Grant Agreement No. 619572

COSIGN

Combining Optics and SDN In next Generation data centre Networks

Programme: Information and Communication Technologies

Funding scheme: Collaborative Project – Large-Scale Integrating Project

Deliverable D4.3 – COSIGN orchestrator interaction with the COSIGN SDN controller platform

Due date of deliverable: December 31, 2015
Actual submission date: December 22, 2015

Start date of project: January 1, 2014 Duration: 36 months

Lead contractor for this deliverable: UPC

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Executive Summary

The current document comprises the next major step of WP4 in the design of the COSIGN orchestrator layer. While previous deliverable D4.2 "COSIGN orchestrator low level architecture and prototype design" [D4.2] focused on the specification of the orchestrator low level architecture design, with the identification of OpenStack (OS) as the reference platform for the development of the orchestrator, the current deliverable focuses on the interaction between the COSIGN orchestrator layer and the control layer, the specification of the control layer clients and the extended operational flows for all the identified use cases in COSIGN.

The document is organized as follows:

Section 2 gives a brief reminder of COSIGN's control layer and its roles.

Section 3 elaborates on the interactions between the orchestrator and the control layer. In particular, it specifies the concrete requirements in terms of needed actions to be triggered at the control layer for each one of the COSIGN use cases, tapping on the capabilities exposed by the northbound interface of the Software Defined Networking (SDN) controller defined in deliverable D3.2 "SDN framework northbound and southbound interfaces specification" [D3.2].

Section 4 specifies the control layer client modules of the COSIGN orchestrator. In this regard, two clients are defined, namely, the Open Virtual Network client, for supporting the control of overlay virtual networks, and the OpenDaylight client, for supporting the configuration of the physical network.

Section 5 presents the operational workflows for each one of COSIGN use cases, showing how the major scenarios required by the use cases are realized through the interaction between the orchestrator upper layers and the infrastructure control clients' layer.

Finally, Section 6 concludes the deliverable by summarizing its goals as part of WP4 progress and as a contribution to the overall project outcomes.

Document Information

Status and Version:	version 6 Final	
Date of Issue:	22/12/2015	
Dissemination level:	Public	
Author(s):	**Name**	**Partner**
	Albert Pagès	UPC
	Salvatore Spadaro	UPC
	Fernando Agraz	UPC
	Jordi Perelló	UPC
	Katherine Barabash	IBM
	Yaniv Ben-Itzhak	IBM
	José Aznar	I2CAT
	Amaia Legarrea	I2CAT
	Cosmin Caba	DTU
	José Soler	DTU
	Giada Landi	NXW
	Chris Jackson	UNIVBRIS
	Alessandro Predieri	IRT
	Domenico Gallico	IRT
Edited by:	Albert Pagès	
Reviewed by:	Giada Landi	NXW
	José Aznar	I2CAT
Checked by :	Sarah Ruepp	DTU

Table of Contents

1 Introduction

1.1 Reference Material

1.1.1 Reference Documents

[D1.1]	COSIGN Deliverable D1.1: Requirements for Next Generation intra DCN Design
[D3.1]	COSIGN Deliverable D3.1: SDN framework functional architecture
[D3.2]	COSIGN Deliverable D3.2: SDN framework northbound and southbound interfaces specification
[D4.2]	COSIGN Deliverable D4.2: COSIGN orchestrator low level architecture and prototype design

1.1.2 Acronyms and Abbreviations

Most frequently used acronyms in the Deliverable are listed below. Additional acronyms may be defined and used throughout the text.

API	Application Programming Interface
BW	Bandwidth
CMS	Cloud Management System
CoS	Class of Service
CRUD	Create, Read, Update and Delete
DB	Database
DC	Data Centre
DCN	Data Centre Network
DHCP	Dynamic Host Configuration Protocol
ML2	Modular Layer 2
NBI	Northbound Interface
ODL	OpenDaylight
OF	Open Flow
OS	OpenStack
OVN	Open Virtual Network
OVS	Open Virtual Switch
P2MP	Point-to-Multi-Point
P2P	Point-to-Point
QoS	Quality of Service
REST	Representational State Transfer
SDN	Software Defined Networking
vApp	Virtualized Cloud Application
VDC	Virtual Data Centre
VTN	Virtual Tenant Network

1.2 Document History

Version	Date	Authors	Comment
1.0	23/10/2015	See authors list	Load initial ToC into the template
1.5	19/11/2015		Final ToC version of the document with assignments
1.6	21/11/2015		Integrated partners contributions
2.0	27/11/2015		First integrated version
2.1	01/12/2015		Addressed IRT, NXW and DTU comments
2.2	02/12/2015		Integrated Bristol contribution to section 5.1.1. Refinements to section 3.1. Integrated IBM contribution
3.0	03/12/2015		Second integrated version

3.1	10/12/2015		Refinements to sections 3 and 4
3.5	14/12/2015		Internal review
4	15/12/2015		Integrated reviewers' suggested changes
5	21/12/2015		Final version for quality check
6	22/12/2015		Submitted version

2 COSIGN Control Layer – Summary

One of the key points that the COSIGN project is tackling in order to satisfy the requirements of future Data Centre (DC) infrastructures is the development of novel optical technologies for the intra-DC Network (DCN) to achieve the necessary bandwidth, latency and high re-configurability that intra-DCN communications require as it has been already identified in deliverable D1.1 "Requirements for Next Generation intra DCN Design" [D1.1]. The development of the related data plane technologies to satisfy the infrastructural needs of the DCN is being done in WP2. In order to fully exploit all the capabilities available at the network substrate, programmatic control of the DCN must be exerted. Such goal is being pursued in the framework of WP3, focused on the development of the control layer infrastructure.

To this goal, the control layer in the COSIGN project has to fulfil the following roles:

- Configuration of the data plane (i.e. the physical DCN infrastructure).
- Provide network intelligence.
- Enable the virtualization of the optical network infrastructure.

To match the above roles, the control plane is devoted to providing the necessary procedures for the dynamic establishment of connectivity in the DCN, which will be later on employed in the upper layers to realize the different COSIGN use cases as examples of services. As for the virtualization of the optical network infrastructure, this is achieved by exposing towards upper layers abstracted representations of the underlying optical data plane, hiding the complexity and the particularities of the specific technological solutions at the data plane. In this way, simpler configuration of the network infrastructure can be achieved.

Here, SDN concept is embraced [1]. The SDN paradigm pursues automatic control and programmability of the network resources. The network intelligence is moved from the data plane to the SDN controller and its applications, effectively decoupling data and control layers. Then, all the intelligence of the networks are kept in a centralized entity, i.e. the SDN controller, which has a global view of the network and acts according to this information to achieve fine control and programmability of the network fabric. Two main interfaces are defined at the SDN controller level: a southbound interface, to enable the communication between data and control planes, and a northbound interface, for the communication between the control and the application layers. Thanks to SDN, it is possible to modify the behaviour of the network, simplify the network configuration and operation, achieve vendor-independent control over the network, and ease the management of the network devices.

Due to these characteristics, SDN has been chosen as a framework for the development and deployment of the COSIGN control plane, since it provides an appropriated architectural design to implement the required functionalities in terms of configuration, management and monitoring. In this regard, deliverable D3.1 "SDN framework functional architecture" [D3.1] surveyed available open source SDN controller software with the objective to select the most suitable candidate to fill the role of the COSIGN SDN controller, putting a strong accent on the flexibility and capabilities that the platform of choice provides.

As shown in Figure 1, the final architectural decision on the control plane was to distribute its functionalities into two different SDN controllers, namely, Open Virtual Network (OVN) and OpenDaylight (ODL). First, the OVN controller is responsible for the control of the logical overlay networks, while at its turn, the ODL controller focuses on the control of the infrastructure network through the use of the OpenFlow (OF) protocol, for which COSIGN is developing novel extensions for the control of optical devices.

The combination of both controllers entail several benefits, which are highly desirable for the realization of the COSIGN control plane. On the one hand, overlay networks allow achieving isolation between tenants and independence between logical domains. On the other hand, direct programmability of the optical infrastructure is a must to ensure an optimized physical network

utilization and provides a proper service differentiation as required by the applications that must be deployed over the DCN. Thus, dedicated controllers are being utilized for the efficient control of both aspects of the network infrastructure.

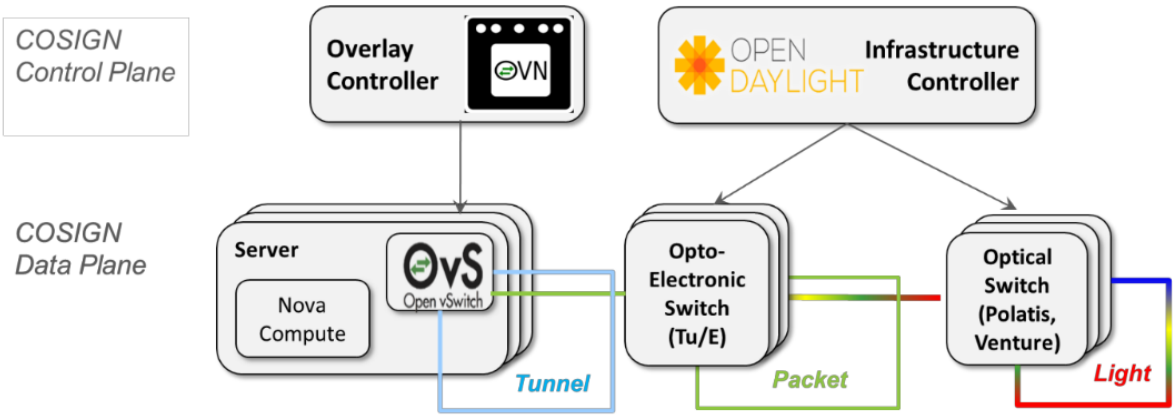

Figure 1 – COSIGN SDN control plane architecture.

Given this framework, this document will expand upon the work already reported in deliverables [D3.2] and [D4.2], providing the details of the interactions between the orchestrator layer and the control plane layer as well as the specifications of the control layer clients at the orchestrator that enable the cross-stratum communication.

3 Interaction between COSIGN Orchestrator and Control Layer

The COSIGN orchestrator is the main responsible for the realization of the three major use cases identified in the project:

- The Virtual Data Centre (VDC) use case, where the user is an external entity (tenant) requesting to the DC owner for the provisioning of a dedicated virtual infrastructure consisting of both compute and network resources and with certain control over some configuration options with direct impact on the physical layer of the provisioned slice so that the tenant can deploy his own programmable services on top of the rented infrastructure.
- The Virtualized Cloud Application (vApp) use case, where the user is an application/service operator focused only on the operational aspects of the provisioned virtual infrastructure.
- The DC Operations and Management (DC O&M) use case, where the user is the DC owner/administrator and seeks to optimize the utilization of the DC infrastructure by tapping on the rich and advanced monitoring and joint orchestration capabilities offered by orchestrator platform.

While the whole coordination of the provisioning of both compute and network resources is a direct responsibility of the orchestrator, achieving the configuration of the network fabric is the responsibility of the network control plane. Previous deliverable [D4.2] described at a high level the required interactions between the orchestrator and the control planes from the perspective of each one of the use cases. For interacting with the DC resources, the orchestrator uses the infrastructure management clients – Nova for compute and Neutron for DCN. In COSIGN, Neutron service will be extended so it has access to two different SDN controllers – the OVN as a Neutron plugin providing the OS virtual networking APIs and the ODL as an extension for managing the optical infrastructure. In what follows, a more detailed description of the interactions between the two layers is presented. For this, the previously defined interactions and requirements are mapped to the specific actions enabled by the SDN controller Northbound Interface (NBI) [D3.2].

3.1 VDC Use Case – Requirements and Mapping to SDN Controller

Section 4.1.4 of deliverable [D4.2] already described the infrastructure-driven interactions and requirements that the VDC use case imposes towards the SDN controller, its NBI and services, which constitute the relevant ones for this section. Thus, the main interactions and requirements towards the SDN controller are:

- Advertisement of network resource utilisation and availability
- Continuous monitoring and updates
- Explicit resource allocation and slicing
- VDC network isolation and QoS guarantees

The matching among these interactions and requirements with the options enabled by the COSIGN controller NBI is fundamental to ensure the proper behavior of the VDC use case operations. Thus, in this section we go deeper into the details on how previously enounced requirements can be mapped to the interfaces and services facilitated from the control plane. For the purpose of the discussion, the primary actors and entities involved on the VDC use case are:

- The VDC service provider, which must orchestrate requests for VDC instances coming from external tenants.
- The VDC service instance, which is a virtual infrastructure realizing a VDC client request.
- The VDC instance owner. Each VDC owner (i.e. tenant) must be able to access and configure all the virtual resources belonging to its own VDC instance through programmable,

Representational State Transfer (REST)-enabled Application Programming Interfaces (APIs). VDC instances belonging to different VDC owners must be properly isolated and the performance of each owner's traffic must be not impacted by the whole traffic at the physical infrastructure.

In general, the VDC provider and service must be able to see the underlying physical resources. They must be able to provision and monitor these physical resources. In addition, they should be able to monitor statistics and activity of individual VDC instances. The VDC owner and instance must be restricted so that any programming of network behaviour is limited to only affect its VDC instance. Based on the monitoring of the whole physical resources, the orchestrator can optimally decide the placement of a new VDC instance and trigger the configuration of both compute and network resources, relying on the entities that are responsible for these actions: Nova for the compute and Neutron plus ODL for the network. Figure 2 depicts a schematic of the whole interaction for the request of a new VDC instance.

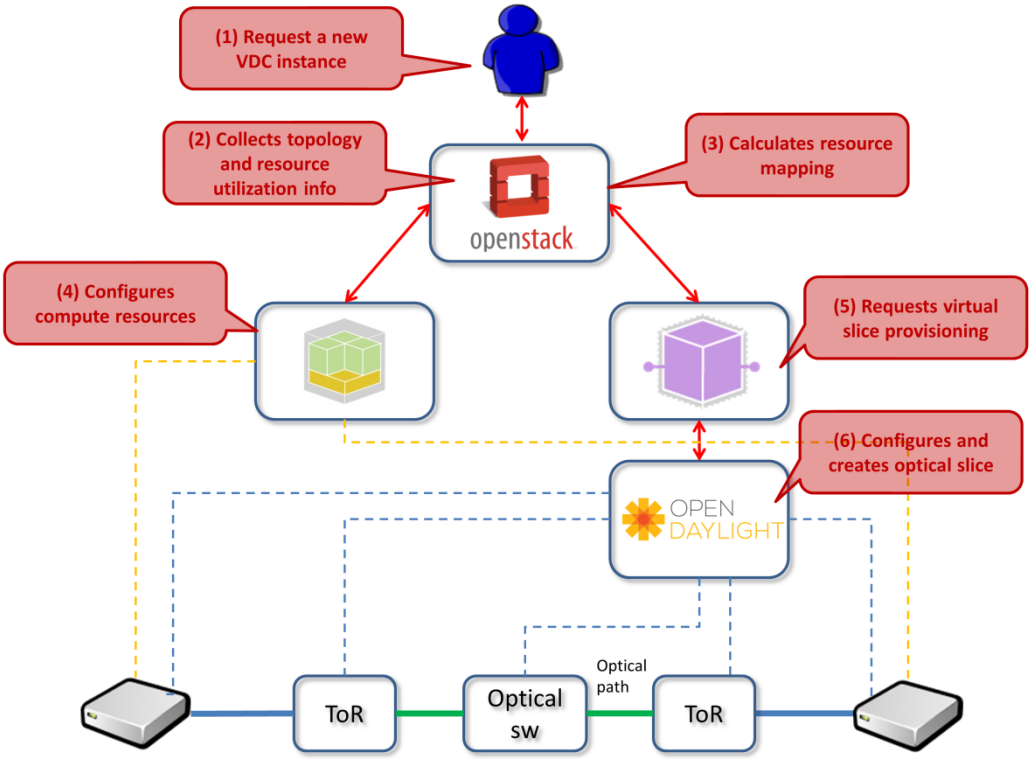

Figure 2 – New VDC creation: Interaction between orchestrator and control layers.

The following sub-sections detail for each of the identified requirements the actions that have to be triggered towards the SDN controller, the involved SDN controller modules/services, the information that can be retrieved/modified and how each of the previously defined entities are able to utilize the services.

3.1.1 Resource Utilisation and Availability

In order to allocate the desired resources to fulfil requests from a VDC owner, physical resource utilisation information must be available to the VDC provider/service. In addition, the topology of the physical network as well as an inventory of devices must be retrievable in order to compute the logical networks that may be supported. Table 1 summarizes the main interactions between the orchestrator and the SDN controller to achieve this purpose.

Table 1 – Summary of interaction between the orchestrator and the SDN controller for DCN and VDC instance resource utilization and availability.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**VDC provider**	**VDC client**
-DCN Orchestrator Network device utilisation	-Flow Statistics URI POST/restconf/operations/opendaylight-flow-statistics:get-all-flows-statistics-from-all-flow-tables -Port Statistics URI POST/restconf/operations/opendaylight-port-statistics:get-all-node-connectors-statistics	Lists flows currently installed in each device (flow table) Flow: -Packet count -Duration -Byte count Port: -Drops -Byte count -Packet Count	Complete view of all flows and ports statistics on all network devices.	Should only see its own flows and port statistics.
-DCN Orchestrator Network device inventory and topology	-Inventory Manager, Topology Manager URI POST/restconf/config/opendaylight-inventory:nodes -Virtual Infrastructure Manager URI GET/restconf/config/virtual-infrastructure-manager:slice/<Tenant ID>/<SliceID>	Device list detailing: -Type -Capabilities -Ports -Etc.	Uses inventory and topology for orchestration and allocation algorithms. Can view full physical network topology.	Should be provided an inventory consisting only of allocated devices and slice.

3.1.2 Continuous Monitoring and Updates

The orchestrator must be able to track runtime information of physical network devices. This will enable the monitoring of resource utilisation and the detection of conditions where service recovery or VDC re-planning is required. Depending on configured policies, service recovery may happen at the network-only level or at the orchestrator level. In the former case the network slice is reconfigured but the VMs are kept in the previous locations. The whole procedure can be performed in the SDN controller, without the active role of the orchestrator that is just notified of the changes. This approach can lead to sub-optimal solutions in terms of resource allocation, but it reduces the service recovery time.

In the latter case, the physical network failure is notified to the orchestrator, where the VDC algorithms compute an alternative solution for the entire VDC instance, including a different placement of VMs. Therefore, this approach requires the migration of VMs and longer recovery time. Similarly, the VDC client should be able to track the utilisation of its own virtual network. If service recovery operations occur, ideally they are transparent to the client. However, they should still receive a notification or report in case the recovery has interfered with service.

Table 2 summarizes the main interactions between the orchestrator and the SDN controller to achieve this purpose.

Table 2 – Summary of interaction between the orchestrator and the SDN controller for VDC continuous monitoring and updates.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**VDC provider**	**VDC client**
-DCN Orchestrator Monitor device utilisation, status and notifications about failures in the physical equipment	-Flow Statistics URI POST/restconf/operations/opendaylight-flow-statistics:get-all-flows-statistics-from-all-flow-tables -Port Statistics URI POST/restconf/operations/opendaylight-port-statistics:get-all-node-connectors-statistics	Lists flows currently installed in each device (flow table) Flow: -Packet count -Duration -Byte count Port: -Drops -Byte count -Packet Count	Must be able to view all traffic and device statistics and information.	View limited to allocated devices and slice.
-Neutron Move VDC instance to new devices. Update deployed virtual slice.	-Virtual Infrastructure Manager URI PUT/restconf/config/virtual-infrastructure-manager:slice/<TenantID>/<SliceID>	Slice: -TenantID -SliceID -New request details (topology, resources)	Can create update/move any virtual slice deployed in the physical network	Limited to programming its allocated virtual slice.

3.1.3 Explicit Resource Allocation and Slicing

After the orchestrator has determined which resources should be allocated to a client to fulfil a request, it is necessary to reserve those resources. With respect to network devices, it is necessary to program flows that will determine their behaviour and set up the VDC. For this purpose, the orchestrator has to notify to the SDN controller the explicit resources to be configured. Within the VDC, it may be necessary for the client to program their own flows onto the devices that have been allocated and exposed for the given VDC instance.

Table 3 summarizes the main interactions between the orchestrator and the SDN controller to achieve this purpose.

Table 3 – Summary of interaction between the orchestrator and the SDN controller for VDC explicit resource allocation and slicing.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**VDC provider**	**VDC client**
-Neutron Enforce mapping between virtual and physical resources to establish a VDC instance	-Virtual Infrastructure Manager URI POST/restconf/config/virtual-infrastructure-manager:slice	Slice: -TenantID -SliceID -Request details (topology, resources)	Must be able to request any mapping between virtual and physical devices to construct VDC instances.	NA
-Neutron Eliminate a deployed VDC instance	-Virtual Infrastructure Manager URI DELETE /restconf/config/virtual-infrastructure-manager:slice	Slice: -TenantID -SliceID	Can eliminate any VDC instance tagged by <SliceID> from any tenant.	Can only request for the elimination of the VDC instances tagged under his <TenantID>.

3.1.4 VDC Network Isolation and QoS Guarantees

To guarantee the isolation of tenants with concurrent VDCs, both compute and network resources of different VDCs should not interfere with one another. For the network this ensures that strict Quality of Service (QoS) guarantees should be met. The VDC service must ensure that any programming of the virtual network infrastructure affects only the VDC for which the modification request originated. Thus, a service must sit between each VDC and the SDN controller to validate any flow creation or modifications, effectively providing a virtual SDN controller.

Table 4 summarizes the main interactions between the orchestrator and the SDN controller to achieve this purpose.

Table 4 – Summary of interaction between the orchestrator and the SDN controller for VDC network isolation and QoS guarantees

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**VDC provider**	**VDC client**
-Neutron Flows to program virtual network instances	-Virtual Tenant Network Manager Ensure that a flow will affect only the traffic of the source VDC by creation of proper filters in the	Flow modification messages: -Pass -Drop -Redirect	Must validate VDC flow programming requests	Must be able to install flows to manage own virtual network infrastructure

	corresponding Virtual Tenant Network (VTN) instance			

3.2 vApp Use Case – Requirements and Mapping to SDN Controller

Section 4.2.4 of deliverable [D4.2] already described the infrastructure-driven interactions and requirements that the vApp use case imposes towards the control plane layer. The primary entities that are involved in the vApp use case are:

- The vApp provider and service, which must orchestrate requests for vApp instances coming from external tenants. vApp instance request mostly includes VMs and virtual network.
- The vApp owner and instance. Each vApp owner (i.e. tenant) must be able to access and configure all the virtual resources belong to its own vApp instance through programmable, Representational State Transfer (REST)-enabled Application Programming Interfaces (APIs). vApp instances belong to different vApp owners must be properly isolated.

vApp use-case provides network performance enhancement, by detecting elephant flow, and other flow scenarios, which can be transferred though an optical circuit path. To that end, vApp use case includes the following modules (see also Figure 3) : 1) *virtual observer* for tagging elephant flows; (2) *physical observer* to manage the optical connectivity for the tagged flows; and (3) *orchestrator algorithms* to setup and manage these entities (e.g,. flows QoS, flow tag constants, available optical circuit to be used, etc).Some of the *orchestrator algorithms* setup is required through the cloud instantiation (e.g., flow tag constants), while others can be configured during the cloud operation (e.g., flows QoS, and available optical circuit to be used).

The vApp use case involves dynamic circuit establishment for prioritized treatment of traffic classified at run time using data plane monitoring, as it will be described below. The Virtual Network Controller, OVN controller, participates in flow classification and in subsequent flow discrimination, under the coordination of the COSIGN Orchestration layer.

For flow classification, the capabilities of the OVN controller are exploited in order to tag the classified flows detected by the sFlow[1] collector connected to OVS switches. Figure 4 exemplifies these actions regarding the tagging of an elephant flow by the OVN controller. The *virtual observer* includes a sFlow collector for sampling traffic flowing over the ovs-switch of each compute node (step 1 of Figure 4). Then, the sFlow analyser included in a virtual observer classifies flows and detects flows to be discriminated (step 2 of Figure 4).

Although simple classifications, e.g. the depicted elephant flows distinguished by the amount of data sent as part of the flow in a time slot, can be classified autonomously at the SDN controller layer, a more complex flow classification requires looping in the Orchestrator component, e.g. for including management level and application level considerations. As traffic flows are classified, new OVN logical flows are set into the SouthBound database (DB) of OVN, e.g. in order to apply DSCP[2] marking rule to the corresponding traffic flow (step 3 of Figure 4). OVN logical flows are then pushed into the OVN controller over the compute nodes (step 4 of Figure 4) and translated into OF rules in the vswitch datapaths in local host kernels (step 5 of Figure 4). As a result, the detected elephant flow is marked to be later discriminated in the network, e.g. using pre-defined DSCP values (step 6 of Figure 4).

For flow discrimination, traffic is monitored at the opto-electronic switches connected directly to the optical switch, by the *physical observer*. The *physical observer* detects elephant flows by their pre-defined DSCP value introduced at classification stage. Then, the physical observer decides whether to use the optical circuits to send the marked flows. The decision, again, can be made either

[1] http://www.inmon.com/technology/

[2] DSCP (Differentiated Services Code Point) is 6-bit field in the IP header for packet classification purposes. The DSCP values are used to identify both the QoS of the flow and the type of the flow (e.g., mice, elephant, short, long, etc ...).

autonomously by the SDN controller (Infrastructure Controller) or by looping in the orchestrator. The decision is made based on the knowledge of the current topology existing optical circuits, load over physical paths, etc. The per-flow decision can be either to send the classified flow over an existing path, optical, electronic, or hybrid, or to create a new circuit to send the classified flow over. In the latter case, the Physical Network Controller creates the new circuit, using the ODL APIs exposed by the Optical Provisioning Service. In addition, the Physical Network Controller configures network switches on the flow's path so that the flow is transmitted according to the forwarding decision, e.g. through the newly create optical circuit to enjoy better network performance, in terms of both latency and bandwidth.

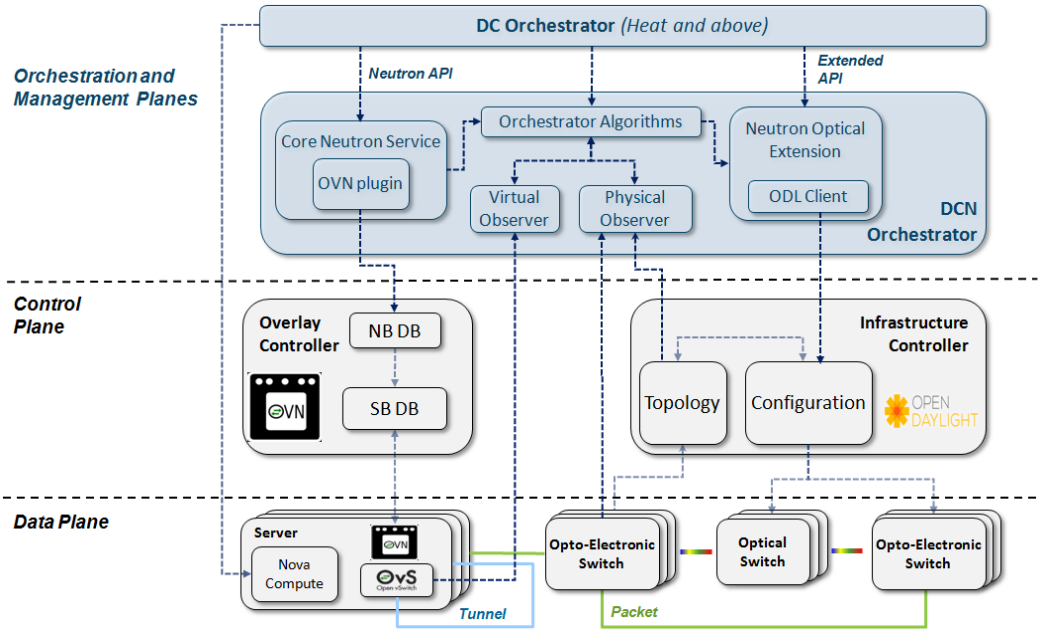

Figure 3 – vApp Orchestration, Control and Data Plane Architecture

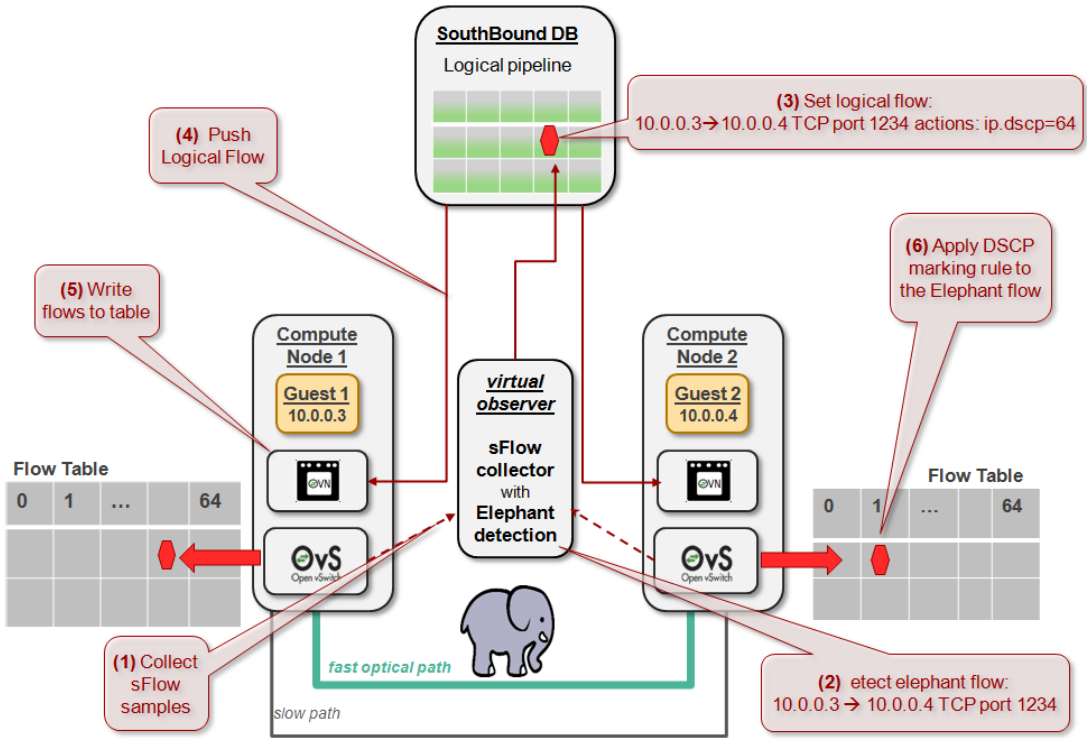

Figure 4 – Elephant flow tagging by OVN.

With this explanation, the main interactions and requirements towards the control layer are:

- Physical network infrastructure monitoring
- Optical connection provisioning
- Virtual networks provisioning

With these, the following sub-sections detail for each of the identified requirements the actions that have to be triggered towards the SDN controller, the involved SDN controller modules/services, the information that can be retrieved/modified and the role of the physical observer.

3.2.1 Physical Network Infrastructure Monitoring

The physical observer has to gain monitoring rights over the physical optical infrastructure in order to decide the most proper optical connectivity to be established once a flow has been classified in the sFlow collector. Table 5 summarizes the main interactions between the orchestrator and the SDN controller to achieve this purpose.

Table 5 – Summary of interaction between the orchestrator and the SDN controller for vApp physical network infrastructure monitoring.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**Physical observer**
-Physical observer Physical topology exploration	-Inventory Manager, Topology Manager URI GET/restconf/config/opendaylight-inventory:nodes	Device list detailing: -Type -Capabilities -Ports -Etc.	The physical observer retrieves all the nodes available in the network, each of them with the list of its ports.
-Physical observer Traffic monitoring at the physical network	-Port statistics URI GET/ restconf/operations/opendaylight-port-statistics:get-all-node-connectors-statistics -Flow statistics URI POST/ /restconf/operations/opendaylight-flow-statistics:get-all-flows-statistics-from-all-flow-tables	List of ports. - Port ID -Port Type -Port Stats List of Flows -Match -List -Flow Stats List of Tables -Table ID -Table Stats List of Queues -Queue ID -Queue Stats	The physical observer has to be able to retrieve statistics about all the ports in a node, the flows at the ports, the flow tables at the node and the queues at the ports.

	-Tables statistics URI POST/restconf/operations/opendaylight-flow-table-statistics:get-flow-tables-statistics -Queues statistics URI POST/restconf/operations/opendaylight-queue-statistics:get-all-queues-statistics-from-given-port		

3.2.2 Optical Connection Provisioning

Once a flow has been classified, an optical connection may be required to send the related traffic over the established lightpath. To this end, in the vApp use case, it must be possible to request the provisioning of optical connectivity services as it fits to the application requirements. The main requester/consumer of this service is the physical observer which interacts with the underlying Physical Infrastructure Controller to achieve this purpose. Table 6 summarizes the main interactions between the orchestrator and the SDN controller to achieve this purpose.

Table 6 – Summary of interaction between the orchestrator and the SDN controller for vApp optical connection provisioning.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**Physical observer**
-Physical observer Creation of an optical connection	-Optical provisioning manager URI POST/restconf/config/optical-provisioning-manager:connection	Optical connection: -Type -Source -Destination -Bidirectional -QoS parameters -Connection ID	The physical observer can decide to request the creation of an optical connection in order to accommodate a flow that has been classified at the sFlow collector.
-Physical observer Modification of an optical connection	-Optical provisioning manager URI PUT/restconf/config/optical-provisioning-	Status (Current status of the connection modifications)	The physical observer can request to modify an existing optical connection in order to better fit the requirements of the flow that employs

	manager:connection/connectionID		the optical connection identified by the Connection ID.
-Physical observer Monitoring of an established optical connection.	-Optical provisioning manager URI GET/restconf/config/optical-provisioning-manager:connection/connectionID	-Connection ID -Status -Connection type: -Recovery -QoS parameters -Time constraints -Monitoring info	The physical observer retrieves the details of an established optical connection with identifier Connection ID
-Physical observer Elimination of an optical connection	-Optical provisioning manager URI DELETE/restconf/config/optical-provisioning-manager:connection/connectionID	-Connection ID	Once an optical connection is not required anymore (due to having reached the end of its lifecycle or changes on the flow supported by the optical connection), the physical observer can request for tearing down the optical connection with the identifier Connection ID

3.2.3 Virtual Network Provisioning

In the vApp use case, it must be possible to request the provisioning of an overlay virtual network. Such task is mainly done through the overlay virtual network manager at the OVN controller. Table 7 summarizes the main interactions between the orchestrator and the SDN controller in regards of creation, update and destruction of virtual network instances.

Table 7 – Summary of interaction between the orchestrator and the SDN controller for vApp overlay virtual network provisioning.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**OVN controller**
-Physical observer List of virtual networks	- Overlay virtual networks manager URI GET/restconf/config/overlay-virtual-networks-manager/tenant_I	Tenant_ID networks: - administrative state - ID - Is shared	Return the list of networks belongs to tenant ID

	D	- Status - Subnets	
-Physical observer Create virtual network	- Overlay virtual networks manager URI POST/restconf/config/overlay-virtual-networks-manager	- administrative state - name - is shared - ID - Is external	Creates a virtual network.
-Physical observer Bulk create virtual networks	- Overlay virtual networks manager URI POST/restconf/config/overlay-virtual-networks-manager	- administrative state - name - is shared - ID - Is external	Creates multiple networks in a single request.
-Physical observer Show virtual network	- Overlay virtual networks manager URI GET/restconf/config/overlay-virtual-networks-manager/ID	Virtual network ID: - administrative state - ID - name - is shared - status - subnets - tenant_ID	Shows information for a specified network.
-Physical observer Update virtual network	- Overlay virtual networks manager URI PUT/restconf/config/overlay-virtual-networks-manager/ID	Virtual network ID: - administrative state - ID - name - is shared - status - subnets - tenant_ID	Updates a specified network.

-Physical observer Delete virtual network	- Overlay virtual networks manager URI DELETE/restconf /config/overlay-virtual-networks-manager/ID	Virtual network ID: - ID	Deletes a specified network and its associated resources.

3.3 DC O&M Use Case – Requirements and Mapping to SDN Controller

Section 4.3.4 of deliverable [D4.2] already described the infrastructure-driven interactions and requirements that the DC O&M use case imposes towards the SDN, its NBI and services, which constitute the relevant ones for this section. Thus, the main interactions and requirements towards the SDN controller are:

- Infrastructure controller (ODL controller)
- Overlay controller (OVN controller)
- Neutron level network abstractions
- Monitoring of network infrastructure and network services performance

The matching among the UC interactions and requirements with the options enabled by the COSIGN controller NBI is fundamental to ensure the proper behavior of the DC O&M use case operations. Thus, in this section we go deeper in the detail on how previous enounced requirements can be mapped to the interfaces and services facilitated from the control plane.

Two reference tenants have been considered for the discussion in terms of "*what type of information could each of them get while benefitting of the interaction with the SDN based control plane*". They are the DC administrator (acting as DC Cloud provider as well) and the DC service instance owner.

- The DC admin is the administrator of the physical DC which operates and manages the entire pool of DC resources (compute and network).
- The DC service owner could be the VDC service (VDC owner), the vApp service (vApp Owner) or any other owner of a service provided through the COSIGN orchestrator.

The next figure shows the steps involved in an operation that belongs to this use case: creation of an optical circuits to support a maintenance task such as VM migration or data transfer between two VMs. The figure depicts the high level workflow and architecture for the DC O&M use case. In general, the admin interacts with the OS platform to perform various operations: monitoring, configuration, etc. With respect to the network side, OS uses ODL to enforce the wanted configuration. As illustrated in the figure, the actual creation of the optical circuit is performed by ODL.

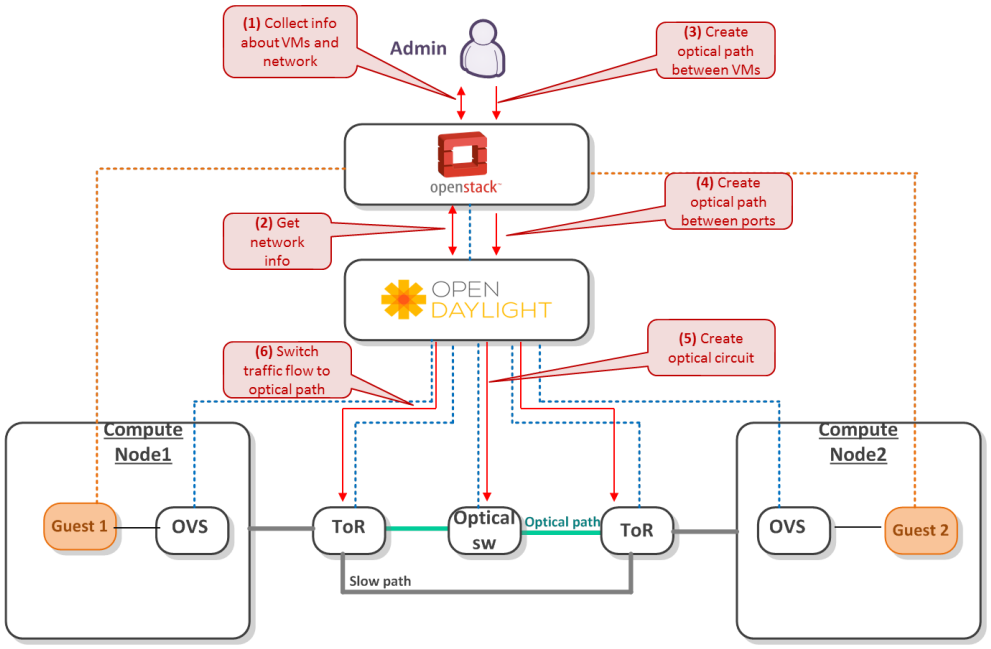

Figure 5 – Example of DC operation: Migrate traffic flow to optical circuit

With these, the following sub-sections detail for each of the identified requirements the actions that have to be triggered towards the SDN controller, the involved SDN controller modules/services, the information that can be retrieved/modified and how each of the previously defined entities can/needs to utilize the services. In this regard, the reported actions are focused on "*Read*" and "*Update*" operations aiming to focus the scope over already existing service instances.

3.3.1 Infrastructure Controller (ODL Controller)

This requirement directly involves the optical resources present in the DCN. Two main actions are expected regarding this requirement:

1. Access to the characteristics and monitoring information of the optical devices
2. Enforcement of specific configuration options (update) on the optical devices and set up connectivity services in the optical DCN.

The provisioning of optical connectivity service and its interface constitutes the entry point to address this requirement at the control layer. According to the SDN controller NBI specification [D3.2] the following aspects are available through this interface to either monitor or commit some changes over the underlying optical fabric. Table 8 summarizes the available information that the considered tenants may be able to monitor and the operational (update) options with which they are enabled.

Table 8 – Summary of interaction between the orchestrator and the SDN controller for DC O&M SDN infrastructure controller level.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**DC admin**	**Service instance owner**
- DC O&M component Monitoring of optical connections present in the DCN infrastructure	-Optical provisioning manager URI GET /restconf/config/optical-provisioning-	The description of the details for each connection established in the DCN. The format is the same used for the single connection. -Connection ID	The DC admin is able to monitor the details of any established optical	The tenant owning one or more optical connections within DCs is only able to get the information corresponding to the

	manager:connection/connectionID GET /restconf/config/optical-provisioning-manager:connection/connectionID	-Status -Connection type: •P2P •P2MP •Anycast -Source and destination endpoints and type -Recovery -QoS parameters: •Minimum reserved BW •Maximum BW •Maximum Delay •Maximum packet loss •Class of Service (CoS) -Time constraints -Monitoring info	connection.	"Connection ID" paths owned by him/her.
- DC O&M component Update and operation actions onto the existing optical connections in the DCN infrastructure	-Optical provisioning manager URI PUT /restconf/config/optical-provisioning-manager:connection/connectionID	Status (Current status of the connection modifications)	The DC admin is able to update and enforce changes on any existing optical network path.	Tenants owning an optical network path can only enforce restricted updates on the "connections ID" they own.

3.3.2 Overlay Controller (OVN Controller)

The DC O&M must have access to the virtual switches that implement the overlay virtual networks, so that it is possible to enable monitoring and control mechanisms on the overlays. The provisioning of overlay virtual networks interface is clearly the entrance towards the SDN controller to be able to monitor and control specific aspects of the overlay virtual networks. More specifically, the use case tenants will be able to monitor and operate the following aspects (see Table 9):

Table 9 – Summary of interaction between the orchestrator and the OVN controller for DC O&M overlay controller level.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**DC admin**	**Service instance owner**
- DC O&M component Monitoring of overlay virtual networks	-OVN controller URI List Networks Show Network	- Tenant ID - Network ID - Network name - Shared network - Network status - Associated subnets -TenantID	The DC admin has complete monitoring rights over each tenant overlay virtual networks.	The tenant identified by the "tenant ID" will be able to request info of his/her owned networks. It is not possible for this user to modify the "shared network" status.
- DC O&M component Update/operate an already existing overlay virtual network.	-OVN controller URI Update network	- Network ID - Network Name - Shared - Network status - Associated subnets - Tenant ID - Router external	The DC admin is able to update and enforce changes on any overlay virtual network.	Tenants owning a virtual overlay network instance can only enforce restricted updates on the networks they own. The "shared" option status cannot be changed by this type of tenant.

3.3.3 High Level Network Abstractions

The DC O&M use case also requires access to high level network abstractions either to monitor them or exert some type of update or modification. Such abstractions could be overlay virtual networks, optical slices and also information and statistics related to ports and flows. The first one has been already described in section 3.3.2. The options for the optical slices match the options enabled by homonymous SDN controller service and are detailed in Table 10.

Table 10 – Summary of interaction between the orchestrator and the SDN controller for DC O&M high level network abstractions.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**DC admin**	**Service instance owner**
- DC O&M component Monitoring of virtual optical slices (e.g. VDC instances)	-Virtual Infrastructure Manager URI GET/restconf/config/virtual-infrastructure-	The description of the details for each virtual optical slice. The format is the same used for the single connection. -Topology	The DC admin is able to monitor the details of any established virtual	The tenant owning one or more virtual optical slices is only able to get the information corresponding to the "Tenant

	manager:slice/<TenantID>/<SliceID>	-QoS: •Minimum BW •Maximum BW •Maximum Delay •Maximum packet loss •CoS - Monitoring info - Time	optical slices.	ID" for indicated "Slice ID". This option is not enabled to tenants requesting for optical connectivity paths.
- DC O&M component Update/operate an already existing virtual optical slice.	-Virtual Infrastructure Manager URI PUT/restconf/config/virtual-infrastructure-manager:slice/<TenantID>/<SliceID>	Status (Current status of the slice modifications)	The DC admin is able to update and enforce changes on any existing optical slice.	Tenants owning a virtual optical slice can only enforce restricted updates on the "SliceID" they own.

3.3.4 Network Infrastructure and Network Services Performance

The DC O&M use case is also supposed to gain monitoring information about the status of specific network nodes, ports and flows traversing the DCN. Such monitoring options can be satisfied by means of the so-called "DCN information service options" specified in [D3.2]. The specific monitoring options of interest have been identified inTable 11. Enabling these options is useful from the DC administrator point of view to perform management options of the DC resources according to the information that can be retrieved with such service.

Table 11 – Summary of interaction between the orchestrator and the SDN controller for DC O&M network infrastructure and network services performance.

Orchestrator required action/information and involved modules	**SDN service/request**	**Information**	**DC admin**	**Service instance owner**
- DC O&M component Monitoring of network nodes	URI GET(/restconf/config/opendaylight-inventory:nodes)	List of nodes -Node ID -Ports	The DC admin is able to retrieve all the nodes available in the network, each of them with the list of its ports.	NA
- DC O&M component Monitoring of node ports	URI POST(/restconf/operations/opendaylight-port-statistics:get-all-node-connectors-statistics)	List of ports. - Port ID -Port Type -Port Stats	The DC admin is able to retrieve statistics about all the ports in a node	NA

- DC O&M component Monitoring of flows	URI POST(/restconf/operations/opendaylight-flow-statistics:get-all-flows-statistics-from-all-flow-tables)	List of Flows -Match -List -Flow Stats	The DC admin is able to retrieve statistics about all the flows in a node	NA
- DC O&M component Monitoring of flow tables	URI POST(/restconf/operations/opendaylight-flow-table-statistics:get-flow-tables-statistics)	List of Tables -Table ID -Table Stats	The DC admin is able to retrieve statistics about all the tables in a node	NA
- DC O&M component Monitoring of queues	URI POST(/restconf/operations/opendaylight-queue-statistics:get-all-queues-statistics-from-given-port)	List of Queues -Queue ID -Queue Stats	The DC admin is able to retrieve statistics about all the queues in a port	NA

4 Orchestrator's Control Layer Clients – Full Functional Specification

4.1 Open Virtual Network Client

According to decisions made in WP3, the Virtual Controller component of the COSIGN SDN controller layer is implemented using the Open Virtual Network (OVN) platform [2], [3]. OVN is an SDN controller built natively to control Open Virtual Switch (OVS) instances to interconnect virtualized guests with (Geneve) overlay tunnels [6]. OVN natively complements the existing capabilities of OVS to add native support for virtual network abstractions, such as virtual L2 and L3 overlays and security groups. Services such as Dynamic Host Configuration Protocol (DHCP) are also included. OVN's design goals include: production-quality implementation that can operate at significant scale, Cloud Management System (CMS) integration exemplified by OS and easy inclusion of advanced network services.

To fulfil its role, COSIGN Orchestrator must communicate with the OVN network virtualization platform as a management client, taking place of a CMS component in the OVN deployment described in the following subsection.

4.1.1 Components of the Open Virtual Network Client

An OVN deployment consists of several components:

1. A Cloud Management System (CMS), which is OVN's ultimate client (via its users and administrators). OVN integration requires installing a CMS - specific plugin and related software. OVN initially targets OS as CMS.
2. An OVN Database physical or virtual node (or, eventually, cluster) installed in a central location.
3. One or more (usually many) hypervisors. Hypervisors must run Open vSwitch. Any hypervisor platform supported by Open vSwitch is acceptable.
4. Zero or more gateways. A gateway extends a tunnel-based logical network into a physical network by bi directionally forwarding packets between tunnels and a physical Ethernet port. This allows non-virtualized machines to participate in logical networks. A gateway may be a physical host, a virtual machine, or an ASIC-based hardware switch that supports the vtep schema.

Figure 6 shows how the major components of OVN and related software interact. Starting from the top, it can be identified:

1. The Cloud Management System, as defined above.
2. The OVN/CMS Plug-in is the component of the CMS that interfaces to OVN. In OS, this is a Neutron plug-in. The plugin's main purpose is to translate the CMS's notion of logical network configuration, stored in the CMS's configuration data base in a CMS-specific format, into an intermediate representation understood by OVN. This component is necessarily CMS-specific, so a new plug-in needs to be developed for each CMS that is integrated with OVN. All of the components in Figure 6 are CMS-independent.
3. The OVN Northbound Database receives the intermediate representation of logical network configuration passed down by the OVN/CMS Plugin. The database schema is meant to be "impedance matched" with the concepts used in a CMS, so that it directly supports notions of logical switches, routers, ACLs, and so on. The OVN Northbound Database has only two clients: the OVN/CMS Plugin above it and ovn-northd below it.
4. ovn-northd connects to the OVN Northbound Database above it and the OVN Southbound Database below it. It translates the logical network configuration in terms of conventional network concepts, taken from the OVN North-bound Database, into logical data-path flows in the OVN Southbound Database below it.

5. The OVN Southbound Database is the centre of the system. Its clients are ovn-northd above it and ovn-controller on every transport node below it. The OVN Southbound Database contains three kinds of data: *Physical Network (PN)* tables that specify how to reach hypervisor and other nodes, *Logical Network (LN)* tables that describe the logical network in terms of "logical data path flows," and *Binding tables* that link logical network components' locations to the physical network. The hypervisors populate the PN and Port_Binding tables, whereas ovn-northd populates the LN tables. OVN Southbound Database performance must scale with the number of transport nodes.

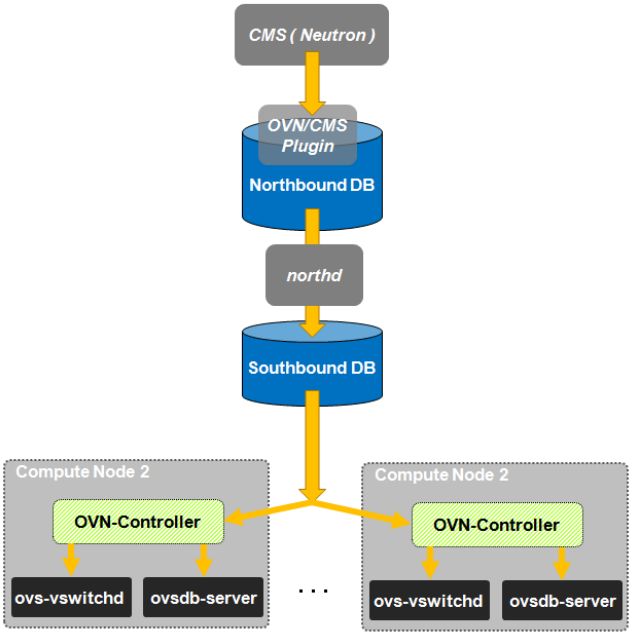

Figure 6 – OVN Architecture

The remaining components are replicated onto each hypervisor:

1. ovn-controller is the OVN's agent on each hypervisor and software gateway. Northbound, it connects to the OVN Southbound Database to learn about OVN configuration and status and to populate the PN table and the Chass is column in Binding table with the hypervisor's status. Southbound, it connects to ovs-vswitchd as an OpenFlow controller, for control over network traffic, and to the local ovsdb-server to allow it to monitor and control Open vSwitch configuration.
2. ovs-vswitchd and ovsdb-server are conventional components of Open vSwitch.

4.2 Open Daylight Client

OS must interact with ODL in order to orchestrate network resources. The interaction between OS and ODL is realized through REST APIs. This requires that ODL exposes the REST APIs and that OS hosts the entity which is able to consume the exposed APIs. The logical entity residing in OS which manages the communication with ODL is termed ***ODL client*** [4].

The OS software component that manages the network resources is Neutron; hence the communication with the ODL controller is managed by Neutron. Neutron has a modular architecture which enables easy interactions with various SDN controllers. Neutron's modularity is enabled through the Modular Layer 2 (ML2) plugin (Figure 7) [5].

The ML2 plugin contains several components, detailed in the next subsection, out of which the most important are the *mechanism drivers* (or simply *drivers*). A mechanism driver is the component that implements the specific interface towards an SDN controller. Each SDN controller that integrates with OS will have its own mechanism driver. Basically, the ML2 plugin together with the ODL mechanism driver (Figure 7) represent the *ODL client*.

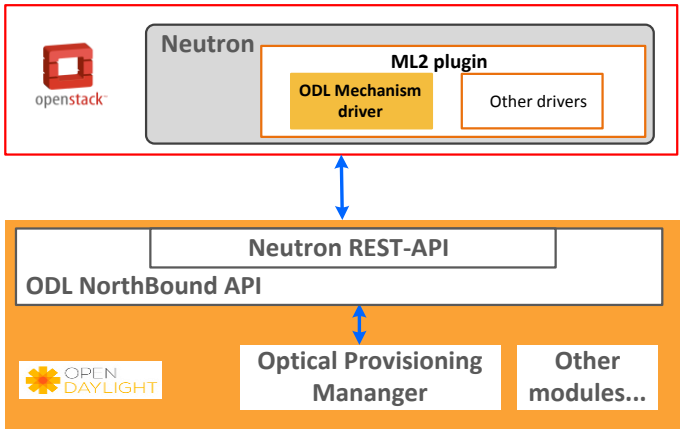

Figure 7 – Generic interaction between ODL client and ODL controller

Within COSIGN, the orchestrator coordinates also optical resources apart from the electronic network elements. This requires implementing extensions in OS in order to support the optical resources. In conclusion, the ODL driver, ML2 plugin, and other Neutron components must be extended with new functionality, related to the new types of resources proposed in COSIGN.

4.2.1 Components of the Open Daylight Client

Figure 8 shows a detailed view of the Neutron architecture with the ML2 plugin. The role of ML2 is to easily extend the support for various ways of implementing the network orchestration in OS, without code duplication as it was with previous standalone Neutron plugins. To do this, the ML2 plugin has a modular architecture with the following components:

1. Neutron server: contains the functionality for the Neutron service, where all the plugins fit in. It is not specific to the ML2 plugin, but other plugins can be used as well.
2. The core ML2 plugin: contains the commons functionality for a plugin, extracted to avoid code duplication.
3. Type manager: manages the selection of the type driver.
4. Type driver: implements a specific type of network virtualization. For example, when a new virtual network is created for a tenant, the type of network can be specified, leading to the selection of a specific type driver to implement the requested virtual network.
5. Mechanism manager: manages the selection of a specific mechanism driver.
6. Mechanism driver: is in charge of binding the request for services (e.g. creation of virtual network) to the specific network controller or data plane device that can fulfill the request.
7. API extensions: implement various extensions for the Neutron API.

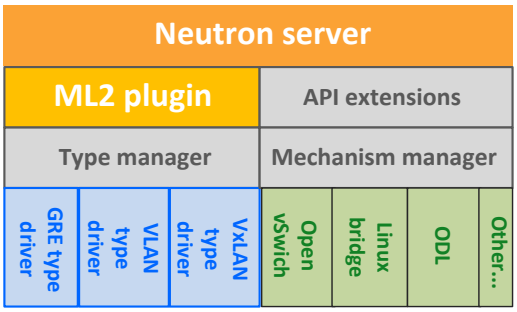

Figure 8 – Architecture of Neutron with ML2 plugin

4.2.2 COSIGN extensions in Open Daylight Client

The Neutron extensions in COSIGN are mainly new Neutron resources that are related to additional network entities and concepts defined at the interface between the SDN controller and the orchestrator. For example, for the VDC use case, the orchestrator needs to manage the additional resources defining

vLinks, *vTORs* or *vOSs* (virtual Optical Switches), i.e. the new elements which a customer of the VDC service can specify in the request of his/her virtual environment. These elements need to be modelled and handled in the different OS components (i.e. at the VDC portal, VDC algorithms, Heat and Neutron) up to the interface with the SDN controller, where they will be instantiated on suitable physical resources by the VTN application.

Traditional approaches are based on resource models which, at the orchestrator level, are completely agnostic of the underlying technologies and infrastructures. They define very abstract resources which are then translated on suitable physical constraints by algorithms running in the SDN controller. The current model of Neutron resources, as well as Neutron interface with ODL (and in particular the VTN application) is based on this assumption. However, this approach has the drawback to consider computing and network domains as separated entities which cannot be optimized as a whole. COSIGN, to overcome this limitation, places the VDC algorithms at the orchestrator level, so that they can take decisions about the mapping between physical and virtual resources for the entire mix of computing nodes and network circuits. The enforcement of these decisions requires the specification of this physical/virtual resource mapping at the north-bound APIs of the SDN controller. In other terms, the requests for *vLinks*, *vTORs* and *vOSs* from Neutron to ODL must include references to concrete physical resources and their configuration (technology dependent), as computed by the VDC algorithms. For example, a *vLink* will include the following parameters:

- Source and destination, defined as *Host Aggregates* for *VLinks* between VMs or Node-IDs and ports in the ODL topology for *VLinks* between *vTORs* or *vOSs*.
- Description of the classifier (e.g. source and destination IP addresses and ports), to identify the traffic to be carried on the *vLink*.
- (Optional) Description of the physical path which must be configured on the data plane to create the *vLink*. In general the path can be described as a list of hops, but each hop needs to be defined according to its specific technology (for example, in an DWDM node Node-ID and ingress-egress port are not enough, since the wavelength must be also selected – assuming the wavelength continuity requirement).

The implementation of new Neutron resources in the ML2 plugin and in its ODL driver requires modifications in the following elements of Figure 8:

- **ML2-Plugin:** definition of Create, Read, Update and Delete (CRUD) methods for the new resources. These methods are invoked by the Neutron core component and implement the management of the interaction with the Neutron DB and the ML2 plugin mechanism driver manager.
- **API extensions:** abstract methods for CRUD operations on the new resources.
- **Mechanism-Manager:** implementation of the methods defined in the API extensions for CRUD operations on the new resources. These methods invoke the proper calls exposed by the ODL driver.
- **ODL driver:** parsing, formatting and management of the REST messages which are exchanged with ODL for CRUD operations on the new resources.

5 Operational Flows for COSIGN Use Cases

Based on the requirements defined for each specific use case in Section 3, this section describes the necessary operational flows for the realization of such requirements and the related actions. Previous deliverable [D4.2] already presented the operational flows for the realization of each of the CRUD methods for all use case, depicting the interaction among the internal modules and interfaces of the COSIGN orchestrator layer. In this section, we will focus on the interaction between the orchestrator and the control plane layer, which is realized through the SDN controllers (ODL and OVN) and the related interfaces.

5.1 VDC Use Case

5.1.1 Resource Utilisation and Availability

The DCN orchestrator, that is, the module that executes the allocation algorithms, must be able to access information on the network devices and their capabilities, the topology of the network and the flows with which each device have been programmed. The REST interfaces for accessing this information are simple and so the workflows are made up of simple requests for information and replies of lists. From the DCN orchestrator, the requests are issued towards the ODL client in the Neutron OS service (via Heat). The ODL client is then the responsible to interact with the corresponding modules/services at the ODL SDN controller. Figure 9 and Figure 10 detail the interactions for reviving the topological information of the physical network and the statistics of the flows and ports at the network, respectively.

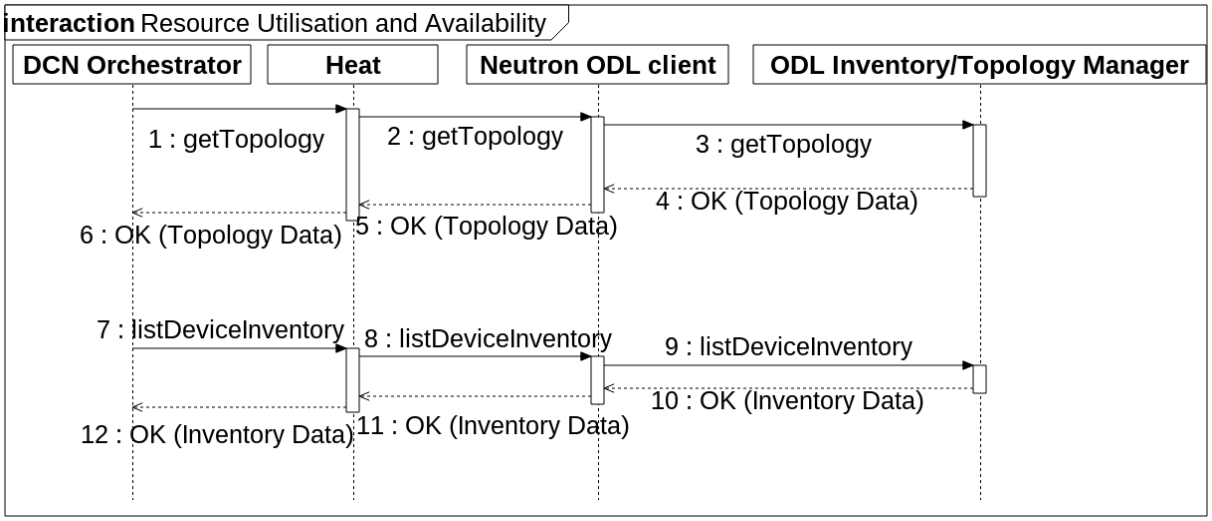

Figure 9 – Workflows for network topology information.

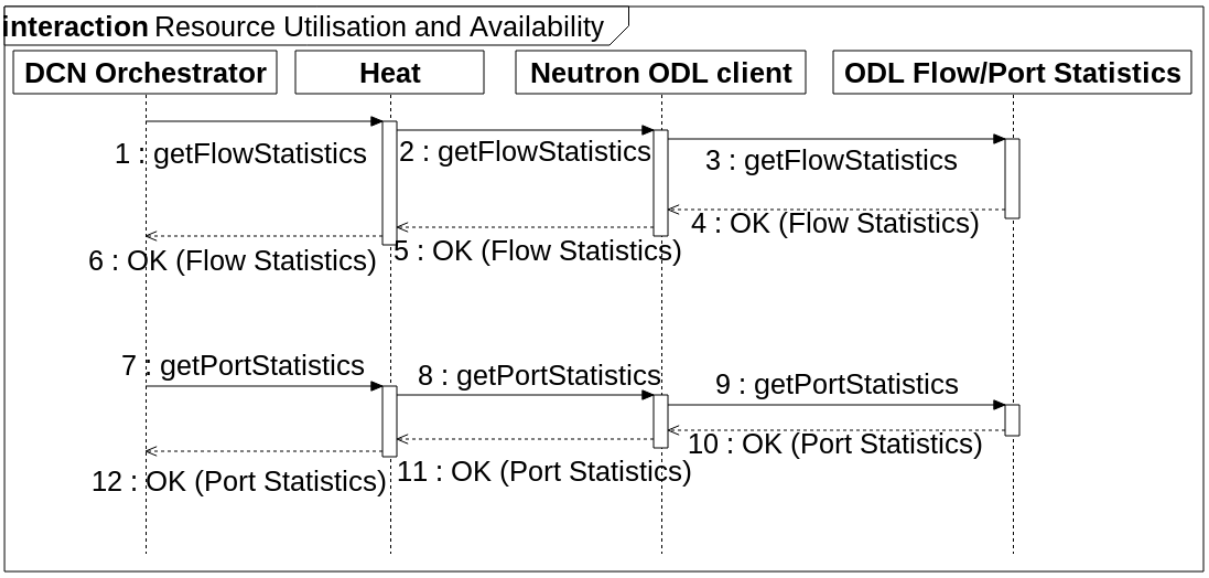

Figure 10 – Workflows for network utilization statistics.

5.1.2 Continuous Monitoring and Updates

Similar to the workflow for accessing information on the network's resources, runtime utilisation statistics and data can be accessed through simple requests to the relevant REST APIs. Such information is then utilized to trigger the update (if necessary) of an already deployed virtual slice, e.g. migrate the slice to other network resources. In such case, the VDC service requests for an update of the slice to the Virtual Infrastructure Manager at the ODL SDN controller via a POST request. At its turn, the SDN controller will trigger the modification of the configuration of the underlying optical resources and inform back the VDC service through the ODL client at Neutron with an acknowledgment (in case the update is successful) or an error if the update cannot be performed or the POST parameters were invalid.

Figure 11 depicts a schematic of the message exchange for the update interaction. Note that we do not depict the message exchange sequence for retrieving the runtime utilisation statistics since it has been already explained in the previous section.

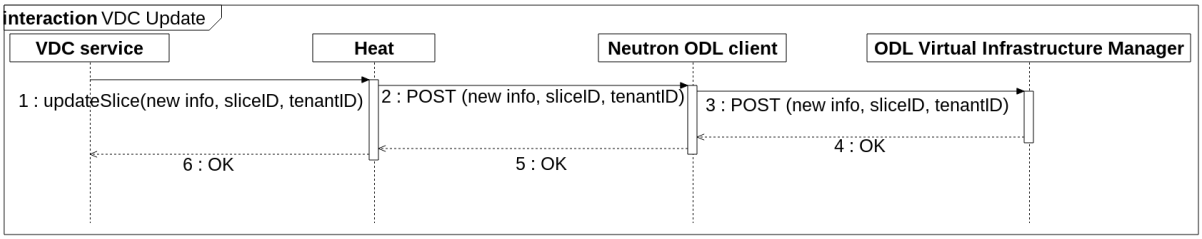

Figure 11 – Workflow for VDC slice update.

5.1.3 Explicit Resource Allocation and Slicing

When the VDC service needs to allocate resources for the creation of a new VDC instance, it simply uses a POST operation to pass requests to the Virtual Infrastructure Manager at the ODL SDN controller, stating the desired characteristics of the VDC instance, such as bandwidth per virtual link, topology, etc. VDC provisioning requests are fulfilled by the manager allocating physical devices to the slice and programming the flows in order to construct the virtual network of the VDC instance. In this regard, the manager informs back to the VDC service with the characteristics of the successfully deployed slice. If there are insufficient optical resources to fulfil the request, the VDC service will be informed directly by the orchestrator (i.e. the Heat module) as it has overall visibility of the underlying physical structure, so no further communication with the control plane is required.

For the case of eliminating an already deployed VDC instance, the VDC service will ask to the Virtual Infrastructure manager for the deletion of the slice passing both the slice and the tenant identifiers. Such action is triggered through a DELETE operation, which is consumed by the manager itself.

Figure 12 depicts the interactions for both creation and elimination of a virtual slice in fulfilment of a VDC request.

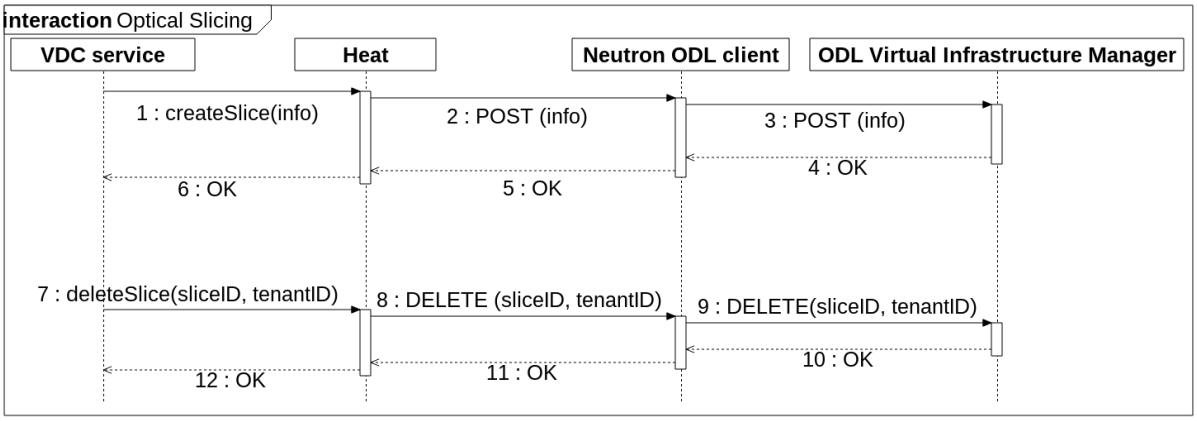

Figure 12 – Workflow for VDC slice creation and elimination.

5.1.4 VDC Network Isolation and QoS Guarantees

The VDC service must be able to configure arbitrary slices and flows while the VDC client must also have access to some programmable network features. However, the VDC service acts as an intermediary to ensure the validity of the client programming attempts. If the request is not valid e.g. an attempt to install a flow on an unallocated device, it can be rejected before being passed to the SDN controller. To configure the desired flows, the VDC service contacts the VTN Manager module at the ODL SDN controller. The VTN Manager is responsible to guarantee the proper flow isolation within a virtual slice, since isolation between optical slices (i.e. VDC instances) is guaranteed at the physical layer by reserving independent optical resources (e.g. wavelengths) for each one of them.

Figure 13 depicts an example of interaction for requesting and validating flows to be configured in an optical slice.

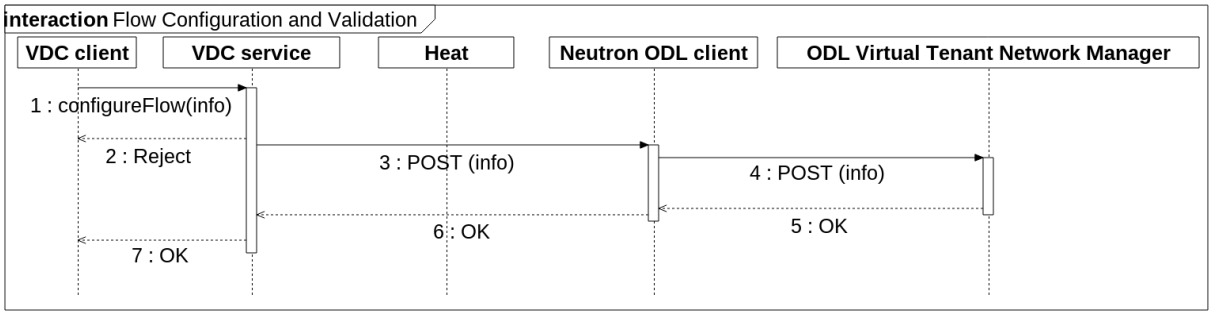

Figure 13 – Workflow for VDC flow configuration and validation

5.2 vApp Use Case

5.2.1 SDN Overlay Virtual Networks Controller Level

According to the flow classification that the physical observer performs, an overlay virtual network has to be established for allowing sending traffic over the virtual path. Additionally, consulting operations about the existing overlay virtual networks have to be allowed for the physical observer. The dynamic creation, consultation and elimination of overlay virtual networks to support the virtual applications and the related flows is essential for the correct operation of the vApp use case. All these operations (CRUD methods) are triggered from the dashboard vAPP service, which then contacts the OVN client at the Neutron module in OS. Then, the physical observer at the OVN contacts the ODL to

execute the desired actions. Such actions can be performed through simple REST APIs, which are present at the control plane layer. The usage of the REST APIs is described in [D3.2] on section 3.2.1. The following figures detail the interactions between the orchestrator and the control plane layer for the CRUD methods.

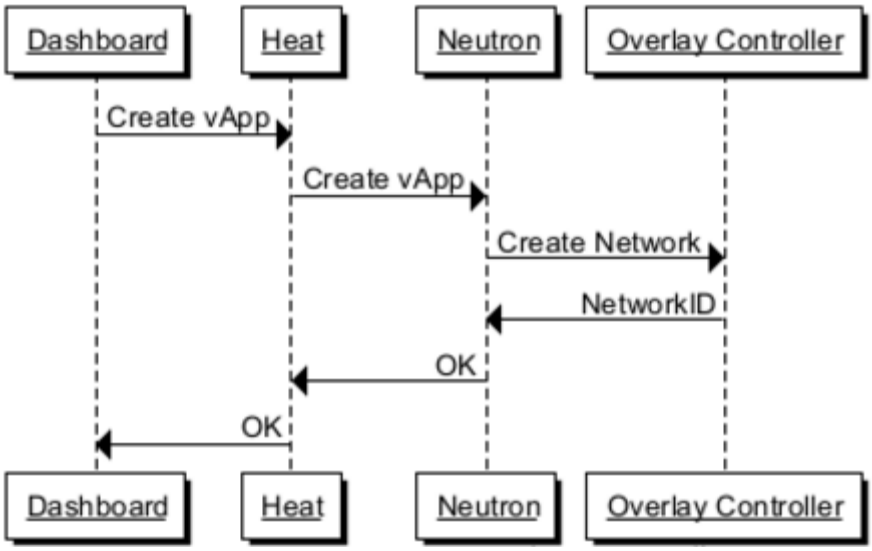

Figure 14 – Workflow for creating new vApp and new virtual network for it

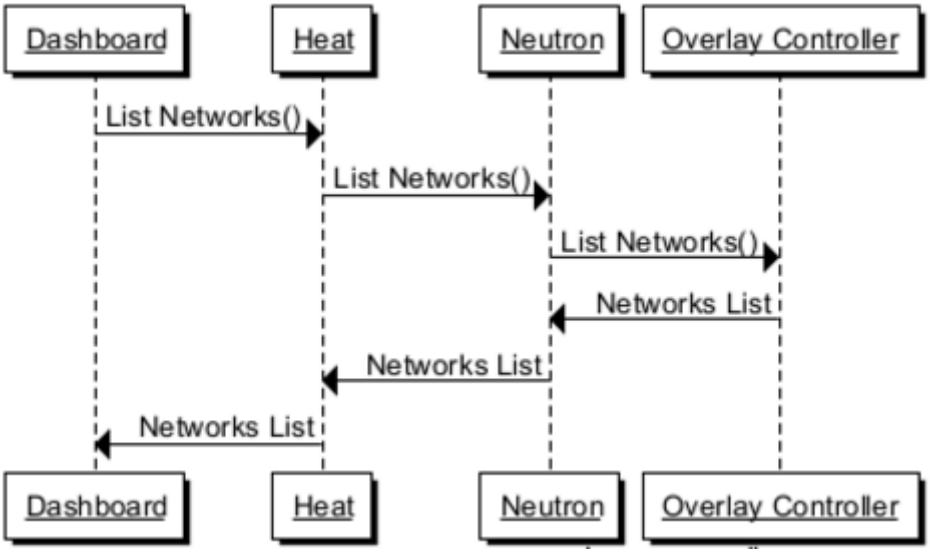

Figure 15 – Workflow for list all current virtual networks

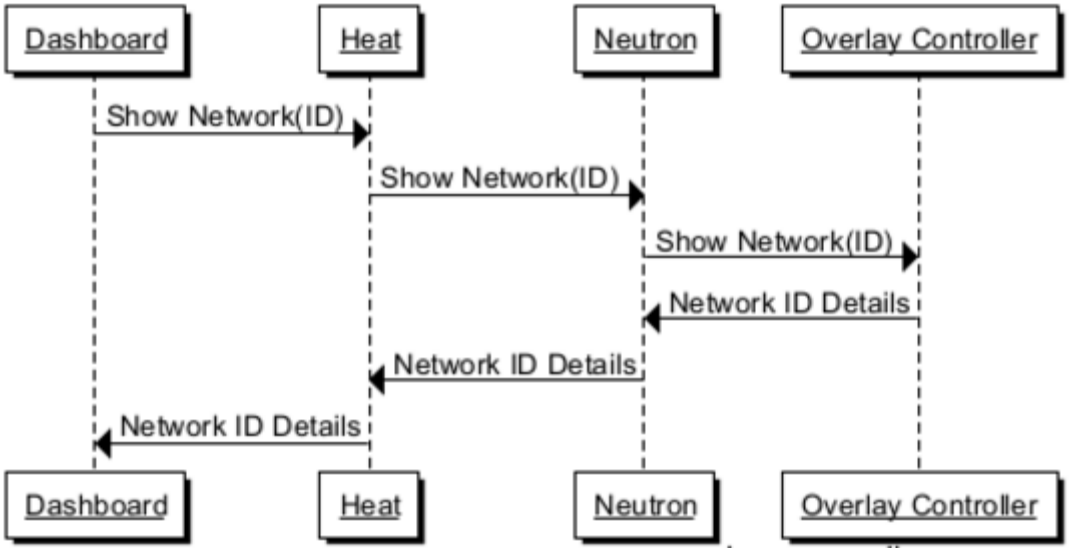

Figure 16 – Workflow for show details of specific virtual network

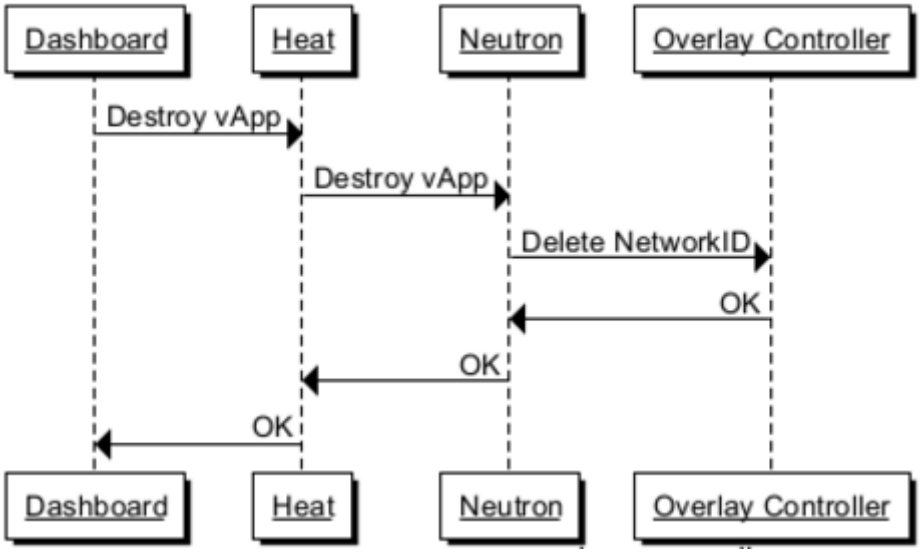

Figure 17 – Workflow for destroy vApp and its virtual network

5.2.2 SDN Physical Topology Controller Level

Information about the physical topology is fundamental for establishing the desired connections to support the flows of the virtual applications. In this regard, once a request for retrieving topological information has been issued (e.g. by the vApp dashboard service), the OVN client at Neutron will ask for the details of the physical topology at the ODL SDN controller. Such information is collected at the physical observer and sent back to the request issuer. Figure 18 shows the message sequence for this interaction and all the involved entities.

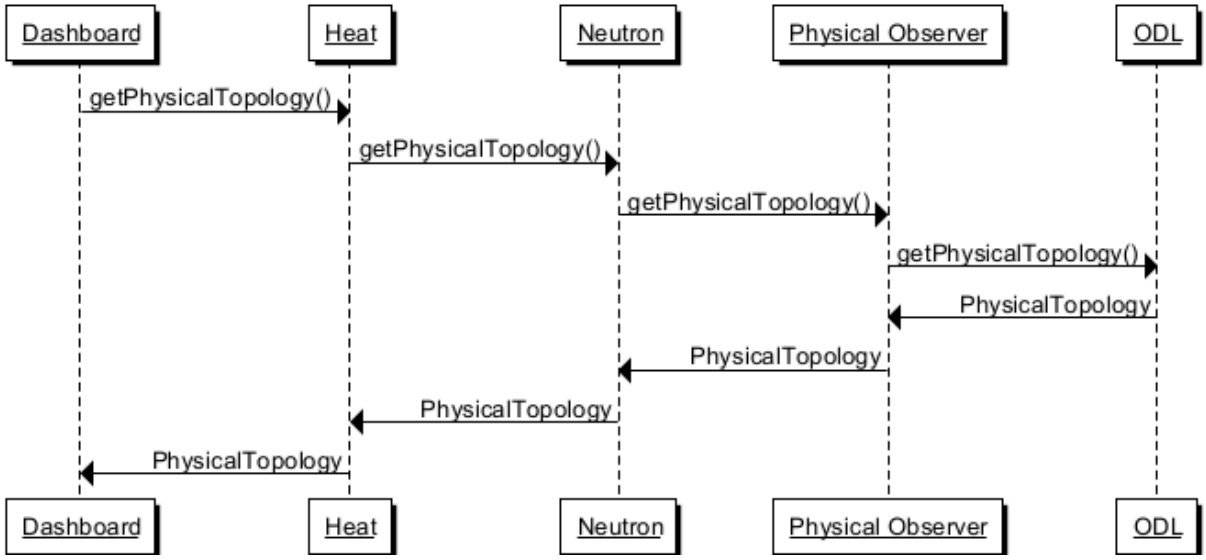

Figure 18 – Workflow for getting physical topology information

5.2.3 SDN Optical Connection Controller Level

In order to support the flows of a vApp, it may be necessary to establish an optical connection (e.g. an elephant flow). For this, the physical observer first discriminates the nature of the flow. Once the flow has been classified and the establishment of an optical connection is needed, the physical observer is contacted to configure the optical path. Once the optical connection has been established, the success of the operation is reported back. Besides the creation of new optical connections, monitoring and tearing down operations of already established optical connections is essential to ensure the good health of the applications running on top of the optical circuits as well as to ensure a dynamic provisioning of optical connectivity between arbitrary pairs of source/destination hosts at the DC infrastructure. Such actions can be performed employing the operations present at the REST API described in [D3.2] on section 3.2.3. The following figures depict the workflows for the main supported interactions, that is, establishment of new optical connections, monitoring and elimination of existing optical connections.

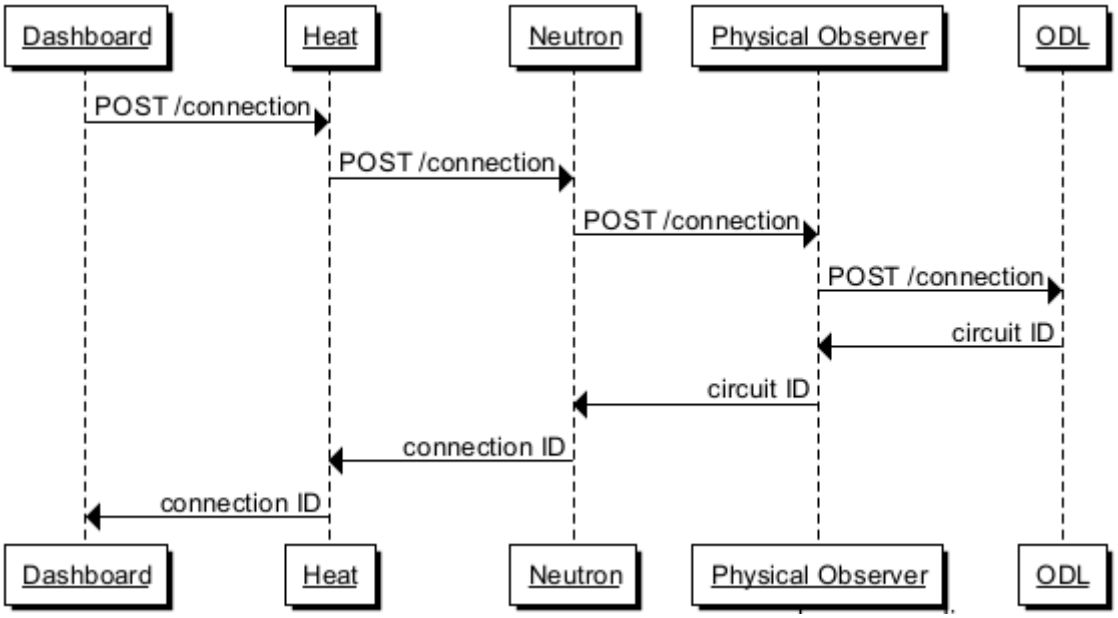

Figure 19 – Workflow for creating a new optical connection

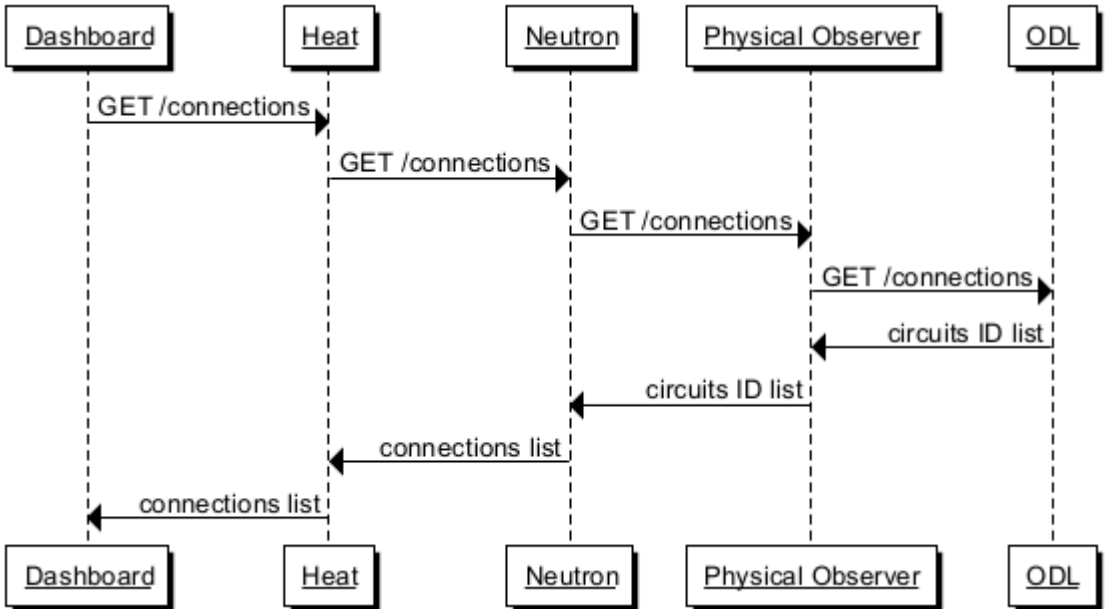

Figure 20 – Workflow for getting list of current optical connections

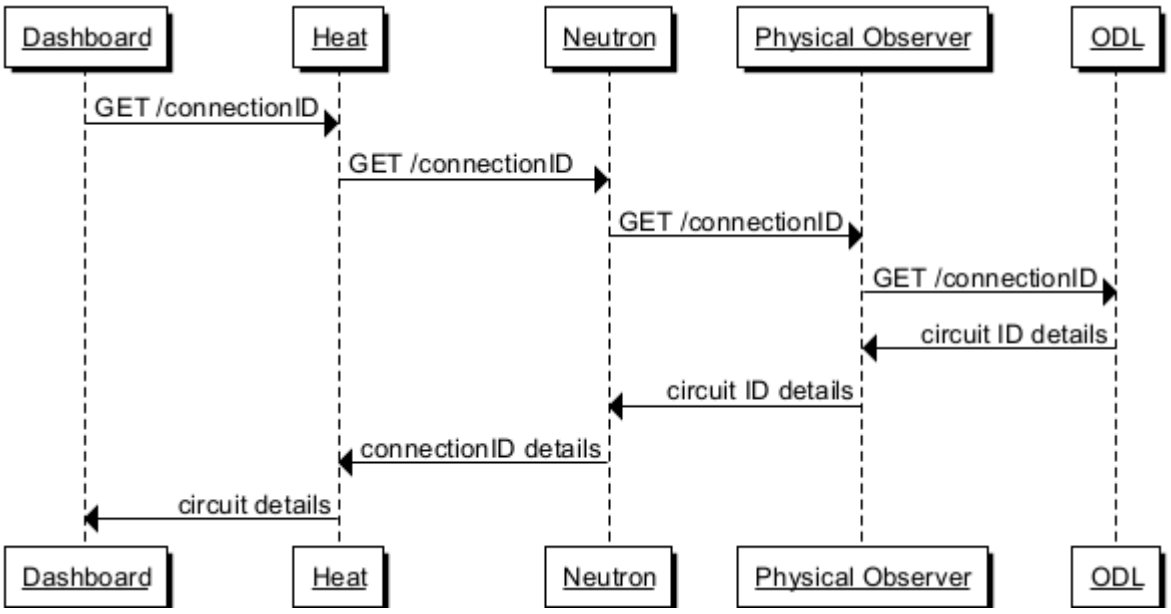

Figure 21 – Workflow for get details of specific optical connection

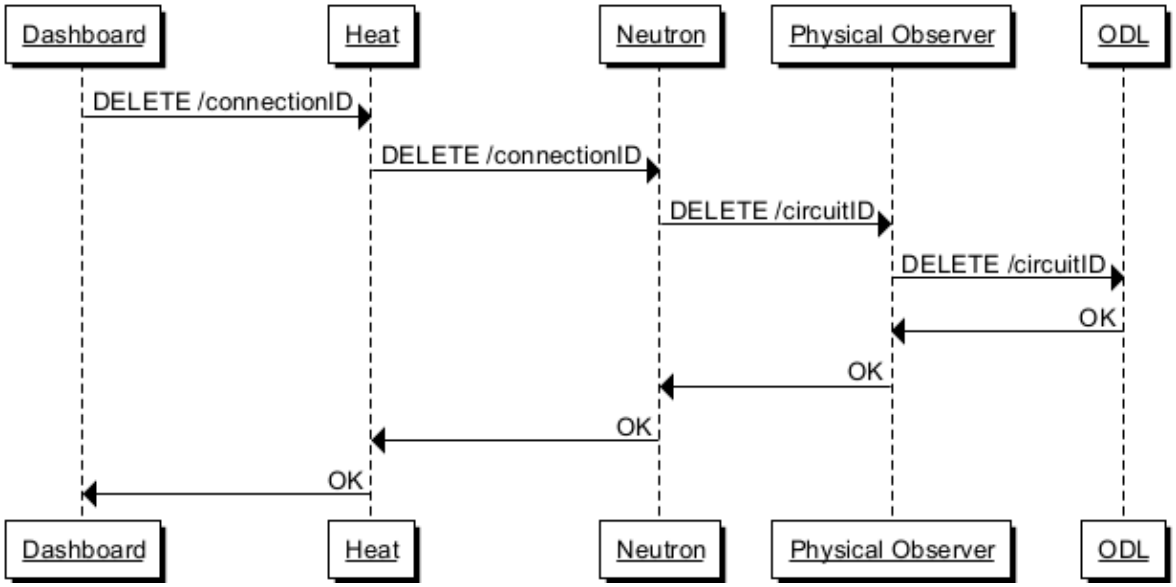

Figure 22 – Workflow for deleting optical connection

5.2.4 SDN Forwarding Rules Controller Level

To ensure proper utilization of the established connections and that flows are correctly routed over them, the vApp use case must be able to modify the related forwarding rules of an arbitrary circuit at the physical network. Such action is triggered by the OVN client at Neutron. The modification of the forwarding rules requires the circuit identifier for the circuit that is going to be modified and the new forwarding rules in the form of OF modification (i.e. flow matching). These options are then passed to the ODL SDN controller to trigger the modification of the forwarding rules. As the other interactions in the vApp use case, the physical observer sits on the middle of the message exchange, acting as an intermediator between Neutron and ODL. Figure 23 depicts the workflow for this operation.

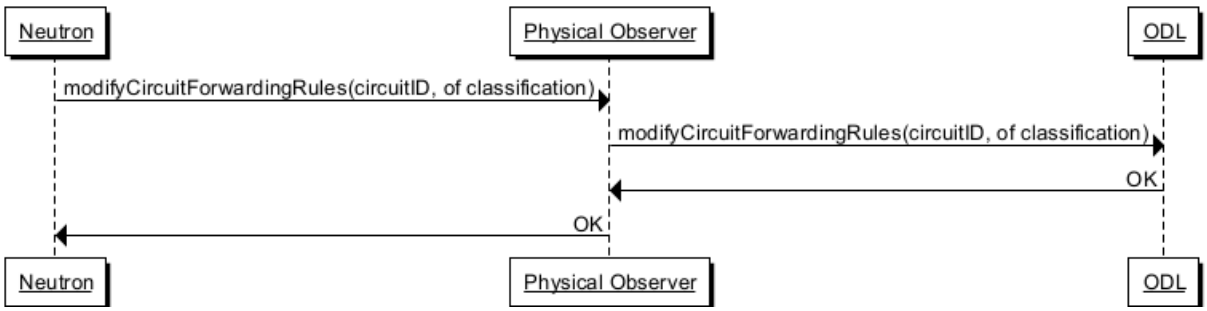

Figure 23 – Workflow for modify forwarding rules over connection ID

5.3 DC O&M Use Case

5.3.1 Infrastructure Controller (ODL Controller)

Optical connectivity at the controller will be mapped to the orchestrator using a new type of link, which comprises optical characteristics (*optical vLink*). Hence, a request for an optical connection arriving at the Heat component can be directly mapped to the controller since it just implies a delegation of the request to the Neutron component and further to ODL. Figure 24 shows the message sequence chart for this interaction.

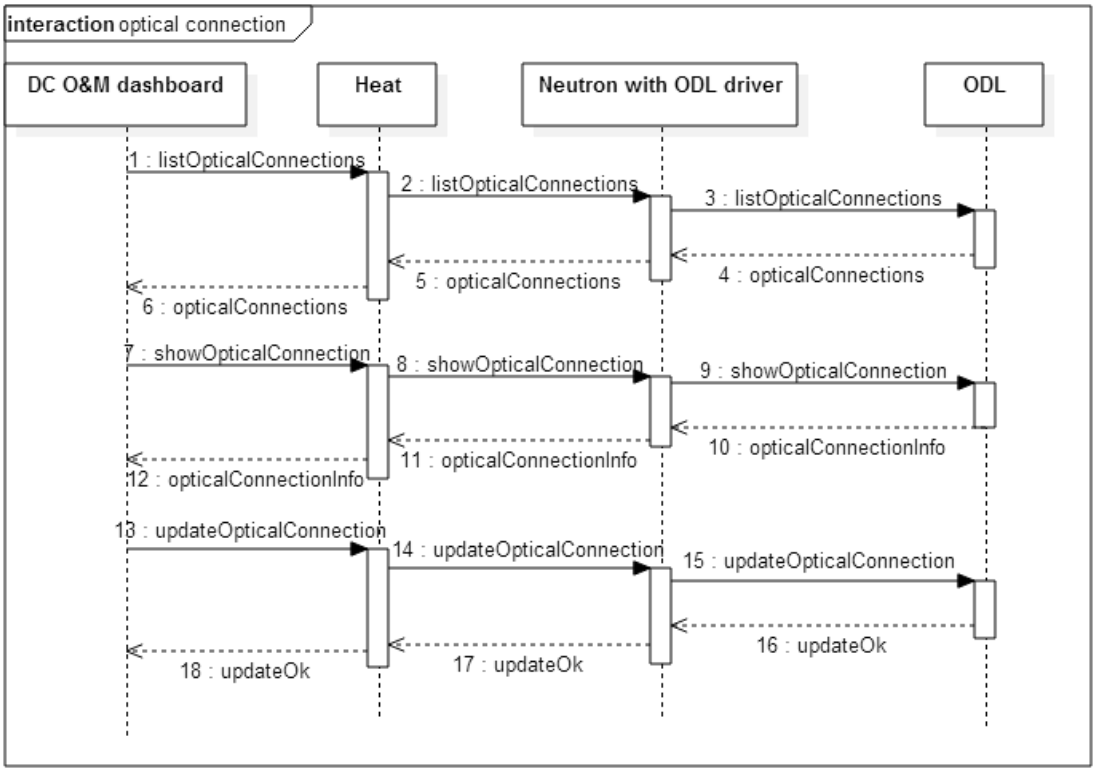

Figure 24 – Workflows for optical connectivity

5.3.2 Overlay Controller (OVN Controller)

The overlay virtual network functionality is provided by the OVN controller. Hence, the Neutron component contains an OVN driver through which the communication with the OVN controller is realized. OS already contains features related to overlay virtual networks but they are implemented through other mechanisms (e.g. OVS plugin, etc.) and not OVN. In COSIGN, these features are implemented using the OVN controller. Moreover, the overlay network will be supported by an optical DCN, which is controlled by the ODL controller. In conclusion, the mapping of requests from OS to OVN is straightforward and it just implies a delegation between orchestrator components (Figure 25). However, in order to create virtual networks, the OVN controller requests optical connectivity from ODL to support the connectivity in the overlay. The "create" operation is not depicted here since it is directly related to the provisioning of the service and less related to management operations.

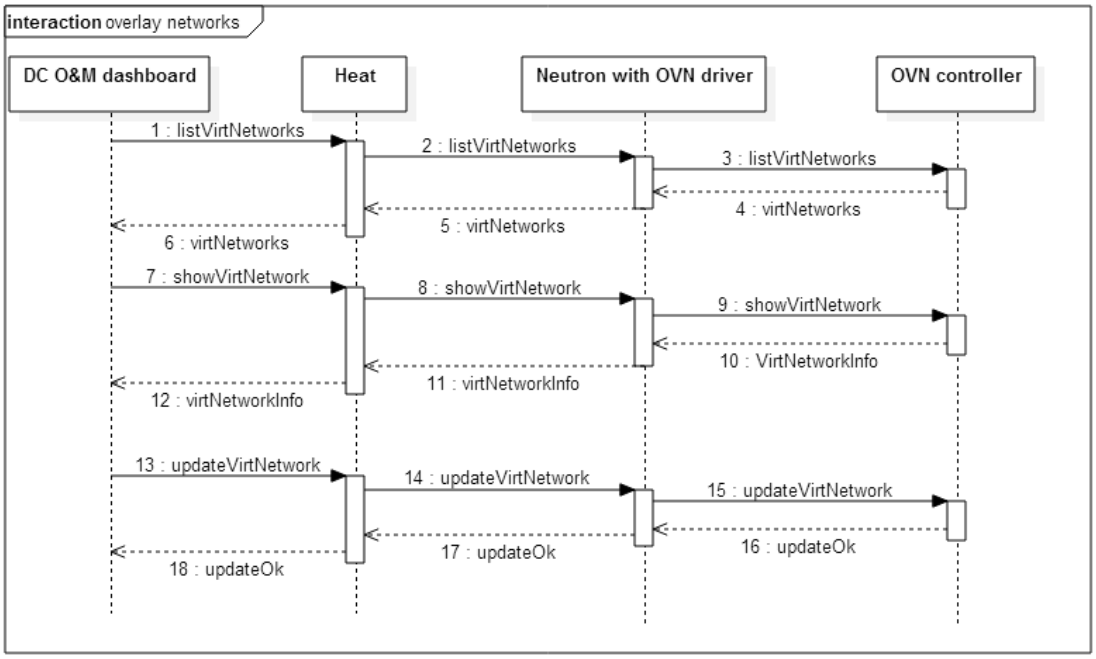

Figure 25 – Workflows for overlay networks

5.3.3 High level network abstractions

The usual management operations for optical slices are related to showing resources information and updates on the resources. The message sequence chart for these operations is depicted in Figure 26. OS does not contain concepts related to optical slices so the DC Admin interacts with the dashboard which further directly interacts with the ODL controller to fulfil the requests. The Virtual Infrastructure Manager component in ODL (section 3.2.2 in [D3.2]) is in charge of providing the functionality for this service. In conclusion, the data models available at the ODL controller will be directly used by the dashboard, and enabled by direct interaction with the ODL REST API.

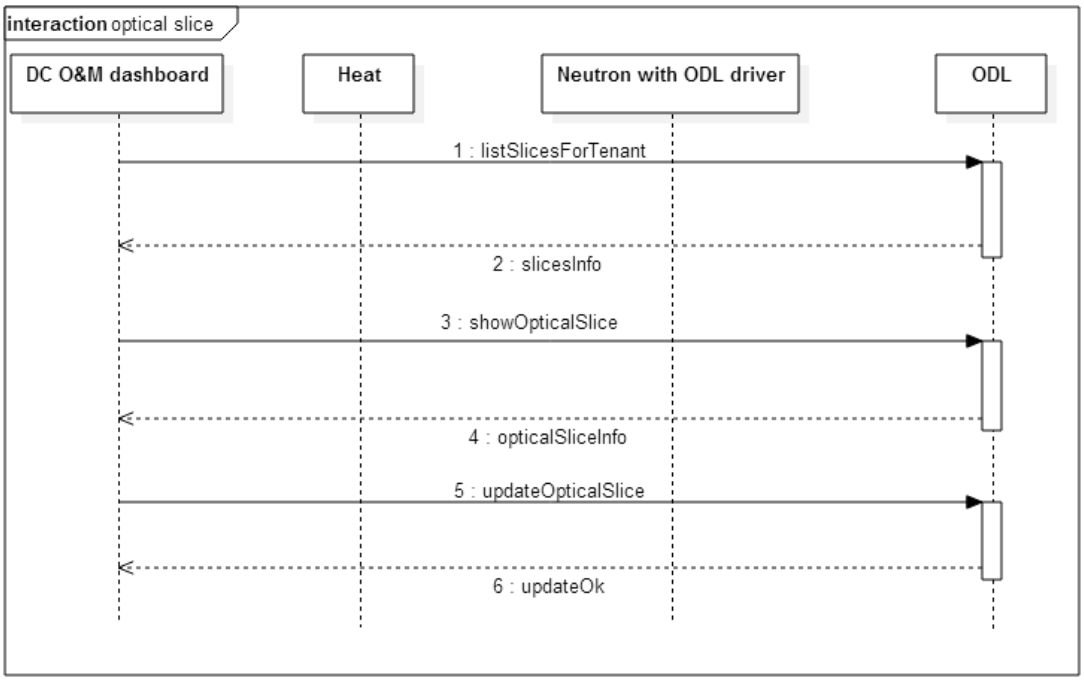

Figure 26 – Workflows for optical slice

5.3.4 Network infrastructure and network service performance

For the operations related to inventory and monitoring of the network infrastructure and services, the orchestrator can process part of them by using the functionality implemented in the OS components. As an example it can be listing the network nodes, listing of ports in a specific node, or showing port statistics (Figure 27). However, if the OS components do not contain the needed extensions then the requests are delegated from the dashboard to the ODL controller. This avoids unnecessarily extending OS components with resources types that are not of use for the orchestrator (e.g. flows and queues).

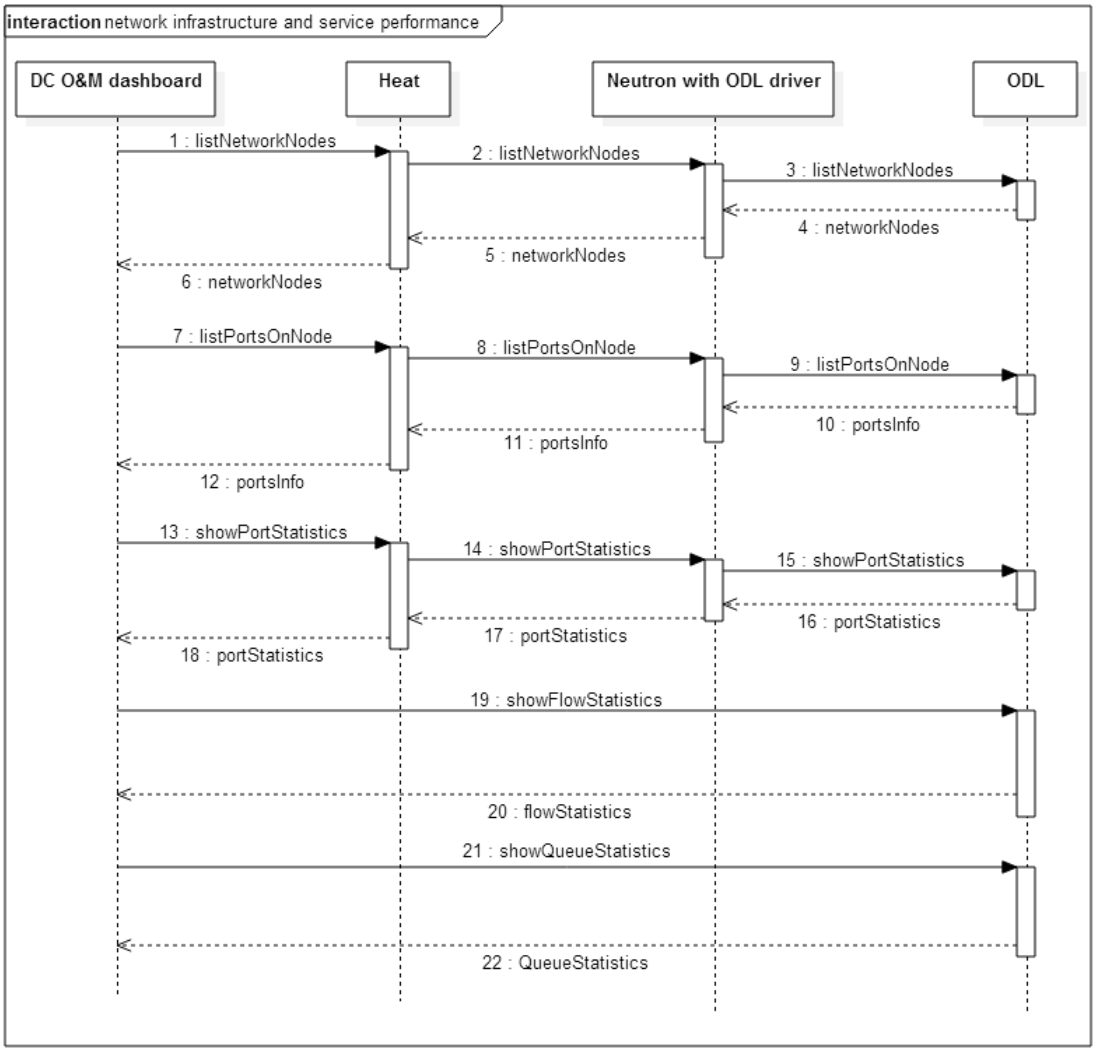

Figure 27 – Workflows for network infrastructure and service performance

6 Conclusions

This deliverable follows up on the previous one of this Work Package [D4.2] by providing a more complete specification of the interaction between the COSIGN Orchestrator and the control plane.

To this end, the specific requirements in terms of actions to be requested towards the control plane are analysed for each one of the COSIGN use cases, namely, Virtual Data Centre (VDC), Virtualized Cloud Application (vApp) and DC Operations and Management (DC O&M). Here, we mapped the specific requirements and actions to the operations and services that are available at the Northbound interface of the SDN controller, identifying how the high level operations at the orchestrator layer can be translated to concrete operations at the control layer.

Additionally, in this deliverable, the necessary control clients that sit within the OpenStack Neutron module at the orchestrator layer to handle the interaction between orchestrator and control layer are introduced and fully specified. The modular structure of the clients, namely, the Open Virtual Network client for the control and configuration of overlay virtual networks, and the OpenDayLight client, for the control and configuration of the physical infrastructure network, is presented. The particular extensions to support the COSIGN architecture are discussed as well.

The WP4 team will continue to interact with the WP3 team to fully flesh out the specified interaction as working code towards the integration of the orchestrator in WP5 for the purpose of demonstration in the concrete scenarios that the WP5 team is considering for the demonstration of the COSIGN software and data plane architecture.

REFERENCES

[1] "Software-Defined Networking (SDN): The New Norm for Networks". Open Networking Foundation.
[2] "OVN: Open Virtual Network for Open vSwitch", http://openvswitch.org/support/slides/OVN-Vancouver.pdf
[3] Open Virtual Network architecture, http://benpfaff.org/~blp/dist-docs/ovn-architecture.7.html
[4] OpenDaylight ML2 driver, https://wiki.opendaylight.org/view/OVSDB:Developer_Guide
[5] OpenStack Neutron ML2 plugin, https://wiki.openstack.org/wiki/Neutron/ML2
[6] https://tools.ietf.org/html/draft-gross-geneve-00