TransportPCE

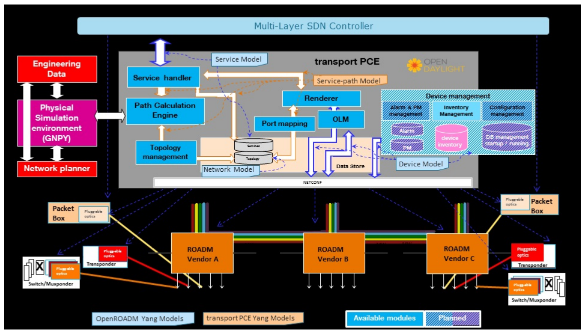
Welcome to TransportPCE

Introduction

TransportPCE describes an application running on top of the OpenDaylight controller. Its primary function is to control an optical transport infrastructure using a non-proprietary South Bound Interface (SBI). It may be interconnected with Controllers of different layers (L2, L3 Controller…), a higher layer Controller and/or an Orchestrator through non-proprietary Application Programing Interfaces (APIs). Control includes the capability to configure the optical equipment, and to provision services according to a request coming from a higher layer controller and/or an orchestrator. This capability may rely on the controller only or it may be delegated to distributed (standardized) protocols.

It provides alarm/fault and performance monitoring, but this function might be externalized to improve the scalability. A Graphical User Interface could be developed in a later step, but is not considered as a priority since automated control does not imply user interactions at the transport controller level.

TransportPCE modular architecture is described on the next diagram. Each main function such as Topology management, Path Calculation Engine (PCE), Service handler, Renderer _responsible for the path configuration through optical equipment_ and Optical Line Management (OLM) is associated with a generic block relying on open models, each of them communicating through published APIs.



The controlled transport infrastructure includes a WDM layer and an OTN layer. The WDM layer is built from ROADMs with colourless (an add/drop port is not dedicated to one wavelength, it accepts potentially any wavelength coming from a tunable transponder), directionless (an added/dropped service can be optically switched to any degree, independently of the physical port it is launched through) and possibly contention-less (no restriction on the wavelength that can be added or dropped from any port) features. The OTN layer is built from transponders, muxponders or switchponders which include OTN switching functionalities

The interest of using a controller to provision automatically services strongly relies on its ability to handle end to end optical services that spans through the different network domains, potentially equipped with equipment coming from different suppliers. Thus, interoperability in the optical layer is a key element to get the benefit of automated control.

Initial design of TransportPCE leverages OpenROADM Multi-Source-Agreement (MSA) which defines interoperability specifications, consisting of both Optical interoperability and Yang data models. North API, interconnecting the Service Handler to higher level applications relies on the Service Model defined in the MSA. The Renderer and the OLM are developed to allow configuring OpenROADM devices through a southbound Netconf/Yang interface and rely on the MSA's device model. Topology Management is also based on the Network model defined in the MSA

Project Facts

Project Creation Date: May 26 2016

Primary Contact: Gilles Thouenon <gilles.thouenon@orange.com>

Project Lead: Gilles Thouenon <gilles.thouenon@orange.com>

Committers:

- Guillaume Lambert <guillaume.lambert@orange.com>
- Christophe Betoule <christophe.betoule@orange.com>
- Balagangadhar Bathula <bb4341@att.com>
- Gilles Thouenon <gilles.thouenon@orange.com>
- Shweta Vachhani <sv111y@att.com>
- Cedric Ollivier <cedric.ollivier@orange.com>

Mailing List: transportpce-dev@lists.opendaylight.org
Archives: mailing list archives

Meetings: See Community Meetings

Repository: https://git.opendaylight.org/gerrit/q/project:transportpce

Jenkins: https://jenkins.opendaylight.org/releng/view/transportpce/

Open Bugs: https://jira.opendaylight.org/projects/TRNSPRTPCE/issues/

This choice does not prevent to extend the range of open-specifications supported by the controller, when they reach the level of maturity expected to launch heavy developments. Thanks to defined modular architecture, some additional modules dedicated to the configuration of other types of equipment could be added at a later step. Some others like the PCE or the Topology Management, less tightly coupled to the equipment models could be complemented to support the corresponding devices.

Another advantage of TransportPCE modular architecture is that, complementing Yang models defining East/West APIs that allows module communications, it could be easily interconnected to external applications, or could host specific plugins provided that they support the published APIs, avoiding to deploy Controller dedicated to specific equipment in silos.

Documentation

Getting Started for Users https://docs.opendaylight.org/projects/transportpce/en/latest/user-guide.html?highlight=transportpce

Getting Started for Developers https://docs.opendaylight.org/projects/transportpce/en/latest/developer-guide.html?highlight=transportpce

Presentations

Tech Work Stream Session 2020

TransportPCE-O...S 20200427.pdf

Santa Clara Sodium DDF 2019

ODL_Sodium_DDF...automation.pdf

San Jose Open Networking Summit North America 2019

ONSNA-Transport...l-Networks-.pdf

Amsterdam neon DDF 2018

Feedback_on_ODL_netconf.pdf

Requirements

Release Planning

Release Notes

Magnesium

- Introduction of OTN
  - OTN topology : management of OpenROADM OTN devices including switching pool
  - Creation/deletion of OTN services using East/west APIs
    - path-computation request, otn-service-path
    - 1GE/ODU0, 10GE/ODU2e, ODU4, OTU4 services
  - OTN rendering function (creation of OTN interfaces and cross connections on devices)
- T-API

    - Implementation of get-T-API-topology allows retrieving an abstracted topology derived from the openroadm-topology and otn-topology layer (Nodes and access-points in SR0)
- Device inventory
    - Experimental support of device inventory (limited to OpenROADM device 1.2.1 in Mg SR0)
- Interconnection to GNPy
    - An interconnection to GNPy is fully supported, including:
        - Topology export to GNPy tool and
        - CE Path validation Tech / impairment aware optical path calculation performed in GNPY according to specific constraints

Sodium

Sodium tPCE release focuses on code refactoring.

- Main goal is to get a robust base (fully tested) aligned with latest developments and bug corrections coming from the different contributors
- Hardened support of OpenROADM 1.2.1 and 2.2.1 releases
- Full support of ietf network topology (RFC 8345 / openROADM Network model 4.1) with consolidated topology building and portmapping functions
- CI/CD environment allowing smooth integration of contributions avoiding regression (+/-1 voting)

Neon

The main features added in Neon release are the following:

- Add support for notification in RPC handling
- Extension of the coverage for OpenROADM Service RPC handling: service-reroute, service-restoration, temp-service-create/delete
- Impairment aware path calculation in PCE (OSNR calculation)
- Management of unidirectional ports in path calculation and path configuration
- ROADM to ROADM service creation, for resource reservation when transponders are not present
- Introduction of transportpce-service-path 1.6
- Device version management (up to release 2.2.1)

Fluorine

- Provides most of the bricks defined in the controller architecture for the WDM layer
- Junit and functional tests developed for the available modules
    - Continuous integration eases collaboration between contributors in different countries, entities & companies

Side-Projects

- Honeynode: a WDM / OTN device simulator based on FD.io honeycomb and used for TransportPCE functional tests
- TransportPCE GUI:
  This GUI Provides a collapsed topological view of the OpenROADM network controlled by Transport PCE:
    - Backend interaction with TransportPCE Data store (ODL MD-SAL)
    - Collapsed view based on CLLI, Network and topology layers

  As well as a view of provisioned services
  Based on Spring boot, Spring Data and Angular

More details and source code available at Orange opensource LFN gitlab space:
https://gitlab.com/Orange-OpenSource/lfn/odl