ABSTRACTION LAYER FOR IMPLEMENTATION OF EXTENSIONS IN PROGRAMMABLE NETWORKS

Collaborative project co-funded by the European Commission within the Seventh Framework Programme

Grant agreement no: 317880
Project acronym: ALIEN
Project full title: "Abstraction Layer for Implementation of Extensions in programmable Networks"
Project start date: 01/10/12
Project duration: 24 months

Deliverable D5.3
Experimental-driven research - Appendix V (Multi-vendor evaluation of HAL in the optical domain experiment results)

Version 7.9

Due date: 30/10/2014
Submission date: 12/11/2014
Editor: Tasos Vlachogiannis (UNIVBRIS), Maider Huarte (EHU)
Author list: Tasos Vlachogiannis (UNIVBRIS).

Dissemination Level		
☒	**PU:**	Public
☐	**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)

<THIS PAGE IS INTENTIONALLY LEFT BLANK>

Table of Contents

<THIS PAGE IS INTENTIONALLY LEFT BLANK>

Figure summary

<THIS PAGE IS INTENTIONALLY LEFT BLANK>

Executive Summary

This document complements deliverable D5.3 “Experimental-driven research” with details about testing scenario steps and all materials like logs, screenshots, packet dumps collected during performing of experimentation tests. We highly encourage to read first D5.3 which contains overview of the experiment, goals to be achieved, experiment environment and also discussion about experiment validation.

<THIS PAGE IS INTENTIONALLY LEFT BLANK>

1 Multi-vendor evaluation of HAL in the optical domain

1.1 Testing scenario

In the accompanied deliverable we have briefly described the procedure followed and the goals aiming to achieve with this demonstration. Here we will provide a comprehensive overview of the experiment by providing the steps we have followed.

The steps we have gone through to run this experiment are the following:

- Starting OpenDaylight controller (ODL)
  - OpenDaylight is running in a different Linux machine and is listening for new devices to connect to it.
- Execute WSS OpenFlow datapath with accompanied configuration file, pointing to ADVA ROADMs.
  - Configuration file contains the management interface of the switch, as well as peering information of the managed switch since there is no protocol similar to Link Layer Discovery Protocol (LLDP)
- Execute Fibre switch OpenFlow datapath, with config file pointing to Polatis fibre switch.
  - Configuration file again contains management interface of the switch and peering information.
- OpenFlow datapaths are sending HELLO messages to the controller and get discovered by ODL.
- ODL controller requests and receives the features of the switch (FEATURES_REQUEST) message.
  - ODL is modified so that is can interpret Circuit extension OpenFlow messages.
- ODL receives FEATURES_REPLY messages from the switches. This is the way ODL registers the switches. Also from the same message ODL parses the peering information and discovers the topology of the optical domain.
- User installs new cross-connection using ODL web interface.
- User tears down existing cross-connection using ODL web interface.

In the next section follows a more detailed description is given by providing the results of these steps backed with the output of the demonstrated software components given as screenshots while performing the experiment.

1.2 Experiment results

Starting with the OpenDaylight controller view we are showing the devices discovered by the SDN controller. Indeed we notice that OpenDaylight is able to discover both the types of optical devices. The process of discovering requires the switch agent to send a HELLO message to the controller and gets a HELLO as a reply. At this point the controller and switch are exchanging the OpenFlow version supported by each entity. The version communicated by the optical agents is 1.0 since there is no official support for optical devices until OpenFlow v1.4. Following the controller sends a FEATURES_REQUEST message to the switch to discover the features of the switch and the latter replies with a FEATURES_REPLY message to the controller. OpenDaylight controller also sends a GETCONF_REQUEST and FLOW_MOD message but these are ignored by the optical switches since they are meant only for packet switches. Finally the agent sends a FEATURES_REQUEST message every 12 seconds and gets an ECHO_REPLY message from the controller to maintain the OpenFlow control channel active.

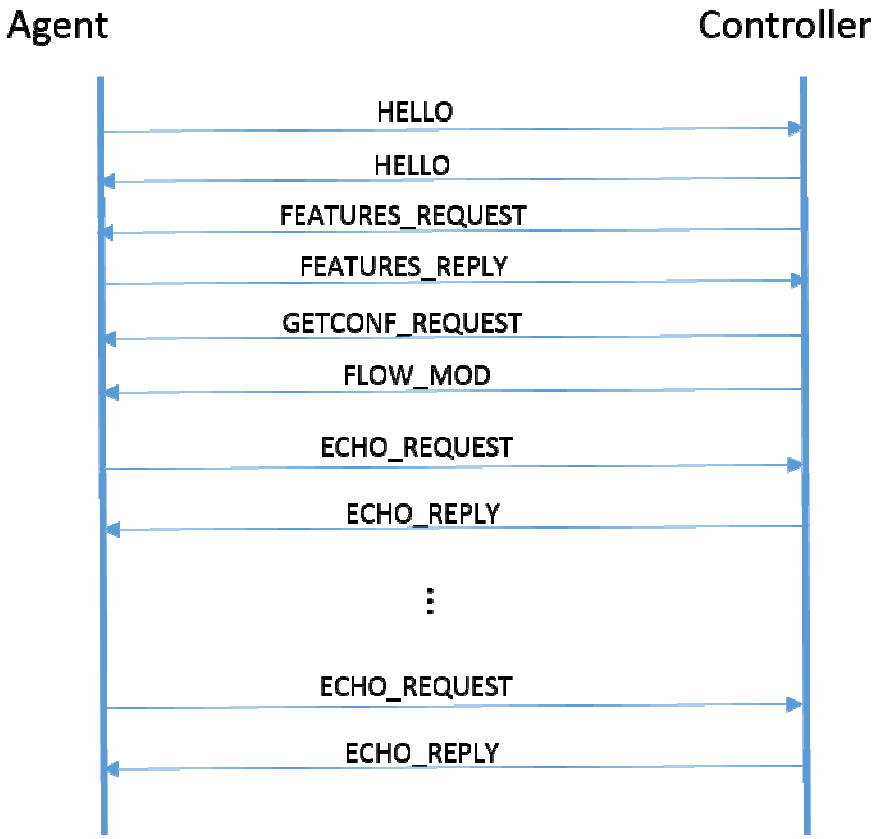

Figure 1-1 OpenFlow messages exchange

At this point the controller and switch are exchanging the OpenFlow version supported by each entity. The version communicated by the optical agents is 1.0 since there is no official support for optical devices until OpenFlow v1.4. Following the controller sends a FEATURES_REQUEST message to the switch to discover the features of the switch and the latter replies with a FEATURES_REPLY message to the controller. OpenDaylight controller also sends a GETCONF_REQUEST and FLOW_MOD message but these are ignored by the optical switches since they are meant only for packet switches. Finally the agent sends a FEATURES_REQUEST message every 12 seconds and gets an ECHO_REPLY message from the controller to maintain the OpenFlow control channel active.

The figures below show the discovered topology after FEATURES_REPLY messages have been parsed by the controller. The second view has some more details about the discovered nodes like the type of each switch and number of ports.

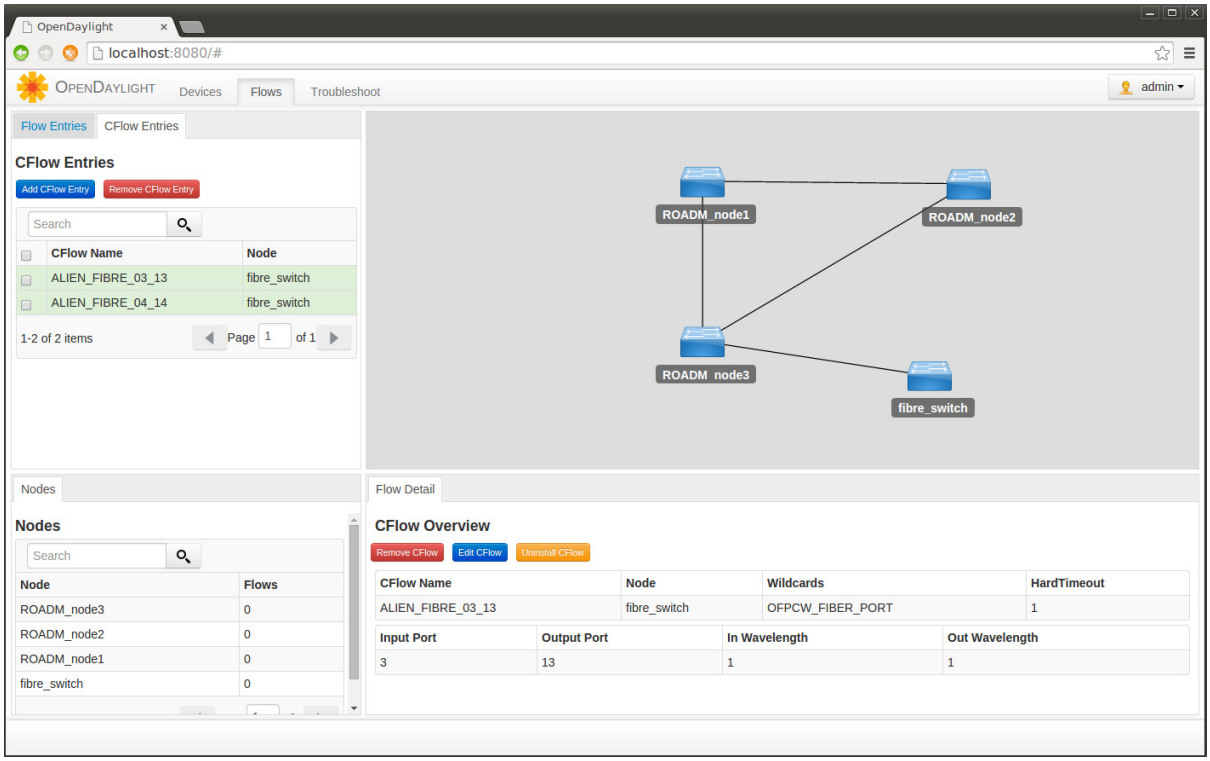

Figure 1-2 Discovered topology - OpenDaylight controller main view

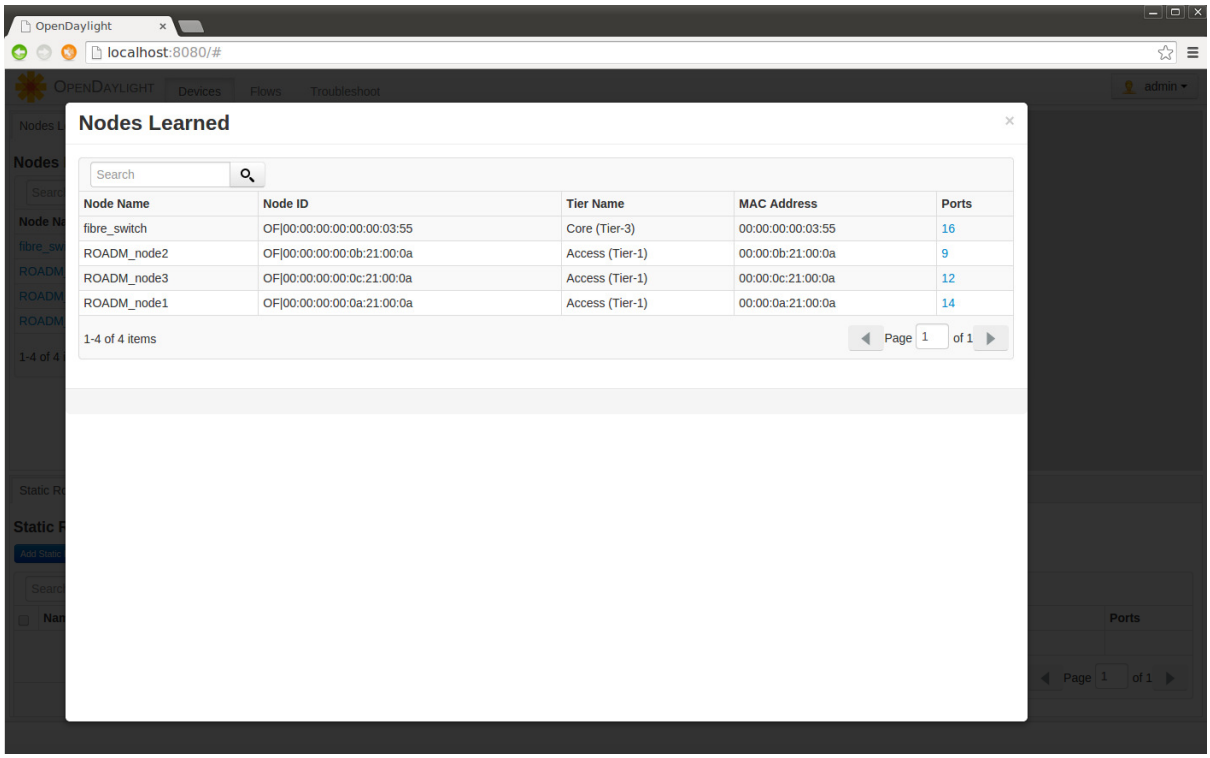

Figure 1-3 Detailed view of discovered switches - OpenDaylight controller

The next figure is taken from OpenDaylight controller's web interface shows the extensions made to install a new cross-connection in the optical switch. First we need to define the node that we want to use and then the input and output port between which we will perform the cross-connection.

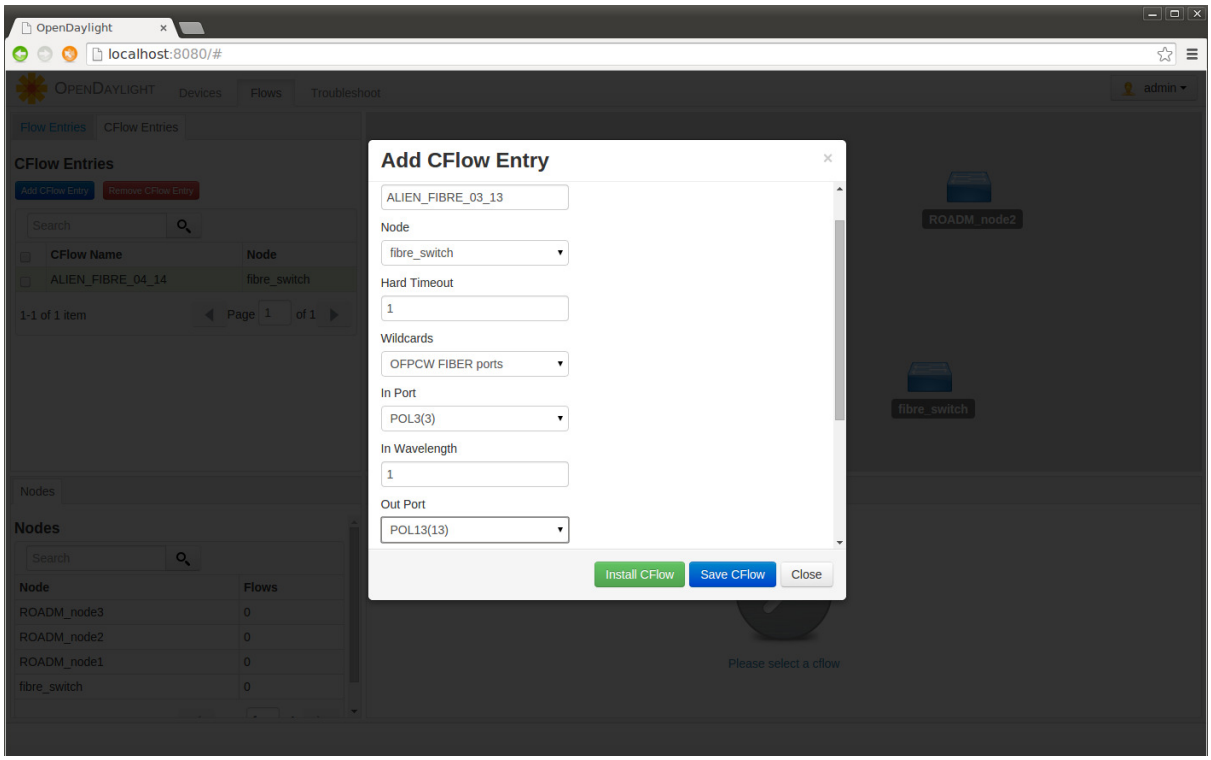

Figure 1-4 Create new cross-connection using OpenDaylight controller

The message is received by the OpenFlow agent and parsed to discover the attributes of the CFLOW_MOD message. The switch needs to perform a cross connection between fibre ports 3 and 13. The agent sends the appropriate TL1 commands to the switch and we see that the cross-connection was successful.

```
Terminal - root@gblallen-VirtualBox: /home/gblallen/ALIEN/rofl-polatis/examples/polatisswitch
File  Edit  View  Terminal  Tabs  Help
ROFL [handle_message]:OpenFlow (1) message received: 3 (bytes 2048)
ROFL [handle_revent]:#### csocket(0xa062c70)::handle_revent()

ROFL [handle_message]:OpenFlow (1) message received: 9 (bytes 3072)

ROFL [handle_message]:SET_CONFIG message received!
ROFL [handle_revent]:#### csocket(0xa062c70)::handle_revent()

ROFL [handle_message]:OpenFlow (1) message received: 7 (bytes 2048)

ROFL [handle_message]:GET_CONF_REQUEST message received!
ROFL [handle_revent]:#### csocket(0xa062c70)::handle_revent()

ROFL [handle_message]:OpenFlow (1) message received: 22 (bytes 13312)

ROFL [handle_message]:CFLOW_MOD message received. Parse frame to construct a new cflow_mod message.
AGENT: Received CFLOW_MOD message with type 22
AGENT: length:52


AGENT: cross-connect: 3<==>13 [0]
AGENT: perform cross connection between ports: 3<==>13
[00:0][03:3][0d:13]sending XC command: ENT-PATCH::3,13:123:;
Client:Message rcv:

   ( nil ) 12-03-13 23:21:18
M  123 COMPLD
;

XC success
AGENT: success!
AGENT: end.
ROFL [handle_revent]:#### csocket(0xa062c70)::handle_revent()

ROFL [handle_message]:OpenFlow (1) message received: 18 (bytes 2048)

ROFL [handle_message]:BARRIER REQUEST received.
AGENT: Handle Barrier request message from polatissw.
... ####
```

Figure 1-5 Handle CFLOW_MOD –Install new cross-connection

Here is the proprietary web-interface of the Polatis switch that shows the cross-connection between ports 3 and 13 was successful.

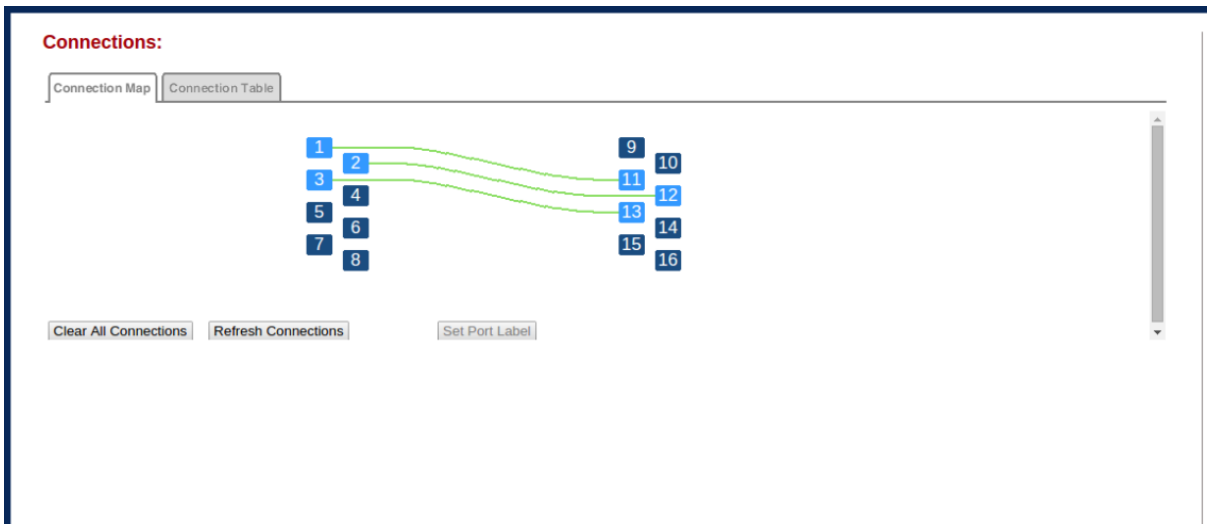

Figure 1-6 Polatis web interface -cross-connection table after installing flow

Finally we are removing the flow we had installed before in the switch by selecting the flow and pressing the “Remove CFlow” red button in OpenDaylight controller as shown in figure 28. Then a CFLOW_MOD message will by send to the OpenFlow agent as shown in the following screenshot.

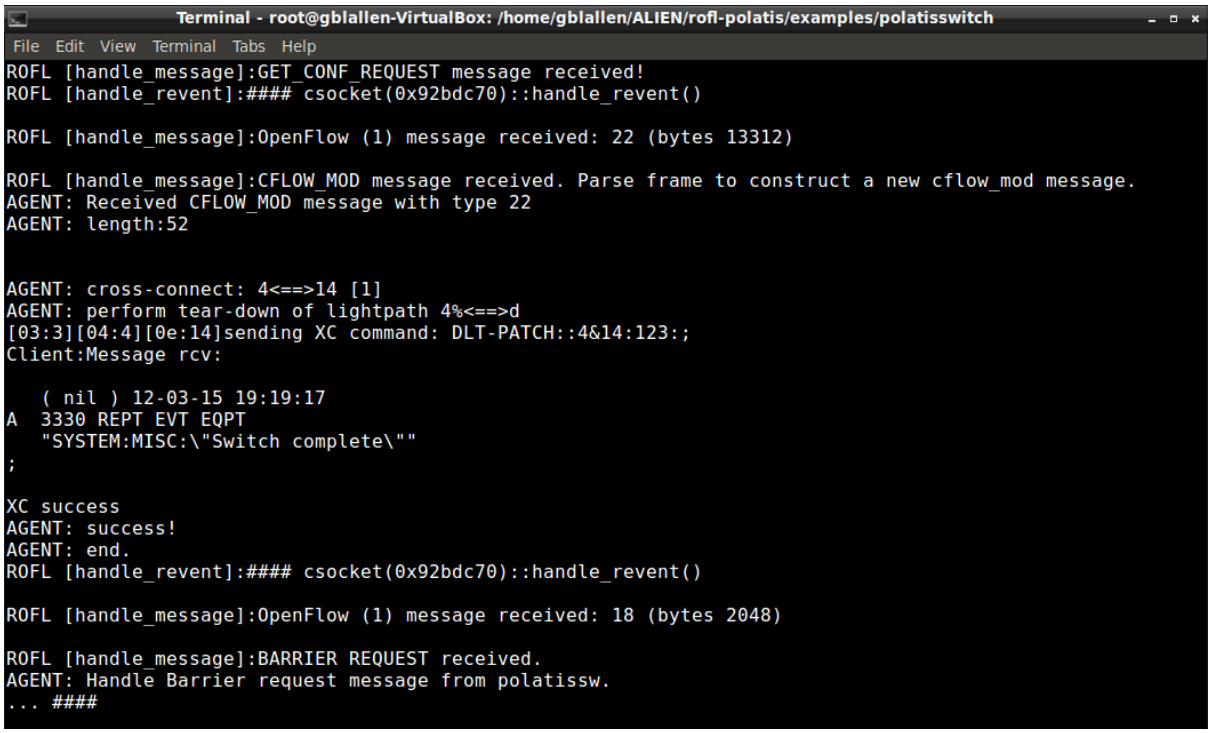

Figure 1-7 Handle CFLOW_MOD – Remove existing cross-connection