SDN Control of Disaggregated Optical Networks with OpenConfig and OpenROADM

Ramon Casellas[1] [0000-0003-0391-9728], Ricard Vilalta[1] [0000-0003-0391-9728], Ricardo Martínez[1] [0000-0003-3097-5485] and Raúl Muñoz[1] [0000-0003-4651-4499]

[1] CTTC/CERCA, Av. Carl Friedrich Gauss, 7, 08860 Castelldefels, Spain
ramon.casellas@cttc.es

Abstract. Most deployed optical transport networks are proprietary, behaving as a closed, highly coupled, single-vendor managed domain. Although their control planes and management systems may export high-level and open northbound interfaces (NBI), the internal details and interfaces are not disclosed to the network operator. However, driven by the requirements of telecommunication and datacenter operators and the need to keep costs down while supporting sustained traffic increase, a trend known as disaggregation has steadily emerged during the past years. It involves composing and assembling open and available components, devices and sub-systems into optical infrastructures and networks, combining "best-in-class" devices, tailored to the specific needs of the aforementioned operators. It has been motivated by factors such as an increase in hardware commoditization, a perceived different rate of innovation of the different components, a promised acceleration in service deployment, or the consequent reduction in operational and capacity expenses. In practice, disaggregation brings multiple challenges, depending on the level that applies (e.g., partial or total, down to each of the optical components) and is taking place in stages. It is commonly accepted that disaggregation implies a trade-off between: i) the opportunities due to the new degree of flexibility provided by component migration and upgrades without vendor lock-in and ii) the potential decrease in performance compared to fully integrated systems and the underlying complexity – including interoperability -, critical in full disaggregation scenarios. From the point of view of control and management, disaggregation heavily relies on the adoption of open interfaces exporting hardware programmability. Disaggregated optical networks are an important use case for the adoption or a unified, model-driven development. In this paper, tutorial in nature, we will introduce the main concepts behind Software Defined Networking (SDN) for disaggregated optical networks, presenting reference architectures and industry common practices related to the adoption of a unified, model driven approach. The second part will cover an overview of selected deployment models (e.g., addressing transceiver and OLS disaggregation) as well as the OpenConfig and OpenROADM optical device models and Transport API (TAPI) interfaces, which constitute the main elements of the implemented SDN control plane. Such control plane targets mainly the metro segment, as defined within the EC Metro-Haul and ONF ODTN projects.

Keywords: SDN, Disaggregated Optical Networks NETCONF/YANG, RESTConf, OpenROADM, OpenConfig.

1 Introduction

A main requirement for the operation of optical transport networks is the automation of the service provisioning process, minimizing manual intervention and across the whole network. This involves setting up connections and configuring the forwarding and switching behavior of intermediate nodes, with increasing traffic dynamicity requiring frequent and complex re-arrangement in an environment of increasing complexity. This complexity comes from the underlying technology – including not only the inherent complexity of the optical technology but also the increasing programmability of the DWDM systems and devices -- and the need to provision such services with multiple technological layers and in networks spanning multiple segments and across administrative domains [1].

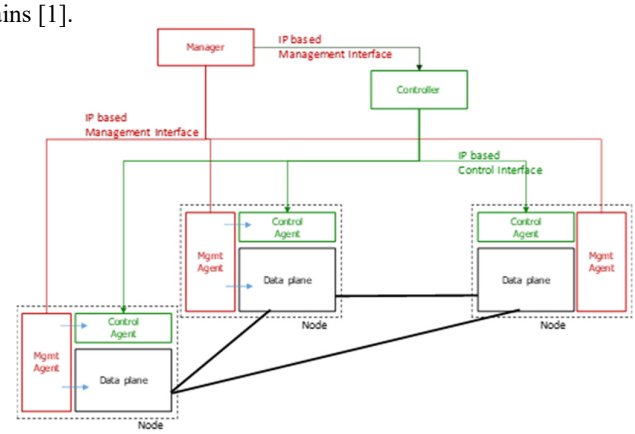

Fig. 1. Logical Functional view of a Centralized control plane.

A "Control plane" is a system and a set of functions especially dedicated to the provisioning of connectivity services, with higher abstractions, refined functional architectures, better addressing the (evolving) requirements, specially designed for and focusing on the configuration and specifics of the forwarding and switching operations, with standard interfaces operating across domains ensuring vendor inter-operability. A control plane that adopts a centralized deployment model (as is the case when following SDN principles) relies on a logical centralized controller, enabling an application layer and separating the control plane from the data plane, avoiding the complexity of distributed control planes and exploiting the programmability of network devices for greater flexibility and control (see Figure 1).

Considering the southbound interface (SBI), and in the scope of disaggregated optical networks, it would be desirable to have a common and unified protocol and underlying switch model, similar to the role of OpenFlow [2] in packet switched networks. In practice, this is not straightforward. On the one hand, OpenFlow is a low level, byte-oriented protocol, complex to extend and, in the optical domain, there seems to be little support, despite having published normative documents [3]. On the other hand, there is a need to interact with multiple elements using multiple (legacy) protocols, there is a

significant heterogeneity in terms of optical devices capabilities, and developing a common interface protocol would likely require agreed-upon hardware models and extensions. In general, a SBI protocol should be extensible, remaining future-proof and enabling generic configuration for existing and new devices. This is accomplished, in part, by decoupling the protocol to transport information and messages between entities from the way information is structured. The community seems to favor approaches based on a unified data modeling language and a standard transport protocol.

2 Model Driven Development

In simple terms, model driven development can be defined as an approach to design SDN controlled networks based on the systematic use of (ideally open and standard) data models. Such models should cover most aspects of network operation and control (e.g., service models, control plane constructs as topologies, connections and devices, e.g., "everything-as-model"…), and be specified using open and standard data modelling languages. In this way, such models can be automatically validated and used – including automated code and stub generation – and application logic can be developed around them. In general, for a given e.g. device, its Information Model macroscopically describes the device capabilities, in terms of operations and configurable parameters, using high level abstractions without specific details on aspects such as a particular syntax or encoding and its Data Model determines the structure, syntax and semantics of the data that is externally visible.

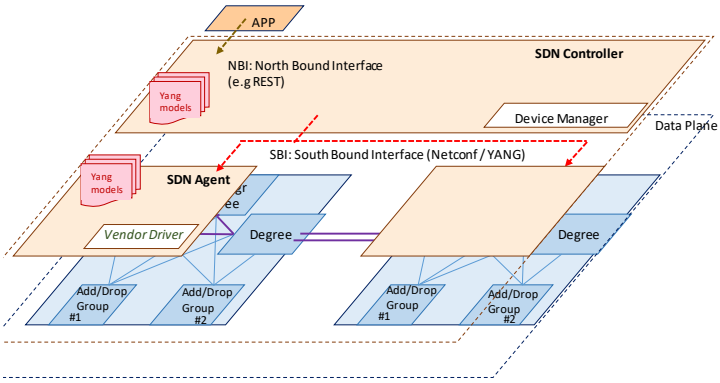

Fig. 2. Logical Functional view of a Centralized control plane.

To adopt a model driven development for disaggregated optical networks (see Figure 2), there are a set of basic requirements. First, a unified information and data modelling language, to describe a device capabilities, attributes, operations to be performed on a device or system and notifications; a common language with associated tools enabling complex models with complex semantics, flexible, supporting extensions and augmentation, a set of "best-practice" and guidelines for model authors. Next, an architecture for remote configuration and control, adopting a client / server paradigm, supporting

multiple clients, access lists, transactional semantics and rollback. Then, an associated transport protocol, which provides primitives to view and manipulate the data, providing a suitable encoding as defined by the data-model, which needs to be flexible, efficient, secure. Finally, interoperability is easier given standard, agreed upon models for devices. This area is a huge activity one, where it is hard to reach consensus (given the controversial aspects) and with Standards Defining Organizations (SDOs) competing and overlapping. Next, we elaborate more on the different requirements and choices.

2.1 The Yang modeling language

A Yang [4] module defines a data model; it includes a header, imports, and include statements, type definitions, configurations and operational data declarations as well as actions (RPC) and notifications. The language is expressive enough to structure data into data trees within the so-called datastores, by means of encapsulation of containers and lists. It is possible to define constrained data types (e.g. following a given textual pattern), to condition the presence of specific data to the support of optional features and, finally, to allow the refinement of models by extending and constraining existing models (by inheritance/augmentation), resulting in a hierarchy.

2.2 The NETCONF Architecture and Protocol

The NETCONF [5] protocol offers primitives to view and manipulate data, providing a suitable encoding as defined by the data-model. Data is arranged into one or multiple configuration datastores (set of configuration information that is required to get a device from its initial default state into a desired operational state). The protocol thus enables remote access to a device, and provides the set of rules by which multiple clients may access and modify a datastore within a NETCONF server (e.g., device). The protocol is, in simple terms, based on the exchange of XML-encoded RPC messages over a secure (commonly Secure Shell, SSH) connection. Out of the different messages and operations, the <get-config> allows retrieving part or the configuration data, the <edit-config> allows changing (creating, deleting, merging or replacing) that data (including adding elements to a container, or adding elements to a list, as allowed by the model) and, <get> allows to retrieve device state information and operational data.

It is worth mentioning that NETCONF typically requires an “XML over SSH” protocol stack, and a lightweight protocol also exists, more adopted to API consumers. A common web/HTTP approach to design services is based on the Representational State Transfer (REST) paradigm, an architectural style that defines a set of constraints to be used for creating Web services. RESTful Web services allow their clients to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. RESTCONF [6] [7][8] relies on REST and provides a HTTP-based API to access the hierarchical data within the running datastore. RESTCONF thus maps NETCONF operations to HTTP operations (such as POST, PUT, and DELETE) used to create and replace resources in the web, and supports two main encodings: XML and JSON.

2.3 Common Network and Device Models

From the point of view of control and management, disaggregated optical networks are an important use case for the use of open interfaces exporting programmability. However, optical networks are particularly challenging to model due to the lack of agreed-upon hardware models, and this is critical for the development of an interoperable ecosystem around disaggregated hardware. Regarding interoperability across multiple vendors, which is a key requirement in a disaggregated network, having common and agreed-upon device and network models simplifies development and integration. There are several SDOs working on this aspect. In this work, we focus on two main groups, which are suitable for the goal of demonstrating the ideas and concepts. OpenROADM MSA [7][8] defines vendor-neutral specifications for ROADMs as well as transponders and pluggable optics. Specifications consist of both interoperability and data models. OpenConfig [9] is a collaborative effort by network operators, has published a set of models providing a configuration and state model for terminal optical devices within a DWDM system, including both client- and line-side parameters.

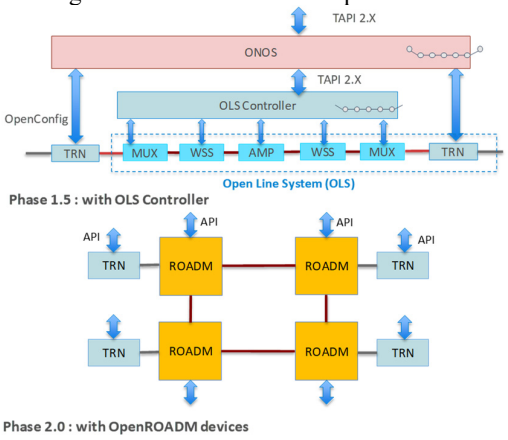

Fig. 3. ODTN phases for the two partial disaggregation models (src: ONF ODTN project)

3 Targeted Use Cases and Disaggregation Models

An in-depth discussion of the different options regarding disaggregation is provided in [10] including partial and full models. Here, we cover mainly: i) partial disaggregation with transceiver and Open Line System (OLS) and ii) partial transceiver and network element disaggregation (see Figure 3), as the main use cases in the scope of the Metro-Haul [11] and the ODTN projects [12]. The Metro-Haul project goal is to design cost-effective, energy-efficient, agile and programmable metro networks, scalable for 5G access and future requirements, including the design of all-optical metro nodes (including full compute and storage capabilities) that interface with both 5G access and multi-Tbit/s elastic core networks. The ODTN (Open and Disaggregated Transport Networks) is an ONF project that aims to rally service providers, hardware vendors and

system integrators, to build a reference implementation using open source software, open and common data models. Both projects deal with disaggregated DWDM systems, including but not limited to transponders and Open Line Systems, amplifiers, multiplexers, all-optical switches and ROADMs, covering the extension of an open source network operating system (ONOS SDN Controller) for control.

4 Selected Data Models and protocols

4.1 Transport API (T-API)

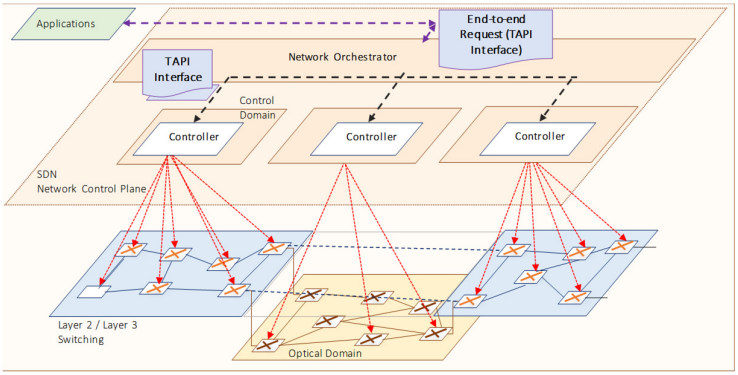

Fig. 4. TAPI SDN controller NBI for Network Orchestration

As stated earlier, SDN Controllers offer proprietary interfaces to applications (or Network Orchestrators), an approach commonly referred to as "vendor domains or islands". This heterogeneity, due to having different controllers interfaces in a multi-domain context, forces the use of "plugins" and it is difficult and expensive to extend (with the so-called umbrella management systems, used by the operators to deploy services spanning multiple domains). As a driving motivation and clear problem statement, there is a need for a standard interface, with common models, to act as a controller NBI (see Figure 4). The Transport API (T-API) [13] published by the ONF meets the main requirements to be a protocol and interface used between an orchestrator and multiple domain controllers. In our case, it constitutes the Optical controller NBI (or the OLS). Of particular interest is the TAPI 2.1 release, given the support for the photonic media and its use to request (network) media channels in either disaggregated model. The main concepts within TAPI are detailed next. A TAPI based interface offers multiple services; in the scope of this paper, we consider the topology and connectivity.

1) Common Context. The TAPI context is the shared information between a TAPI client (user) and the TAPI server (SDN controller). The model defines a TAPI domain as being able to provide services between Service Interface Points (or SIPs) mainly characterized by their universally unique identifiers (UUIDs). A basic operation for a client is to "retrieve" the context in order to obtain the list of SIPs, so connectivity services are requested between two (or more) exported SIPs. **2) Topology context**

and models. If a given TAPI server supports topology model, it augments the TAPI shared context with a (list of) topology(ies). Each topology is composed of a list of nodes, which, in turn, have Node Edge Points (NEPs). Links connect two NEPs. The model is flexible enough to support recursive topologies and different levels of abstraction. The level of detail exported is configurable by policy. A client is thus able to obtain an (abstracted) view of the topology and map TAPI SIPs to external NEPs. **3) Connectivity context and models.** Finally, the third model augments the shared context in order to support Connectivity Services. The instantiation of a connectivity service relies on the instantiation of several connections (e.g. one end-to-end and internal at each TAPI node). For this, Connection End Points (CEPs) are instantiated over NEPs (and contain information about the connections) and connections involve two or more CEPs.

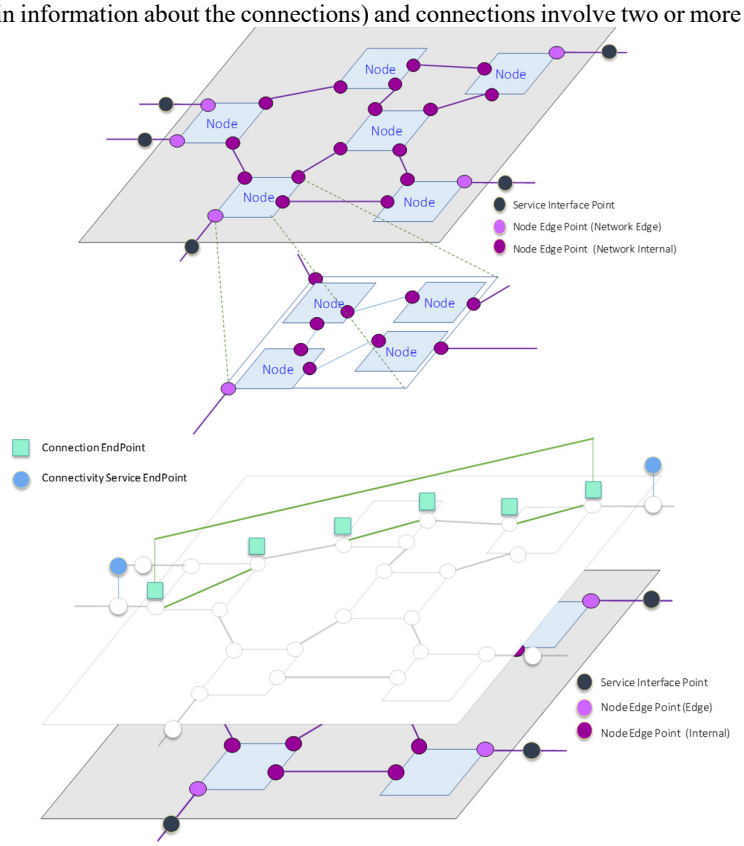

Fig. 5. Illustration of TAPI models: the notion of TAPI domain (recursive) topology [up] and the Connectivity Service / Connection concepts [bottom].

The user is able to specify, within a request, the desired protocol and layer as well as attributes and applicable constraints (e.g. path constrains, including or excluding elements etc.). Being NBI and expected to be easily consumed by users and applications, most TAPI implementations rely on RESTCONF as transport protocol.

4.2 OpenConfig Terminal Device

OpenConfig focuses on compiling a consistent set of vendor-neutral YANG data models. These data models cover a variety of network elements, from routers to optical switch. We use both the platform and the terminal-device models. The former defines a platform as having software and hardware components (such as line cards, ports, transceivers or optical channels). The latter allowing the activation of optical channels – in terms of frequency, power and operational mode – within line ports and the association (mapping) of client signals to optical channels in a quite flexible way.

4.3 OpenROADM Device Model

The device model proposed by OpenROADM is sketched in Figure 6. A ROADM is composed of a given number (N) of directions or degrees (DEG) and a given number of add/drop stages (named Shared Risk Groups or SRGs in OpenROADM terminology). A degree has in and out amplifiers and a WSS to mux/demux the signals, towards other degrees or towards add/drop stages. In simple terms, the Yang mode defines a first section related to the device information (node identifiers, vendor, model, serial number, geolocation, etc.) followed by a section that includes a list of circuit-packs, describing the physical architecture including their components ports and naming, as well as the correspondence in terms of actual racks and shelves.

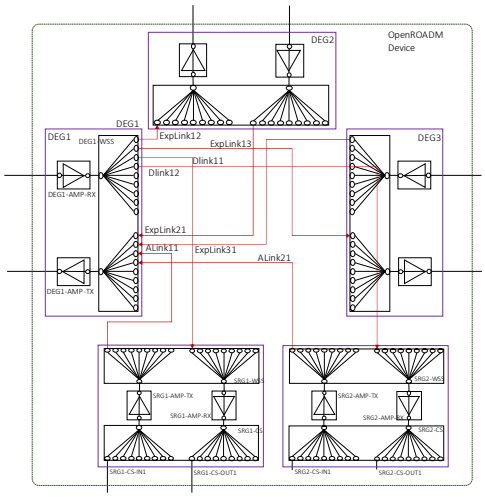

Fig. 6. Representation of an OpenROADM device as per its Yang model

The next section details the set of ROADM interfaces (which can be supported over either logical interfaces or physical ports). Another section lists the internal links (links between ports of a given component or circuit pack), the physical links (links between different components, such as a link between Degree 1 and 2, named ExpLink12 or the drop link between Degree 3 and SRG1 named DLink31) and external links (between ROADMs). In either case, links are listed keyed by their name and the associated circuit

pack ports. Next, the model also includes two lists for the main component blocks: a list of degrees, numbered, saying how many optical channels are supported as well as flexi-grid capabilities and a list of SRGs (Add / Drop groups), numbered, with a list of add drop port pairs. Finally, the roadm-connection list includes the connections that are active (that have been established) in the device. It is worth noting that each connection is identified by its name (a string, but OpenROADM papers provide guidelines on how to name them) along with two references to existing interfaces.

5 Network Operations and Service Workflow

Fig. 7. Initial Capability Discovery between the SDN controller and the OpenROADM.

Network Discovery: From the point of view of networks operation, the procedure is as follows: first, the network operator configures the SDN controller (in our case, the implementation is based on ONOS) with the list of devices, and the NETCONF credentials. This means IP addresses, Netconf ports (default is 830), user credentials, etc. The SDN controller establishes a persistent NETCONF session with the device. Let us focus on the OpenROADM devices. First, there is an initial capability exchange, in which the NETCONF client (ONOS) discovers what models and features are supported (Fig.7) using a HELLO. Next, the SDN controller issues <get> and <get-config> messages as needed to retrieve info about the devices. Typically, a <get> operation with a subtree filter of <org-openroadm-device><info/> allows retrieving basic data to add the device into the SDN controller device manager. Similar operations on the circuit packs and ports are used to retrieve internal connectivity and to discover port capabilities (Fig. 8). For example, we query the list of circuit packs and their ports, e.g. the DEGREE1 AMPRX and see the DEG1-AMPRX-IN, this is a multi-wavelength port, external, and its partner port is DEG1-AMRTX-OUT. We can also see a logical connection point, the DEG1 Trail Termination Point for RX for that degree.

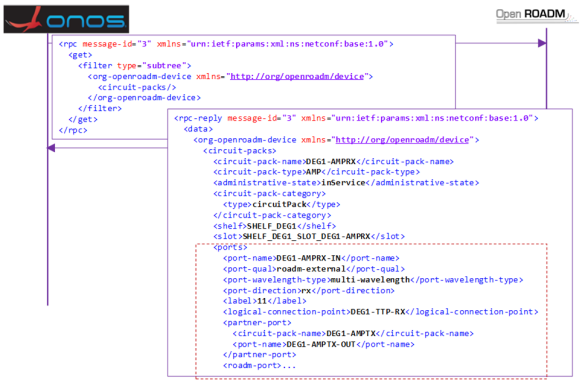

Fig. 8. NETCONF <get> operation to retrieve the list of circuit packs of the OpenROADM.

If the device supports it, we can retrieve external links to discover how OpenROADMs are inter-connected. This allows the SDN controller to construct a network topology view. If this is not supported, other means of topology discovery need to be defined, including, if need be, manual provisioning at the SDN controller. The result is that the SDN controller is aware of the network topology and end-to-end services may be requested, using, for example, the GUI (Figure 9).

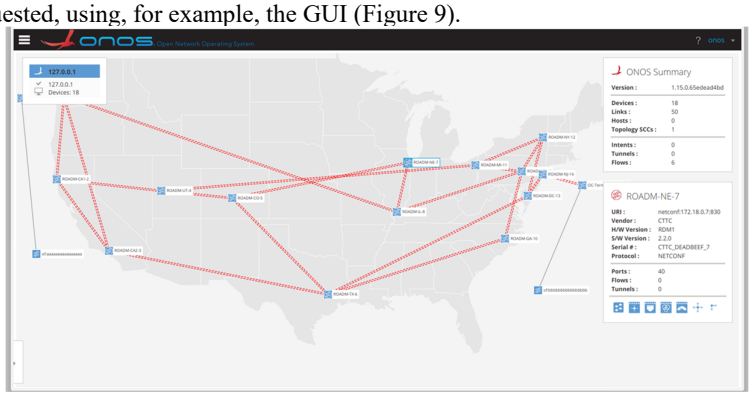

Fig. 9. ONOS GUI with NSFNET topology (14 OpenROADM and 2 OpenConfig devices).

Service Creation: In order to request a service, the user or operator retrieves the list of available Service Interface Points (SIPs) using the TAPI NBI and proceeds to request a connectivity service between a pair of SIPs (for a point-to-point connection). The request can specify if it applies e.g. to a digital signal between two transceiver client ports or, alternatively a network media channel between two transceiver line ports (or two OLS ports for an OLS controller). The SDN controller maps those SIPs to node edge points, and performs a routing and spectrum assignment process that finds the k-shortest path between the devices and performs first fit spectrum allocation. Once completed, flow and forwarding rules are configured at each device: for the Terminal Device, a logical channel association is instantiated within the device between a client

(transceiver) port and an optical channel component bound to the line port of the device. For each of the OpenROADM devices across the path, a ROADM internal connection is requested: i) OTS and OMS (optical transport and optical multiplex) interfaces are created within each degree (if not existing), ii) Supporting Media Channel and NMC interfaces are created, iii) Followed by the creation of a roadm-connection object.

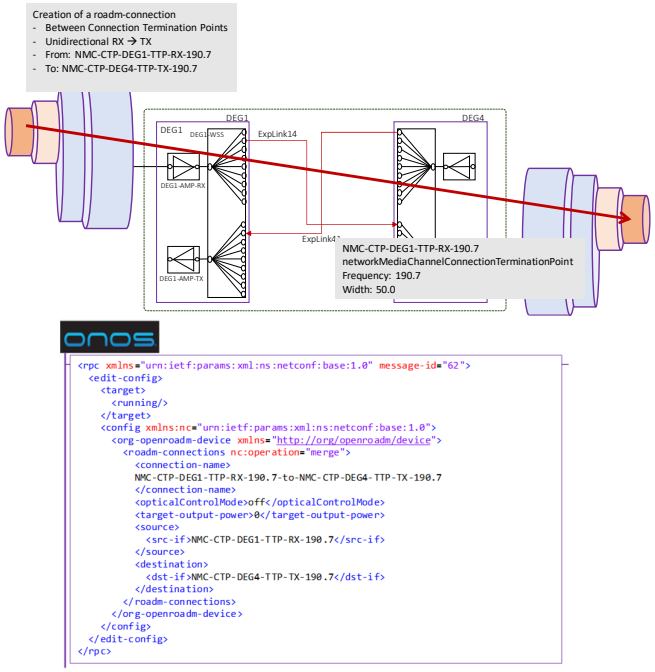

Fig. 10. Creation of a ROADM connection between two Connection Termination Points.

For example, to create a unidirectional express connection supporting a 50GHz signal centered at 190.7 GHz from degree 1 to degree 4 (Fig. 10), the SDN controller proceeds as follows. First, assume that the OTS and OMS interfaces are configured at the ROADM degrees; these interfaces correspond to the lower layers of the OTN model; OTS corresponds to the Optical Transmission Section, e.g. between amplified fiber sections and OMS to the Optical Multiplex section. Next, create Media Channel (MC) interfaces (via a <edit-config> operation) over the OMS interfaces (by convention named MC-TTP-DEG1-TTP-RX-190.7 and MC-TTP-DEG1-TTP-RX-190.7, with a min-freq: 190.675 and max-freq: 190.725, which specifies the supporting media channel. Next, create Network Media Channel (NMC) interfaces over the MC interface, specifying a center frequency and width, inducing Connection Termination Points (CTPs). Finally, establish the unidirectional connection between CTPs, from the interface NMC-CTP-DEG1-TTP-RX-190.7 to NMC-CTP-DEG4-TTP-TX-190.7. Once the cross-connections have been provisioned at all the nodes along the path and the transceivers configured with the appropriate transmission parameters, the service is active.

6 Conclusions

We have given an introduction to model driven development, the adopted framework for SDN for transport networks, leveraging on the increase of device programmability. We have introduced the main concepts behind SDN for disaggregated optical networks, presenting reference architectures and industry common practices, the use of the Yang data modelling language and the NETCONF/RESTCONF protocols, covering a subset of deployment models in partial and full disaggregation. We have focused on a clear use case for model driven development, i.e., disaggregated optical networks, with emphasis on the use of TAPI interfaces and OpenROADM device models. Even if the trend is perceived as positive, model driven development will not reach its full potential unless the framework, languages, etc. can use a set of common, agreed- upon models.

Acknowledgements We would like to thank Alessio Giorgetti (CNIT) and Roberto Morro (TIM) for contributions to the development of OpenROADM ONOS drivers. This paper has been funded by the EC through the H2020-METRO-HAUL (671598) and by the Spanish MICINN AURORAS (RTI2018-099178-B-I00) projects.

References

1. R. Casellas, R. Martinez, R. Vilalta, and R. Munoz, "Control, management, and orchestration of optical networks: Evolution, trends, and challenges," Journal of Lightwave Technology, vol. 36, no. 7, pp. 1390–1402, 2018
2. Open Networking Foundation (ONF) Technical Specification TS-025, "OpenFlow Switch Specification v1.5.1" March 26, 2015
3. Open Networking Foundation (ONF) Technical Specification TS-022, "Optical Transport Protocol Extensions", Version1.0, March 15, 2015
4. M. Bjorklund, Ed., "The YANG 1.1 Data Modeling Language", IETF Request for Comments 7950, August 2016.
5. R. Enns, Ed. "Network Configuration Protocol NETCONF", IETF Request for Comments 6241, June 2011.
6. A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF protocol," IETF Request for Comments 8040, January 2017.
7. The Open ROADM Multi-Source Agreement (MSA) http://www.openroadm.org
8. R. Morro, F. Lucrezia, P. Gomes, et al., "Automated end to end carrier ethernet provisioning over a disaggregated wdm metro network with a hierarchical sdn control and monitoring platform," in 2018 European Conference on Optical Communication (ECOC)
9. OpenConfig project and data models http://openconfig.net and https://github.com/openconfig/public/tree/master/release/models
10. E. Riccardi et al, "An Operator view on the Introduction of White Boxes into Optical Networks", JLT, v36, I15, pp3062-3072, 2018.
11. The Metro-haul project http://metro-haul.eu
12. Open Networking Foundation (ONF), The Open Disaggregated Transport Network (ODTN) project, https://www.opennetworking.org/odtn/
13. Open Networking Foundation (ONF), Transport API project https://wiki.opennetworking.org/display/OTCC/TAPI.