	99,2	96,7

The above improvement in both BBR performance and S-BVT resource utilization when increasing the K value (in RSA-CR and RSA-IM) is attained at the expenses of slightly increasing the average setup time as seen in Table 4. That is, the larger is the number of K pre-computed spatial paths, the higher the RSA computation time is. Additionally, in the RSA-IM algorithm, we observe that regardless of the HT, the average setup time is larger than those results attained by the RSA-CR algorithm. The reason of this is that in the RSA-IM algorithm for each req's optical flow, all the pre-computed K-SPs are checked. This, however, does not happen in the RSA-CR algorithm where for

all req's optical flow only a single K-SP is checked for each iteration. Consequently, the RSA-IM algorithm tends to search over more candidate K-SPs when computing each req, resulting in a larger computational time, and in turn, higher average setup time.

4 CONCLUSIONS

This deliverable provides the validation of the SDN controller designed in the context of WP2 and reported in [D2.2]. The SDN controller integrates the key functionalities needed to manage and orchestrate all the PASSION devices, implementing the required flexibility and dynamicity of the PASSION use cases, with emphasis given to the *pay-as-you-grow* scenario.

The NorthBound Interface (NBI), based on RESTful API messages, is described in detail and the message exchange tracing is reported to verify the compliance with respect to the specifications. Furthermore, a brief overview of possible extended NBI capabilities, leveraging an SDN hierarchical approach, has been presented as a possible evolution of the PASSION project, not addressed here. SouthBound Interface (SBI) is also based on RESTful API and the same validation process has been applied. Further extensions to the SBI SBVT Tx will be tackled in upcoming activities to cover advanced functionalities such as the support of a Digital Signal Processing (DSP) system in the S-BVT Tx element. This allows the use of adaptive algorithms for multicarrier modulations as described in [JLT2019].

A relevant part of this deliverable is devoted on the verification of the devised RSA algorithms applied to a complex network scenario. Performance evaluation is based on bandwidth blocked ratio (BBR) figure, which is estimated by extensive emulations for different traffic load situations. The average number of transmitters and receivers used, and the average setup time are also reported.

The SDN Controller is now available for the next integration steps, when the agent applications related to PASSION building blocks are ready to be tested in the final phase of the project (i.e. T5.4 and T5.5).

5 REFERENCES

[D2.2] “Overall network architecture and control aspects definition”, PASSION D2.2, January 2020.

[ITU-T_G694.1] Recommendation ITU-T G.694.1, “Spectral grids for WDM applications: DWDM frequency grid”, February 2012.

[ONDM2020_RMartinez] R. Martínez, et. al., “Experimental Evaluation of an On-line RSA Algorithm for SDN-Controlled Optical Metro Networks with VCSEL-based S-BVTs”, accepted as oral presentation in 24th International Conference on Optical Network Design and Modelling (ONDM), Castelldefels, Spain, May 2020.

[BROADNETS2020_RMartinez] R. Martínez, et. al., “Experimental Evaluation of RSA Algorithms for SDN programmable VCSEL-based S-BVT in High-Capacity and Cost-Efficient Optical Metro Networks”, submitted in Workshop Go2Edge in EAI BROADNETS 2020

[EUCNC2017_RMunoz] R. Muñoz, et. al., “The ADRENALINE Testbed: An SDN/NFV Packet/Optical Transport Network and Edge/Core Cloud Platform for End-to-End 5G and IoT Services,” In Proc., of European Conference on Networks and Communications (EuCNC), June 2017.

[JLT2019] M. Svaluto Moreolo, et. al., "Synergy of Photonic Technologies and Software-Defined Networking in the Hyperconnectivity Era," IEEE/OSA Journal of Lightwave Technology, Special issue “Photonic Networks and Devices,” Vol. 37, No. 16, pp. 3902 - 3910, May 2019.

6 ACRONYMS

AWG	Arrayed waveguide Gratings
BBR	Bandwidth Blocked Ratio
BVT	Bandwidth-Variable Transceiver
CO-Rx	Coherent Receiver
DSP	Digital Signal Processing
ERO	Explicit Route Object
FS	Frequency Slot
HLn	Hierarchy Level n
HT	Holding Time
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LO	Local Oscillator
MAN	Metropolitan Area Network
NBI	Northbound Interface
NCF	Nominal Central Frequency
REST	Representational State Transfer
RSA	Routing and Spectrum Assignment
SBI	Southbound Interface
SBVT	Sliceable Bandwidth-Variable Transceiver
SDN	Software Defined Networking
TED	Traffic Engineering Database
VCSEL	Vertical-Cavity Surface-Emitting Laser
WDM	Wavelength-Division Multiplexing
WSS	Wavelength Selective Switch
XML	Extensible Markup Language