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 D5.2
CCN and ALIEN developments integration over OFELIA

Version 11.2

Due date:	31/07/14
Submission date:	22/09/14
Editor:	Maider Huarte (UPV/EHU)
Internal reviewers:	Jon Matías, Eduardo Jacob (UPV/EHU)
Author list:	Eduardo Jacob, Victor Fuentes, Jon Matías, Maider Huarte (UPV/EHU); Damian Parniewicz, Łukasz Ogrodowczyk, Bartosz Belter (PSNC); Tasos Vlachogiannnis (UNIVBRIS); Richard G. Clegg (UCL); Marek Michalski, Mairus Zal (PUT); Marc Bruyere (DELL/FORCE)

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>

Abstract

The Alien HAL allows hardware that was previously not OpenFlow compatible to be placed under an OpenFlow control framework. In this deliverable, the integration of these developments and the CCN experiment under OFELIA, and resulting conclusions will be shown. Integration under the OFELIA Control Framework, represents an additional validation of the developments done in the project. It is important to remind that every partner has led the development of a kind of equipment and also that the connection of its laboratory to OFELIA is specific. In the second section, for each of the developments, a description of the specific OFELIA integration setup, the testing done and conclusions will be presented. In the third section the deployment of the CCN experiment, namely CONET, under OFELIA and over the ALIEN developments will be shown. Finally some conclusions will be presented.

<THIS PAGE IS INTENTIONALLY LEFT BLANK>

Table of Contents

Figure Summary

Table Summary

Executive Summary

After testing the ALIEN HAL developments at each partner site locally, functional tests have been carried out to demonstrate that they actually work integrated in OFELIA, along with a CCN under OFELIA application experiment.

The integration under the OFELIA Control Framework testing is an additional validation of the developments done in the project, where every partner has led the development of a different kind of equipment in a different context, as the connection of the facilities of each partner to OFELIA is specific. In the second section, each partner has explained their experiment, giving a description of the specific OFELIA integration setup, the testing done and the conclusions from their experience. The descriptions of the OFELIA integration setups are written around two main points, the connection to OFELIA (which, in some cases, implies a whole new OFELIA island) and the integration of the developments themselves. The tests are explained with the OFELIA resources involved, the workflow steps and the results (given with screenshots and/or logs where possible). The conclusions of the tests present the functionalities that were successfully demonstrated, as well as the lessons learned thanks to the experimentation in real OFELIA environments, as opposed to the local tests previously run.

The CONET experiment, where the CCN application has been executed under OFELIA, is shown in the third section of this deliverable. The CONET experiment allows validating HAL-enabled devices, like the DOCSIS ALIEN device, by using a third party application (i.e. CCNx) adapted to the OpenFlow environment, i.e. CONET (an application that generates both data plane and control plane traffic).

This deliverable reports the outcomes of integration of ALIEN resources with OFELIA. This complex environment, covering OFELIA and ALIEN islands, will be used as a base for final experiments to be conducted within WP5 and reported in D5.3.

This deliverable is targeted mainly at network engineers and technical teams working in the SDN domain. The document contains low-level technical details of the implementation of new OFELIA islands with non-OpenFlow equipment, complemented with conclusions and lessons learned from this process. The results reported in this deliverable may help other network administrators in joining OFELIA or other existing SDN-enabled network infrastructures.

<THIS PAGE IS INTENTIONALLY LEFT BLANK>

1 Introduction

The Alien HAL allows hardware that was previously not OpenFlow compatible to be placed under OpenFlow control.

In the context of the validation of the developments done under ALIEN, OpenFlow compliance tests (with the use of OFTest) were run as part of the software creation and delivery process. In this deliverable, their integration under the OpenFlow control interface, which is an additional step in the validation of the compatibility of the work done in the project, is presented.

This integration is run in a tight parallel process which involves setting up the connection to OFELIA infrastructure, integrating the development under its control and running a specific test application which shows the achievement of the work done. This gives an additional proof of the functionalities of the development and helps to detect glitches or problems that could be undetected with a pure protocol testing.

It is important to remind that each partner has led the development of the ALIEN HAL for a kind of equipment and also that the connection of its laboratory to OFELIA is specific. There are partners that were originally part of OFELIA (UNIVBRIS, CREATE-NET and EICT), others have decided to directly link their development to an existing OFELIA island (UCL and DELL) and, finally, some others have decided to fully comprehend the details and setup a new OFELIA island (EHU, PSNC and PUT). In the case of EHU, the connection to OFELIA is done through i2Cat, and in the PSNC and PUT it is done through UNIVBRIS. This is the reason for describing the situation of every development.

In section 2, for each of the developments, a description of the specific OFELIA integration setup, the testing done and conclusions will be presented. In section 3 the deployment of the CCN experiment, namely CONET, over OFELIA and with the use of some of the developments will be shown. Finally some conclusions will be presented in section 4.

<THIS PAGE IS INTENTIONALLY LEFT BLANK>

2 Integration of developments in OFELIA

In this section the integration of ALIEN developments in OFELIA is described. For each of the developments, the same information is provided: a description of the integration, which involves describing the process to link the equipment to OFELIA, the details of the configuration of the equipment and how the control plane is extended from OFELIA to the newly integrated hardware is presented; later, a description of the experiment devised to show the functional performance is given; finally, some conclusions are also presented.

It is important to understand that, in the same way the connection of different partners to OFELIA is specific, the equipment object of the integration is different and thus, it is not possible to just use the same experiment for every case.

2.1 Integration of DOCSIS

In this section the integration of the DOCSIS development in OFELIA is shown. EHU decided to create an OFELIA island and to link it to OFELIA through a dedicated 1Gbps layer 2 link provided by Spanish NREN RedIRIS to the i2Cat OFELIA Island. It is important to note that this raw bandwidth is compatible with the uplinks of the CMTS equipment used. This task would not have been possible without the help of i2Cat and iMinds OFELIA teams. The fact that OFELIA project is not funded anymore, gives even more value to these teams' work, and also explains why at some time there have been some delays. Just to put everything in context, the development shown here implements an access network that spawns from the aggregation switch to the end user. This access network technology has the possibility to offer and enforce different service flows with different bandwidth to support different applications. The work done is described in this paper [ALHINP].

2.1.1 Description of the integration

This section describes how the EHU resources and DOCSIS ALIEN have been integrated in OFELIA. Firstly, the EHU OFELIA Island is described in detail, showing the available resources and how the control and data planes have been connected to OFELIA. Once the EHU Island was deployed, the DOCSIS ALIEN was connected (both control and data planes) to OFELIA in two different setups, depending on the tests to be performed.

1) EHU OFELIA Island

The EHU OFELIA Island comprises computational and OpenFlow-enabled datapath resources, which are exposed to OFELIA experimenters through the appropriate Aggregate Managers configured in the EHU's OFELIA Control Framework (OCF located at https://10.216.64.4/).

The EHU Island resources are shown in Figure 2-1:

- 1 DOCSIS ALIEN device exposed as OF1.0 switch (ALIEN OpenFlow datapath).
- 1 NEC IP8800 OF1.0 switch (OpenFlow datapath).
- 2 Servers for deploying Virtual Machines (Computation resources).

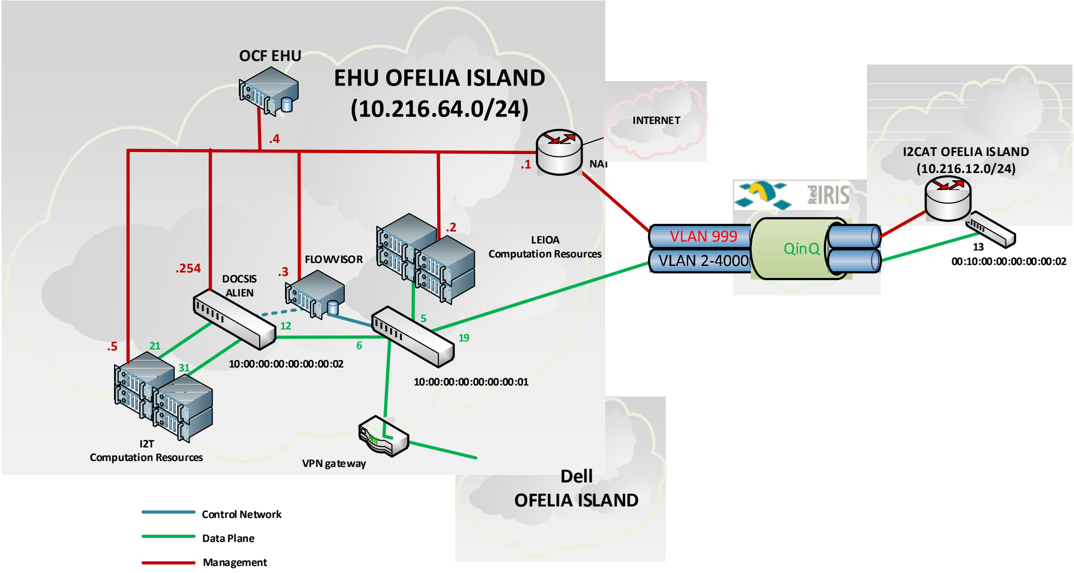


Figure 2-1 EHU/i2Cat Islands configured for DOCSIS Integration

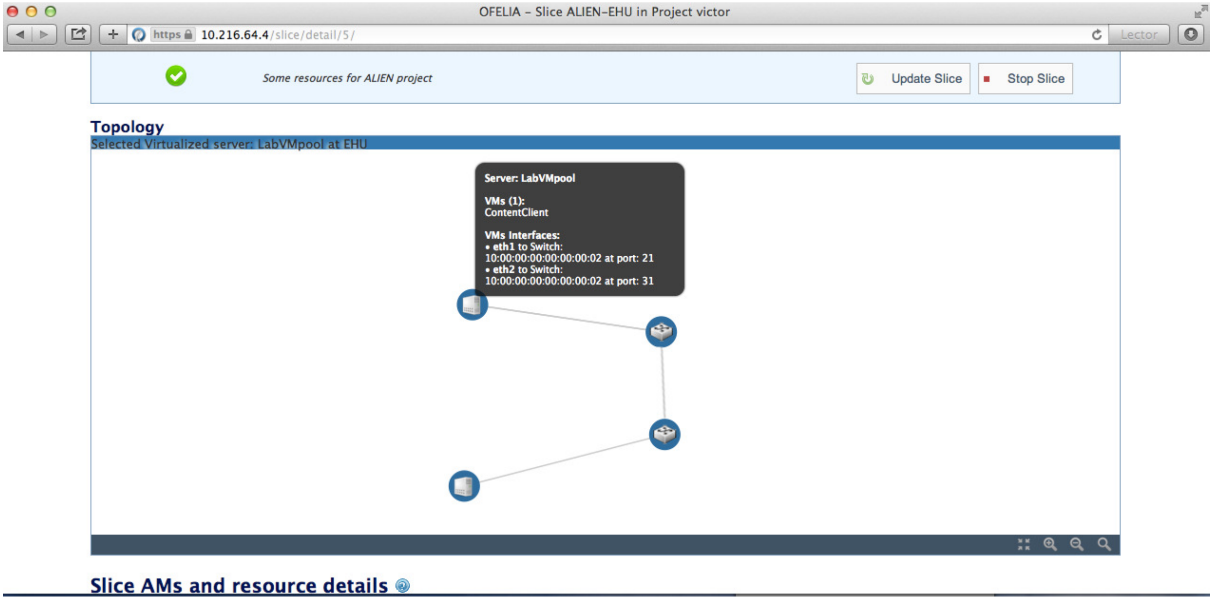
The EHU's OCF manages the resources exposed to OFELIA by using two different AM: OFAM for OpenFlow resources and VTAM for computational resources. Both Aggregate Managers are available for any other OCF in OFELIA by using the following URIs:

```
OFAM: https://10.216.64.4:8443/xmlrpc/xmlrpc/
VTAM: https://10.216.64.4:8445/xmlrpc/plugin/
```

The OFAM manages the EHU FlowVisor (IP address 10.216.64.3), which is responsible for controlling the OpenFlow datapaths exposed to OFELIA by EHU, i.e. NEC switch and DOCSIS ALIEN. The following commands are executed in the FlowVisor, and show the two DPIDs connected to it and the characteristics (e.g. number of ports and list of port identifiers) of each DPID. The first one (DPID 10:00:00:00:00:00:00:01) is the NEC switch exposing 21 ports, whereas the second one (DPID 10:00:00:00:00:00:00:02) is the DOCSIS ALIEN which exposes 3 ports: the uplink (port 12) and two clients (cable modems) currently connected to the DOCSIS platform (ports 21 and 31).

```
root@vmfv:~# fvctl list-datapaths
Password:
Connected switches:
1 : 10:00:00:00:00:00:00:01
2 : 10:00:00:00:00:00:00:02

root@vmfv:~# fvctl list-datapath-info 10:00:00:00:00:00:00:01
Password:
{
  "connection": "/10.216.192.222:6633-->/10.216.192.193:61935",
  "current-flowmod-usage": {
    "0680c092-dbe8-4a3a-bc0a-ec11b62d9cdb": 2,
    "b474443d-6215-44ea-a511-05d925b3ebdc": 0,
    "fvadmin": 0
  },
  "dpid": "10:00:00:00:00:00:00:01",
  "num-ports": 21,
  "port-list": [
    1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
25, 26
  ],
  "port-names": [
    "GBE0/1", "GBE0/2", "GBE0/3", "GBE0/4", "GBE0/5", "GBE0/6", "GBE0/7",
"GBE0/8", "GBE0/9", "GBE0/10", "GBE0/11", "GBE0/12", "GBE0/13",
"GBE0/14", "GBE0/15", "GBE0/16", "GBE0/17", "GBE0/18", "GBE0/19",
"10GBE0/25", "10GBE0/26"
  ]
}

root@vmfv:~# fvctl list-datapath-info 10:00:00:00:00:00:00:02
Password:
{
  "connection": "/10.216.64.3:6633-->/10.216.64.254:44350",
  "current-flowmod-usage": {
    "0680c092-dbe8-4a3a-bc0a-ec11b62d9cdb": 0,
    "b474443d-6215-44ea-a511-05d925b3ebdc": 0,
    "fvadmin": 0
  },
  "dpid": "10:00:00:00:00:00:00:02",
  "num-ports": 2,
  "port-list": [
    31, 21, 12
  ],
  "port-names": [
    "",
    ""
    ]
}
```

The following screenshot Figure 2-2 was automatically generated by the EHU's OCF and shows the resources (and information about them) and the topology of the EHU OFELIA Island.


OFELIA – Slice ALIEN–EHU in Project victor
https 10.216.64.4/slice/detail/5/
Lector
Update Slice
Stop Slice

Some resources for ALIEN project

Topology

Selected Virtualized server: LabVMpool at EHU

Server: LabVMpool

VMs (1):
ContentClient

VMs Interfaces:
• eth1 to Switch:
10:00:00:00:00:00:00:02 at port: 21
• eth2 to Switch:
10:00:00:00:00:00:00:02 at port: 31

Slice AMs and resource details

Figure 2-2 Screenshot from EHU OCF showing the resources and topology of EHU Island

Regarding the control plane and data plane connection of the EHU Island to OFELIA, this is done through the i2Cat Island. This is the first time that the same layer 2 link (1G) is used to connect both control and data planes between two islands. VLAN identifiers are used to distinguish between the control (VLAN 999) and data traffic (VLANs 2-998 and 1000-4000). This is a quite complex scenario and requires Q-in-Q support in the link between both entities provided by RedIRIS. Instead of using Internet to connect the EHU control plane to iMinds Island and then to i2Cat Island (most common setup), this setup enables a direct control connection between EHU and i2Cat, and as a consequence, the latency is reduced and the dependency with iMinds is removed. A second control plane connection to iMinds can be configured for redundancy (this connection has not been deployed yet).

In addition to this connection between the EHU and i2Cat Islands, the EHU Island is also providing data plane connection to the DELL Island. This means that the data plane of the DELL Island is connected to OFELIA through the EHU Island. A layer 2 tap-based VPN is established by using OpenVPN software (the same software is used by OFELIA to provide access to the resources to the experimenters). More details about this connectivity are provided in the Integration of Cavium section.

Control plane connection to OFELIA through i2Cat Island

OFELIA delegated two /24 subnets to EHU for control and management purposes, i.e. 10.216.64.0/24 and 10.216.192.0/24. The EHU Island makes use of the 10.216.64.0/24 subnet to interconnect all the EHU resources to the control plane. A router (10.216.64.1) is properly configured to announce this subnet to OFELIA. The EHU router is directly connected to the i2Cat router by a /30 network (10.216.192.248/30). A NAT interface has been also added in the router to provide an Internet connection for the VMs created and the Computation Resources Pool (i.e. the 10.216.64.0/24 subnet) by using the EHU campus resources.

All the control plane traffic (i.e. 10.216.64.0/24 subnet) is tagged with VLAN id 999 and sent to i2Cat Island by using a Q-in-Q encapsulation through RedIRIS. They are using the same VLAN identifier to carry both control (VLAN 999) and data plane traffic (VLAN 2-998 and 1000-4000) to i2Cat.

As previously mentioned, the control configuration takes into account the two islands configuration and it is quite complex. In the following QUAGGA router configuration an overview of the different networks involved is shown.

```
vmquagga# sh run
Building configuration...
Current configuration:
!
log syslog notifications
log facility local0
!
interface eth0
description I2T Island Management
ip address 10.216.64.1/24
ipv6 nd suppress-ra
!
interface eth1
description I2CAT XC
ip address 10.216.192.250/30
ip ospf authentication message-digest
ip ospf cost 5
ip ospf message-digest-key 1 md5 xxxxxx
ipv6 nd suppress-ra
!
interface eth2
ipv6 nd suppress-ra
! interface lo
ip address 10.216.192.255/32
!
router ospf
ospf router-id 10.216.192.255
passive-interface default
no passive-interface eth1
network 10.216.64.0/24 area 0.0.0.16
network 10.216.192.248/30 area 0.0.0.16
network 10.216.192.255/32 area 0.0.0.16
!
ip route 0.0.0.0/0 10.216.192.245
ip route 10.0.0.0/8 Null0 254
ip route 172.16.0.0/12 Null0 254
ip route 192.168.0.0/16 Null0 254
!
ip forwarding
ipv6 forwarding
!
line vty
!
end
```

In the EHU Island, some IP addresses from the delegated subnet are statically assigned to some resources. The following summarizes such assignment, and is relevant to properly understand some of the captures presented in this section.

```
10.216.64.1        ROUTER
10.216.64.2        OXA (LEIOA)
10.216.64.3        FLOWVISOR
10.216.64.4        OCF (EXPEDIENT)
10.216.64.5        OXA (I2T)
10.216.64.15-35    LEIOA Computation Resources
10.216.64.200-220  I2T Computation Resources
10.216.64.254      DOCSIS ALIEN if FLOWVISOR is bypassed (for TESTING purposes)
```

Data plane connection to OFELIA through i2Cat Island

The EHU data plane is connected to OFELIA through a layer 2 link (1 Giga) between EHU and i2Cat provided by RedIRIS. In order to support multiple experiments simultaneously running on OFELIA infrastructure, all the data traffic is tagged with a VLAN tag (the identifier must be one from the VLAN range assigned to the experiment). This means that each experiment uses a different VLAN and this tag must not be removed or changed when crossing from one island to another. As a consequence, the data plane connection between EHU and i2Cat must support that the VLANs transparently cross from one site to the other. Thus, Q-in-Q support is a must for the connection provided by RedIRIS.

Generically, the VLANs from 2 to 4000 are used in OFELIA for tagging experiments at data plane traffic. Since the VLAN 999 is used by i2Cat to tag the control traffic, this VLAN is reserved and not assigned to any experiment. Then, all the VLANs from 2 to 998 and from 1000 to 4000 are transported transparently by RedIRIS (by Q-in-Q) from one site to the other.

The two edge OpenFlow datapaths that interconnect the EHU and i2Cat Islands are:

EHU	DPID 10:00:00:00:00:00:00:01 port 19.
i2Cat	DPID 00:10:00:00:00:00:00:02 port 13.

This information must be configured in the OCFs to show the inter-island connectivity information to the experimenters of OFELIA. Such information is crucial to properly request the appropriate OpenFlow resources to setup an end to end path between these islands.

2) DOCSIS ALIEN integration in EHU Island

Previously, it was shown how the DOCSIS ALIEN can be connected at control plane to the EHU FlowVisor in order to expose the DOCSIS platform through the OCF as a resource to other experimenters. This is possible because in this setup the DOCSIS ALIEN is exposing OpenFlow 1.0, which is the only version supported by FlowVisor. To overcome this limitation, different setups are defined for the DOCSIS ALIEN device integration in EHU.

The first setup is the one that has been already explained. The DOCSIS development is put under the control of the OFELIA's FlowVisor, and thus, the access to this development is enforced with the OFELIA Control Framework.

The second setup gets rid of the FlowVisor control and directly exposes the DOCSIS ALIEN control plane to a previously defined controller. This controller must be reachable from the OFELIA control network (any resource at OFELIA is candidate to control the DOCSIS). In this case, the IP address of the controller must be provided to the DOCSIS ALIEN proxy and the DOCSIS ALIEN can be also accessible directly through its IP address instead of through Island's FlowVisor. The only limitation is that the whole DOCSIS platform is assigned to just one experiment. The main benefit is that the DOCSIS ALIEN can expose OpenFlow versions beyond 1.0 (e.g. 1.2 or 1.3). Moreover, OpenFlow extensions can be also tested, which effectively implies that any specific extensions can be used (based on a direct connection between DOCSIS ALIEN and its controller), for example to make use of the different bandwidths available in the access network. This setup is reflected in previous figure (Figure 2-1) with the use of the dotted lines.

The two already described setups differentiate at control plane integration; however, in both scenarios the data plane remains the same. The data plane connection details are the following:

• NEC	DPID 10:00:00:00:00:00:00:01 port 6.
• DOCSIS ALIEN	DPID 10:00:00:00:00:00:00:02 port 12.

This information is relevant to properly provide data plane connectivity between the DOCSIS ALIEN device (and possible clients behind it) to the EHU Island (as well as other OFELIA islands).

2.1.2 Functional testing

The functional testing of DOCSIS ALIEN device was performed demonstrating how this device can be used as any another OpenFlow resource integrated into EHU OFELIA Island to support the creation of flows between a client and a server, all of them installed by an standard OpenFlow controller like POX located in another island.

The HAL-based DOCSIS Proxy is the key component that enables the entire DOCSIS network and its helpers (i.e. aggregation helper and residential gateway helper) to be exposed as a unique OpenFlow switch, which is able to communicate with the OpenFlow controller, and to perform actions requested by the controller, presenting the same behaviour as a single OpenFlow switch.

Instead of developing a specific application for the POX, it was decided to run the learning switch application that comes with the POX installation. Afterwards, it was also tested the learning switch that comes with the FloodLight controller. By doing so, it is ensured that the test performed with a third party application is more neutral than a control application built by DOCSIS ALIEN developers. The learning switch application (control plane application) is independent from the end to end application (data plane application) running between the end nodes. This means that a video transfer, scp (secure copy), ssh (secure shell) or even a ping application between end nodes (data plane) requires similar support from the OpenFlow perspective at control plane (i.e. packet-in, packet-out and flowmod). For simplicity, the learning switch application (POX and FloodLight) was tested with ping and scp.

Apart from the learning switch application, the DOCSIS ALIEN was also tested with another third party application, CONET application, as later described in the next 3 CONET experiment section

In order to demonstrate the integration of the DOCSIS ALIEN device in OFELIA, one end host was located behind the DOCSIS platform at EHU Island and the other end host was located at i2Cat Island, which demonstrates the data plane integration in OFELIA. The POX controller (and FloodLight controller in the second test) was also located at i2Cat and directly controlled the DOCSIS ALIEN device located at the EHU Island, which also demonstrates the control plane integration in OFELIA.

It must be highlighted that the integration of the DOCSIS ALIEN device (both control and data planes) in OFELIA was demonstrated at TNC2014 (Dublin, 19-22 May 2014). In that demo shown in Figure 2-3, the POX controller located at i2Cat Island enabled a broadcast flow for video traffic and a bidirectional unicast flow for a ssh connection between a client located behind the DOCSIS platform at EHU and a server located at i2Cat. The controller generated the required flowmods to establish the path between the client and the server for each type of flow.

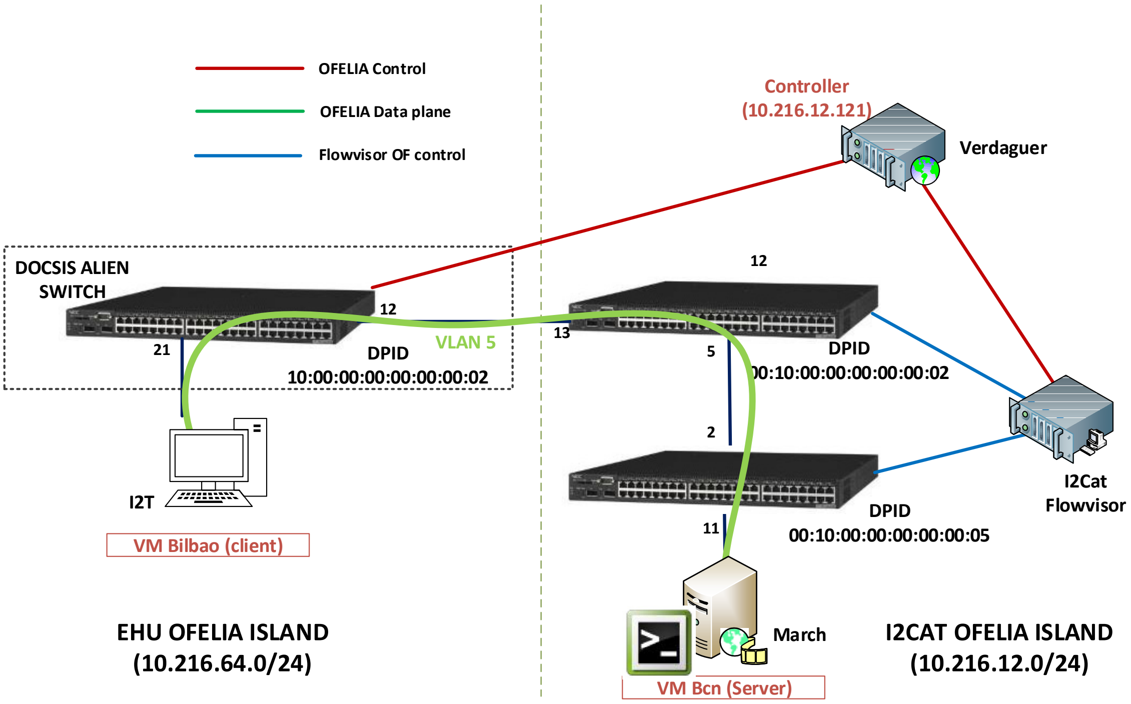


Figure 2-3 Setup of demo at TNC2014 demonstrating the DOCSIS ALIEN integration in OFELIA

Currently, the DOCSIS ALIEN developers are working on enabling the support for providing at the DOCSIS ALIEN a different bandwidth depending on the traffic (e.g. some flows with a low bandwidth and some others with a high bandwidth). They are working on OpenFlow extensions to expose that functionality (i.e. setting the bandwidth) to the OpenFlow controller. The results from this work will be detailed in D5.3.

1) OFELIA resources involved in functional testing

As previously mentioned, the integration of DOCSIS ALIEN device in OFELIA and the validation of its OpenFlow implementation are performed using resources from two OFELIA islands:

- EHU Island:
  - 1 DOCSIS ALIEN device exposed as OF1.0 switch.
  - 1 NEC IP8800 OF1.0 switch.
  - 1 VM end host (data plane) deployed in the server behind the DOCSIS platform.

- i2Cat Island:
  - 2 OF switches.
  - 1 VM end host (data plane).
  - 1 VM OpenFlow POX controller (control plane).

The following Figure 2-4 shows the computation and OpenFlow resources (including the DOCSIS ALIEN) involved in the functional testing and the actual topology deployed for the test.

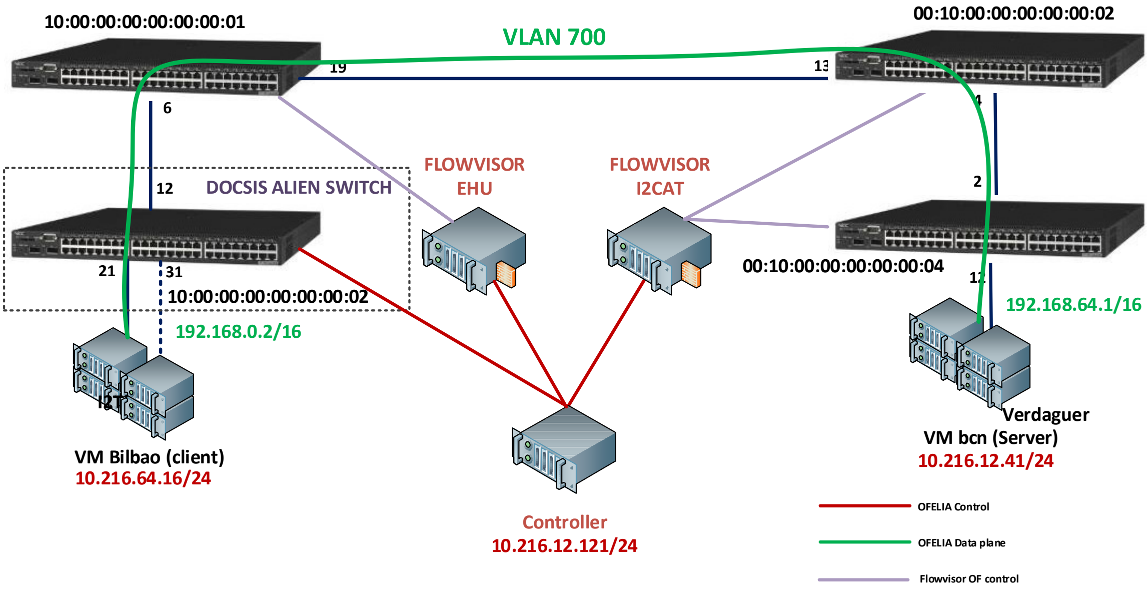


Figure 2-4 Setup for functional testing of DOCSIS ALIEN integration

2) Testing workflow

The functional testing performed to validate the DOCSIS ALIEN integration in OFELIA and the HAL implementation for DOCSIS Proxy can be represented as a workflow with several steps defined by the learning switch application of POX controller. A similar workflow can be described for ping and scp applications. The following workflow describes the ping application for simplicity:

- Step 0 (data plane application starts)

- The end host at EHU launches the application (E2E data plane): ping.
- The end host sends an ARP request to obtain the destination MAC address.

- Step 1 (ARP request -> flooding)
  - The ARP request packet generates a packet-in message (OF) to the controller.
  - The learning switch learns the SRC MAC address (source end host).
  - The controller sends a packet-out to flood the ARP request.
  - The ARP request floods the data plane.

- Step 2 (ARP reply -> flowmod)
  - The ARP request packet reaches the target host at i2Cat.
  - The target end host generates the ARP response packet.
  - The ARP response packet generates a packet-in message (OF) to the controller.
  - The learning switch leans the SRC MAC address (destination end host).
  - The controller sends a flowmod to install an exact flow entry for ARP reply packet.
  - The ARP reply from the buffer is sent to the appropriate outport.

- Step 3 (ICMP echo request -> flowmod)
  - The ARP reply packet reaches the source host at EHU.
  - The end host at EHU sends the ICMP echo request to reach the end host at i2Cat.
  - The ICMP echo request generates a packet-in message (OF) to the controller.
  - The controller sends a flowmod to install an exact flow entry for ICMP (EHU->i2Cat).
  - The ICMP echo request from the buffer is sent to the appropriate outport.

- Step 4 (ICMP echo reply -> flowmod)
  - The ICMP echo request reaches the destination host at i2Cat.
  - The end host at i2Cat sends the ICMP echo reply to the end host at EHU.
  - The ICMP echo reply generates a packet-in message (OF) to the controller.
  - The controller sends a flowmod to install an exact flow entry for ICMP (i2Cat->EHU).
  - The ICMP echo reply from the buffer is sent to the appropriate outport.

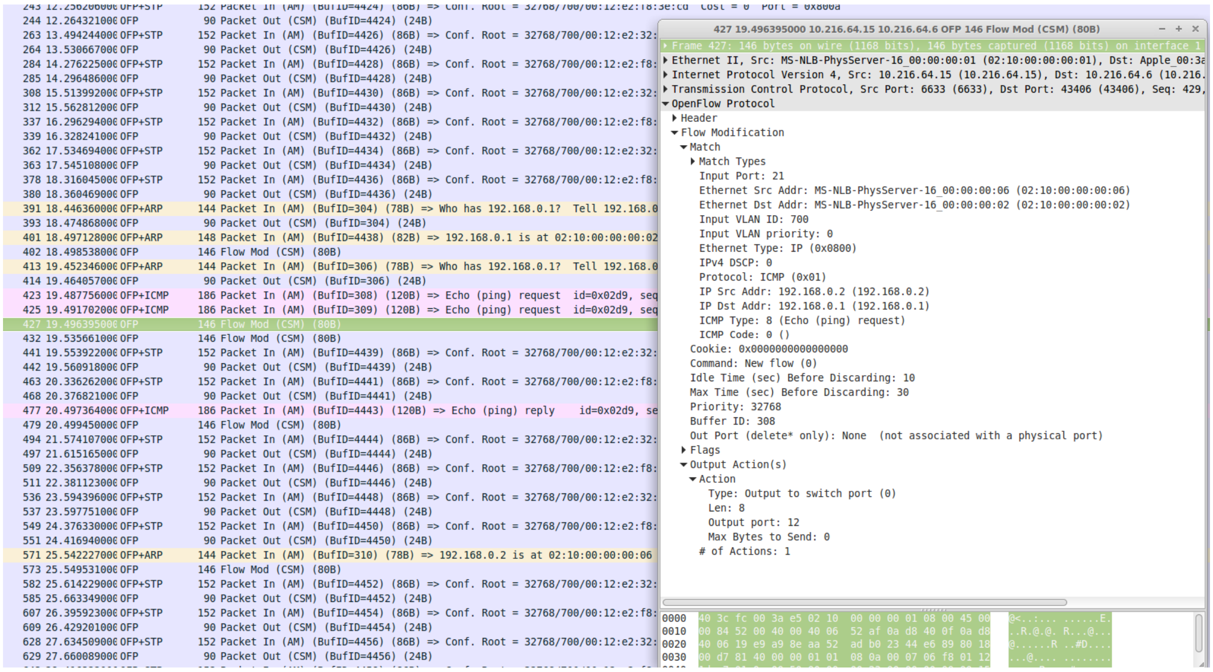
- Step 5 (ping at terminal)
  - The ICMP echo reply reaches the source host at EHU.
  - The ping data is displayed in the terminal.

3) Testing results

Control plane capture (OpenFlow messages)

The following capture shown in Figure 2-5 (summarized in Figure 2-6) was obtained by Wireshark. The capture was done at the DOCSIS Proxy, which had one interface connected to the controller and another interface connected to the DOCSIS platform where the Residential Gateway Helper (RGH) and the Aggregation Helper (AGH) were located. The DOCSIS Proxy spoke OFv1.0 to the controller and OFv1.2 to the helpers. Due to limitations in the OpenFlow dissector of Wireshark, it is not possible to show two different versions of OpenFlow at the same time. As a consequence, the summary from this capture has been edited to show the interaction of the DOCSIS Proxy with the controller and the helpers in a common view (Figure 2-6). By doing so, the steps described previously in the workflow can be easily identified in the Proxy to Controller interaction.

This capture shows the OpenFlow messages generated as a result of a ping application at data plane with a learning switch at control plane, but it is similar (i.e. the OpenFlow messages exchanged) to the capture that can be obtained with other data plane applications such as scp or ssh.



```
243 12.25626000€ OFP+STP     152 Packet In (AM) (BufID=4424) (86B) => Conf. Root = 32768/700/00:12:e2:f8:3e:cd  Cost = 0  Port = 0x800a
244 12.26432100€ OFP          90 Packet Out (CSM) (BufID=4424) (24B)
263 13.49424400€ OFP+STP     152 Packet In (AM) (BufID=4426) (86B) => Conf. Root = 32768/700/00:12:e2:32:
264 13.53066700€ OFP          90 Packet Out (CSM) (BufID=4426) (24B)
284 14.27622500€ OFP+STP     152 Packet In (AM) (BufID=4428) (86B) => Conf. Root = 32768/700/00:12:e2:f8:
285 14.29648600€ OFP          90 Packet Out (CSM) (BufID=4428) (24B)
308 15.51399200€ OFP+STP     152 Packet In (AM) (BufID=4430) (86B) => Conf. Root = 32768/700/00:12:e2:32:
312 15.56281200€ OFP          90 Packet Out (CSM) (BufID=4430) (24B)
337 16.29629400€ OFP+STP     152 Packet In (AM) (BufID=4432) (86B) => Conf. Root = 32768/700/00:12:e2:f8:
339 16.32824100€ OFP          90 Packet Out (CSM) (BufID=4432) (24B)
362 17.53469400€ OFP+STP     152 Packet In (AM) (BufID=4434) (86B) => Conf. Root = 32768/700/00:12:e2:32:
363 17.54510800€ OFP          90 Packet Out (CSM) (BufID=4434) (24B)
378 18.31604500€ OFP+STP     152 Packet In (AM) (BufID=4436) (86B) => Conf. Root = 32768/700/00:12:e2:f8:
380 18.36046900€ OFP          90 Packet Out (CSM) (BufID=4436) (24B)
391 18.44636000€ OFP+ARP     144 Packet In (AM) (BufID=304) (78B) => Who has 192.168.0.1?  Tell 192.168.0
393 18.47486800€ OFP          90 Packet Out (CSM) (BufID=304) (24B)
401 18.49712800€ OFP+ARP     148 Packet In (AM) (BufID=4438) (82B) => 192.168.0.1 is at 02:10:00:00:00:02
402 18.49853800€ OFP         146 Flow Mod (CSM) (80B)
413 19.45234600€ OFP+ARP     144 Packet In (AM) (BufID=306) (78B) => Who has 192.168.0.1?  Tell 192.168.0
414 19.46405700€ OFP          90 Packet Out (CSM) (BufID=306) (24B)
423 19.48775600€ OFP+ICMP    186 Packet In (AM) (BufID=308) (120B) => Echo (ping) request  id=0x02d9, seq
425 19.49170200€ OFP+ICMP    186 Packet In (AM) (BufID=309) (120B) => Echo (ping) request  id=0x02d9, seq
427 19.49639500€ OFP         146 Flow Mod (CSM) (80B)
432 19.53566100€ OFP         146 Flow Mod (CSM) (80B)
441 19.55392200€ OFP+STP     152 Packet In (AM) (BufID=4439) (86B) => Conf. Root = 32768/700/00:12:e2:32:
442 19.56091800€ OFP          90 Packet Out (CSM) (BufID=4439) (24B)
463 20.33626200€ OFP+STP     152 Packet In (AM) (BufID=4441) (86B) => Conf. Root = 32768/700/00:12:e2:f8:
468 20.37682100€ OFP          90 Packet Out (CSM) (BufID=4441) (24B)
477 20.49736400€ OFP+ICMP    186 Packet In (AM) (BufID=4443) (120B) => Echo (ping) reply    id=0x02d9, se
479 20.49945000€ OFP         146 Flow Mod (CSM) (80B)
494 21.57410700€ OFP+STP     152 Packet In (AM) (BufID=4444) (86B) => Conf. Root = 32768/700/00:12:e2:32:
497 21.61516500€ OFP          90 Packet Out (CSM) (BufID=4444) (24B)
509 22.35637800€ OFP+STP     152 Packet In (AM) (BufID=4446) (86B) => Conf. Root = 32768/700/00:12:e2:f8:
511 22.38112300€ OFP          90 Packet Out (CSM) (BufID=4446) (24B)
536 23.59439600€ OFP+STP     152 Packet In (AM) (BufID=4448) (86B) => Conf. Root = 32768/700/00:12:e2:32:
537 23.59775100€ OFP          90 Packet Out (CSM) (BufID=4448) (24B)
549 24.37633000€ OFP+STP     152 Packet In (AM) (BufID=4450) (86B) => Conf. Root = 32768/700/00:12:e2:f8:
551 24.41694000€ OFP          90 Packet Out (CSM) (BufID=4450) (24B)
571 25.54222700€ OFP+ARP     144 Packet In (AM) (BufID=310) (78B) => 192.168.0.2 is at 02:10:00:00:00:06
573 25.54953100€ OFP         146 Flow Mod (CSM) (80B)
582 25.61422900€ OFP+STP     152 Packet In (AM) (BufID=4452) (86B) => Conf. Root = 32768/700/00:12:e2:32:
585 25.66334900€ OFP          90 Packet Out (CSM) (BufID=4452) (24B)
607 26.39592300€ OFP+STP     152 Packet In (AM) (BufID=4454) (86B) => Conf. Root = 32768/700/00:12:e2:f8:
609 26.42920100€ OFP          90 Packet Out (CSM) (BufID=4454) (24B)
628 27.63450900€ OFP+STP     152 Packet In (AM) (BufID=4456) (86B) => Conf. Root = 32768/700/00:12:e2:32:
629 27.66008900€ OFP          90 Packet Out (CSM) (BufID=4456) (24B)
```

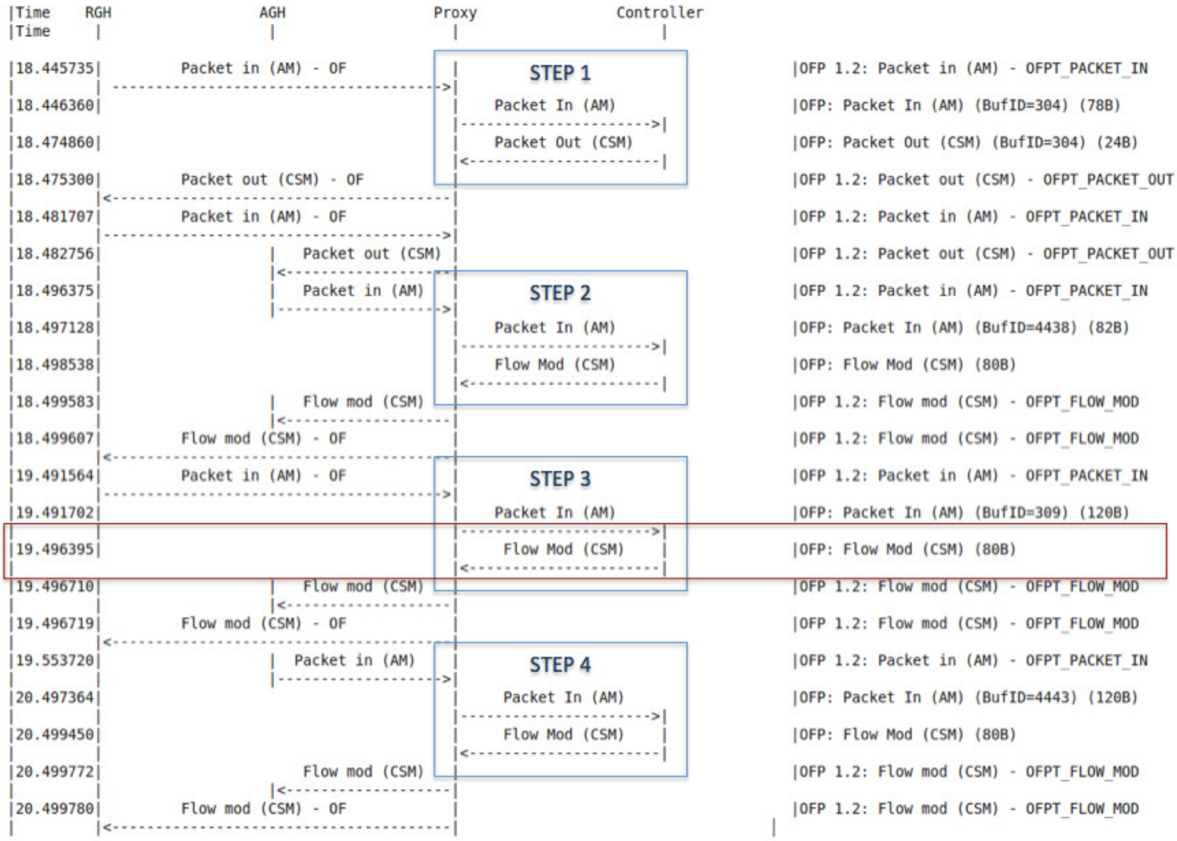
427 19.496395000 10.216.64.15 10.216.64.6 OFP 146 Flow Mod (CSM) (80B)

```
▸ Frame 427: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface 1
▸ Ethernet II, Src: MS-NLB-PhysServer-16_00:00:00:01 (02:10:00:00:00:01), Dst: Apple_00:3a
▸ Internet Protocol Version 4, Src: 10.216.64.15 (10.216.64.15), Dst: 10.216.64.6 (10.216.
▸ Transmission Control Protocol, Src Port: 6633 (6633), Dst Port: 43406 (43406), Seq: 429,
▾ OpenFlow Protocol
  ▸ Header
  ▾ Flow Modification
    ▾ Match
      ▸ Match Types
        Input Port: 21
        Ethernet Src Addr: MS-NLB-PhysServer-16_00:00:00:06 (02:10:00:00:00:06)
        Ethernet Dst Addr: MS-NLB-PhysServer-16_00:00:00:02 (02:10:00:00:00:02)
        Input VLAN ID: 700
        Input VLAN priority: 0
        Ethernet Type: IP (0x0800)
        IPv4 DSCP: 0
        Protocol: ICMP (0x01)
        IP Src Addr: 192.168.0.2 (192.168.0.2)
        IP Dst Addr: 192.168.0.1 (192.168.0.1)
        ICMP Type: 8 (Echo (ping) request)
        ICMP Code: 0 ()
      Cookie: 0x0000000000000000
      Command: New flow (0)
      Idle Time (sec) Before Discarding: 10
      Max Time (sec) Before Discarding: 30
      Priority: 32768
      Buffer ID: 308
      Out Port (delete* only): None  (not associated with a physical port)
    ▸ Flags
    ▾ Output Action(s)
      ▾ Action
          Type: Output to switch port (0)
          Len: 8
          Output port: 12
          Max Bytes to Send: 0
        # of Actions: 1
```

```
0000  40 3c fc 00 3a e5 02 10  00 00 00 01 08 00 45 00   @<..:... ......E.
0010  00 84 52 00 40 00 40 06  52 af 0a d8 40 0f 0a d8   ..R.@.@. R...@...
0020  40 06 19 e9 a9 8e aa 52  ad b0 23 44 e6 89 80 18   @......R ..#D....
0030  00 d7 81 40 00 00 01 01  08 0a 00 07 06 f8 01 12   ...@.... ........
```

Figure 2-5 Wireshark capture of OpenFlow v1.0 messages exchanged with a ping application at data plane and learning switch application (POX) at control plane

The following figure summarizes the complete exchange of OpenFlow messages (shown in previous capture in Figure 2-5). As an example, the flowmod highlighted in that previous Wireshark capture corresponds to the flowmod at 19.496395 seconds from the POX Controller to the DOCSIS Proxy.

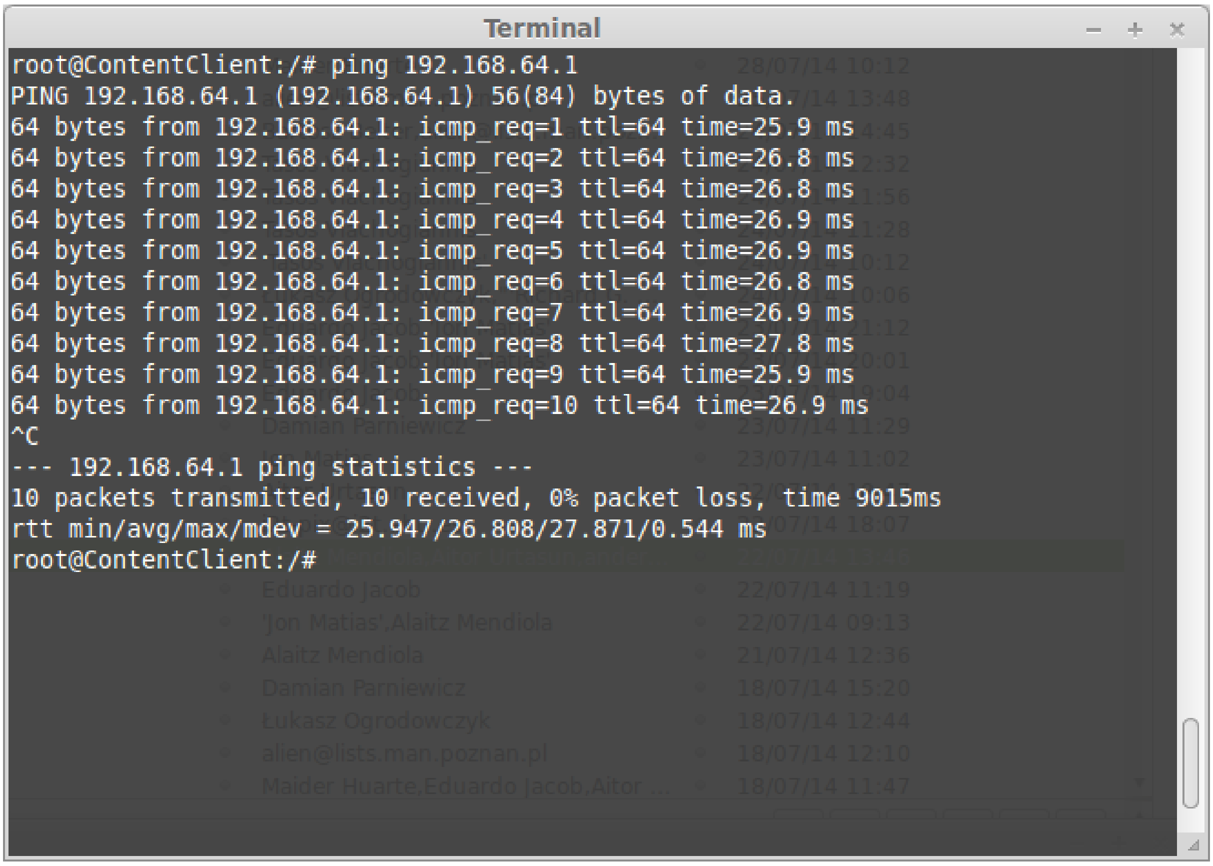


```
|Time     RGH               AGH                Proxy              Controller
|Time      |                 |                   |                    |
|18.445735|     Packet in (AM) - OF              |                           |OFP 1.2: Packet in (AM) - OFPT_PACKET_IN
|         | ---------------------------------->|
|18.446360|                                     |    Packet In (AM)         |OFP: Packet In (AM) (BufID=304) (78B)
|                                               |--------------------->|
|18.474860|                                     |    Packet Out (CSM)       |OFP: Packet Out (CSM) (BufID=304) (24B)
|                                               |<---------------------|
|18.475300|     Packet out (CSM) - OF           |                           |OFP 1.2: Packet out (CSM) - OFPT_PACKET_OUT
|         |<-----------------------------------|
|18.481707|     Packet in (AM) - OF             |                           |OFP 1.2: Packet in (AM) - OFPT_PACKET_IN
|         |----------------------------------->|
|18.482756|                 |  Packet out (CSM) |                           |OFP 1.2: Packet out (CSM) - OFPT_PACKET_OUT
|         |                 |<-----------------|
|18.496375|                 |  Packet in (AM)   |                           |OFP 1.2: Packet in (AM) - OFPT_PACKET_IN
|         |                 |----------------->|
|18.497128|                                     |    Packet In (AM)         |OFP: Packet In (AM) (BufID=4438) (82B)
|         |                                     |--------------------->|
|18.498538|                                     |    Flow Mod (CSM)         |OFP: Flow Mod (CSM) (80B)
|         |                                     |<---------------------|
|18.499583|                 |  Flow mod (CSM)   |                           |OFP 1.2: Flow mod (CSM) - OFPT_FLOW_MOD
|         |                 |<-----------------|
|18.499607|     Flow mod (CSM) - OF             |                           |OFP 1.2: Flow mod (CSM) - OFPT_FLOW_MOD
|         |<-----------------------------------|
|19.491564|     Packet in (AM) - OF             |                           |OFP 1.2: Packet in (AM) - OFPT_PACKET_IN
|         |----------------------------------->|
|19.491702|                                     |    Packet In (AM)         |OFP: Packet In (AM) (BufID=309) (120B)
|         |                                     |--------------------->|
|19.496395|                                     |    Flow Mod (CSM)    |    |OFP: Flow Mod (CSM) (80B)
|                                               |<---------------------|
|19.496710|                 |  Flow mod (CSM)   |                           |OFP 1.2: Flow mod (CSM) - OFPT_FLOW_MOD
|         |                 |<-----------------|
|19.496719|     Flow mod (CSM) - OF             |                           |OFP 1.2: Flow mod (CSM) - OFPT_FLOW_MOD
|         |<-----------------------------------|
|19.553720|                 | Packet in (AM)    |                           |OFP 1.2: Packet in (AM) - OFPT_PACKET_IN
|         |                 |----------------->|
|20.497364|                                     |    Packet In (AM)         |OFP: Packet In (AM) (BufID=4443) (120B)
|         |                                     |--------------------->|
|20.499450|                                     |    Flow Mod (CSM)    |    |OFP: Flow Mod (CSM) (80B)
|         |                                     |<---------------------|
|20.499772|                    Flow mod (CSM)   |                           |OFP 1.2: Flow mod (CSM) - OFPT_FLOW_MOD
|         |                 |<-----------------|
|20.499780|     Flow mod (CSM) - OF             |                           |OFP 1.2: Flow mod (CSM) - OFPT_FLOW_MOD
|         |<-----------------------------------|                     |
```

Figure 2-6 Summary of the OpenFlow messages exchanged captured by Wireshark with a ping application at data plane and learning switch application (POX) at control plane

Data plane capture (ping output)

The following screenshot shows the output from data plane application, i.e. ping, used for the functional testing of DOCSIS ALIEN integration in OFELIA. Moreover, it can be used to measure round-trip latency at data plane between EHU Island and i2Cat Island, which is 26.808 ms on average.



```
root@ContentClient:/# ping 192.168.64.1
PING 192.168.64.1 (192.168.64.1) 56(84) bytes of data.
64 bytes from 192.168.64.1: icmp_req=1 ttl=64 time=25.9 ms
64 bytes from 192.168.64.1: icmp_req=2 ttl=64 time=26.8 ms
64 bytes from 192.168.64.1: icmp_req=3 ttl=64 time=26.8 ms
64 bytes from 192.168.64.1: icmp_req=4 ttl=64 time=26.9 ms
64 bytes from 192.168.64.1: icmp_req=5 ttl=64 time=26.9 ms
64 bytes from 192.168.64.1: icmp_req=6 ttl=64 time=26.8 ms
64 bytes from 192.168.64.1: icmp_req=7 ttl=64 time=26.9 ms
64 bytes from 192.168.64.1: icmp_req=8 ttl=64 time=27.8 ms
64 bytes from 192.168.64.1: icmp_req=9 ttl=64 time=25.9 ms
64 bytes from 192.168.64.1: icmp_req=10 ttl=64 time=26.9 ms
^C
--- 192.168.64.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 25.947/26.808/27.871/0.544 ms
root@ContentClient:/#
```

Figure 2-7 Output from ping application at data plane with learning switch application (POX) at control plane

2.1.3 Conclusions

During validation of DOCSIS ALIEN platform and corresponding developments the following OpenFlow functionalities were tested and successfully demonstrated:

- OpenFlow 1.0
- Flow entry match to:
    - In_Port
    - Ethernet type (recognize IPv4 and ARP)
    - Source IPv4 address
    - Destination IPv4 address
    - IP protocol (TCP)
    - Source TCP port
    - Destination TCP port

- Supported actions:
    - Forward to a port
    - Drop

- Flow entry add
- Flow entry remove

- Packet-in
- Packet-out

In this DOCSIS case, valuable lessons were learned from integration in real scenarios, listed below:

- Packet-out was not properly working, and it was discovered thanks to the tests.
- ARP match internal IPs (1.0 and 1.2 different behaviour)
- Flooding mechanism implemented
- xDPd helpers (RGH and AGH) updated to latest versions due to VLAN matching 0.4.3
- OFELIA (also in CONET) interisland connectivity: allocate a free VLAN
    - Flowspaces manually approved.
    - Every change for the resources in charge in one island implies re-approval of the whole setup. Changes in one (controller IP, flowspace, vlan) affect all.

2.2 Integration of EZappliance

Integration of EZappliance into OFELIA environment was done with usage of complete new PSNC OFELIA Island where EZappliance device was deployed and interconnected to old OFELIA UNIVBRIS Island through GEANT layer2 link. HAL software implemented specifically for EZappliance (i.e.: xDPd based OpenFlow agent with specific EZchip NP-3 driver) were deployed on top of EZappliance platform enabling OpenFlow control for purpose of OFELIA experiments. The successful integration of EZappliance and full functional validation of its HAL prototype were possible thanks to control and data plane tests using simple ping tool and also more complex media streaming scenario. For the testing experiment purposes, an OpenFlow controller logic had to be implemented by EZappliance integration team in PSNC.

2.2.1 Description of the integration

The integration description contains information about hardware used in each OFELIA island as well as data plane topologies and control plane connections. Additionally, OpenFlow configuration is presented with specific attention to EZappliance device configuration.

1) PSNC Island

PSNC OFELIA Island is established using four devices (2 server machines and 2 network devices):

- **HP Proliant DL360 G7** - used for deploying all management and control software required by OFELIA environment (OCF components, OpenVPN and QUAGGA)
- **IBM System x3550 M3** - server with XEN installed, controlled by OCF Virtualization Aggregation Manager and participating in data plane as endpoint
- **EZappliance** - ALIEN platform with HAL installed (xDPd-for-EZappliance) exposing OpenFlow protocol, dynamically configuring data plane connections
- **Juniper EX4200** - manually configured switch, providing static data plane connectivity through GEANT and PIONIER networks to other islands

Data plane connectivity and OFELIA management software deployment is presented in Figure 2-8. In regards of data plane, PSNC Island is connected with Bristol Island and PUT Island. Both islands are connected with two VLANs: 601 and

602. PSNC Island is participating in OFELIA control network thanks to Open VPN connection to main control network hub located in iMinds Island and QUAGGA routing daemon dynamically configuring routing entries towards other OFELIA Islands. All OCF software components (Expedient GUI, Aggregate Managers, FlowVisor) as well as hardware managed by OCF (server with XEN and EZappliance platform) have assigned IP addresses from OFELIA control network pool officially assigned to PSNC Island. OCF Virtualization Aggregate Manager is managing an IBM server where Virtual Machines are deployed as part of OFELIA slices. EZappliance platform is virtualized by OCF OpenFlow Aggregate Manager and OpenFlow FlowVisor.

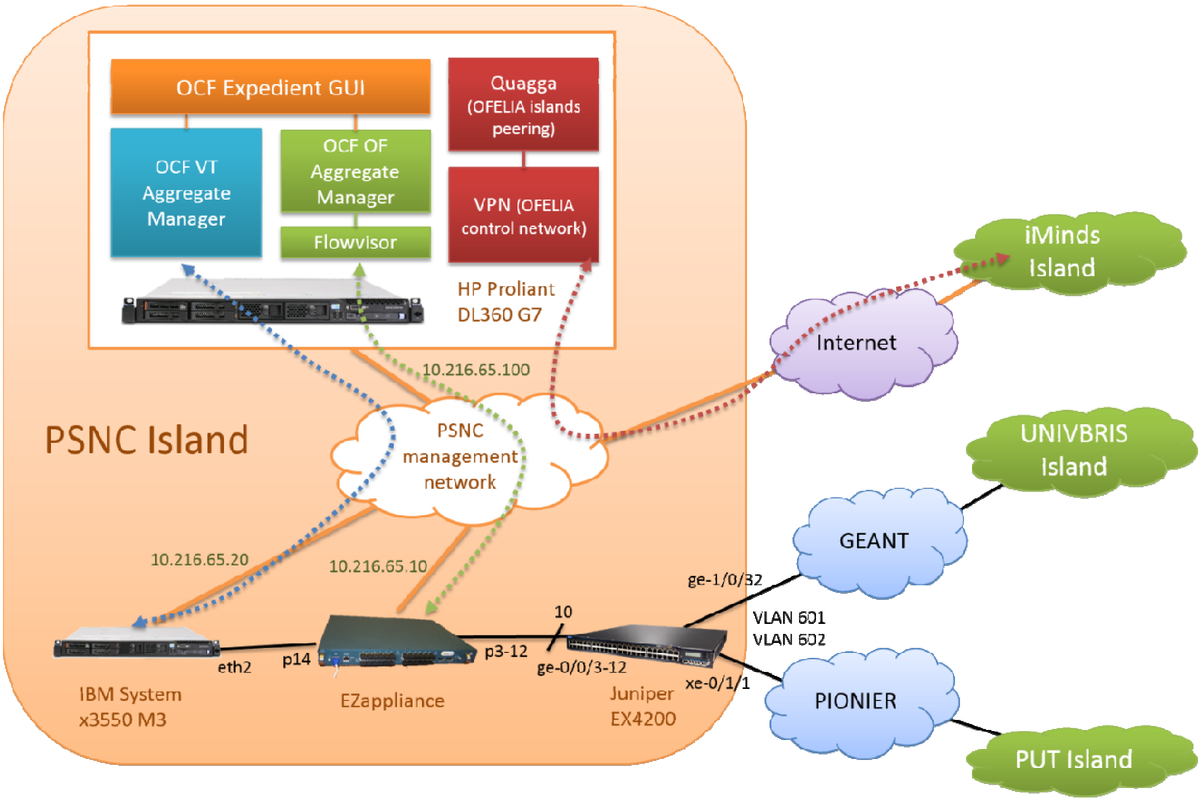


Figure 2-8 PSNC Island overview

Control configuration

The configuration of OSPF QUAGGA daemon exposes control network of PSNC Island as OSPF area 17 with network addressing 10.216.65.0/24 to other OFELIA islands:

```
 ! -*- ospf quagga configuration-*-
 !
hostname Ofelia_PSNC_ospfd
 !
interface tap1
ip ospf authentication message-digest
ip ospf message-digest-key 1 md5 xxxxxx
ip ospf cost 5
interface tap2
```

```
ip ospf authentication message-digest
ip ospf message-digest-key 1 md5 xxxxxx
ip ospf cost 15
 !
router ospf
router-id 10.216.0.17
log-adjacency-changes detail
network 10.216.0.0/23 area 0
network 10.216.2.0/23 area 0
network 10.216.65.0/24 area 17
passive-interface eth0
 !
area 0 range 10.216.0.0/23
area 17 range 10.216.65.0/24
area 0 authentication message-digest
area 17 authentication message-digest
 !
log file /var/log/quagga/ospfd.log
```

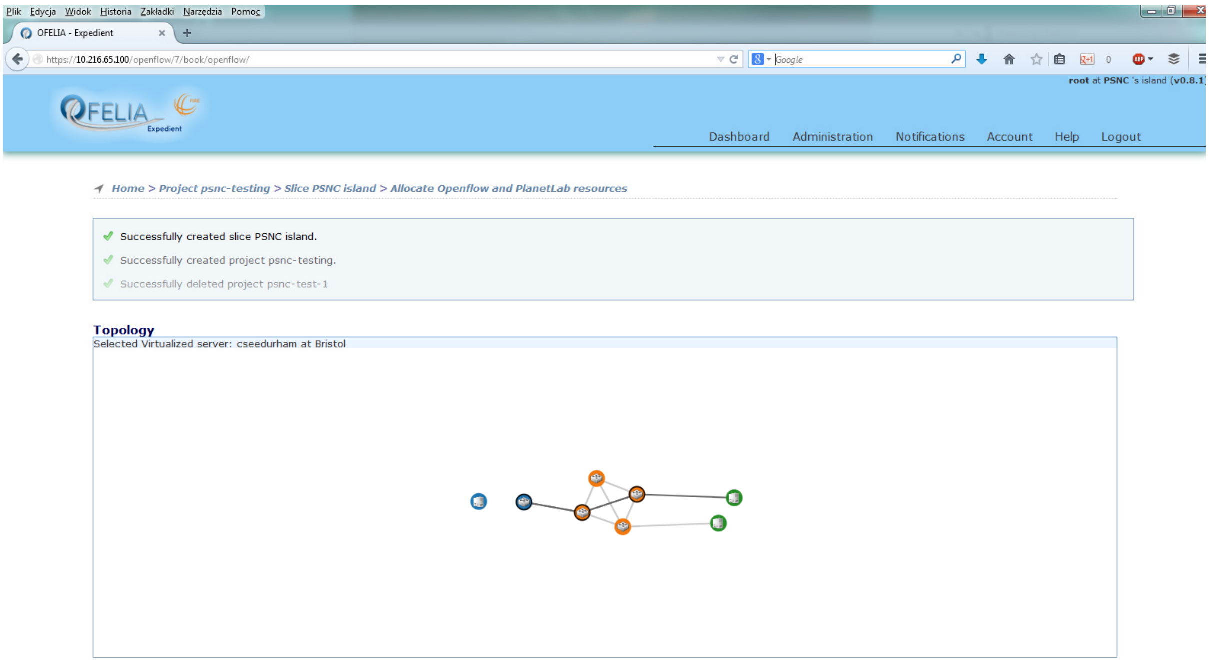
In PSNC Island, the following control network addresses were statically assigned:

```
 10.216.65.10        EZappliance HSP (xDPd)
 10.216.65.20        PSNC-Server-Xen-1 with OXA
 10.216.65.100       OCF (Expedient, AMs, Flowvisor)
 10.216.65.101-200   IP range for VMs created with OCF
 10.216.65.254       Management network router
```

And PSNC's OpenFlow and Virtualization Aggregate Managers are available for any Expedient in OFELIA test bed by these URIs:

```
OFAM: https://10.216.65.100:8443/xmlrpc/xmlrpc/
VTAM: https://10.216.65.100:8445/xmlrpc/plugin/
```

Figure 2-9 presents a screenshot from PSNC OCF Expedient showing a flowspace definition (which device ports and which VLAN tag will be used) for a slice created for testing EZappliance integration. On the topology picture one data plane link between PSNC server and EZappliance is not shown because EZappliance is not supporting LLDP protocol required for OFELIA automatic network topology discovery.


Plik Edycja Widok Historia Zakładki Narzędzia Pomoc
OFELIA - Expedient
https://10.216.65.100/openflow/7/book/openflow/
Google
root at PSNC 's island (v0.8.1
OFELIA Expedient
Dashboard
Administration
Notifications
Account
Help
Logout

Home > Project psnc-testing > Slice PSNC island > Allocate Openflow and PlanetLab resources

✓ Successfully created slice PSNC island.
✓ Successfully created project psnc-testing.
✓ Successfully deleted project psnc-test-1

Topology
Selected Virtualized server: cseedurham at Bristol

Figure 2-9 Screenshot from PSNC OCF Expedient showing the slice topology for EZappliance integration

Diagram shown in Figure 2-10 presents the same slice as in Figure 2-9 but provides more details regarding OpenFlow identifiers (DPID numbers, port identifiers). Two OFELIA default VMs were requested in PSNC Island (for testing client and POX controller) and one OFELIA default VM set in Bristol Island (for Media Server). Traffic between Client and Media Server VM was tagged with VLAN tag 700. The network path between servers is configured by POX OpenFlow controller.

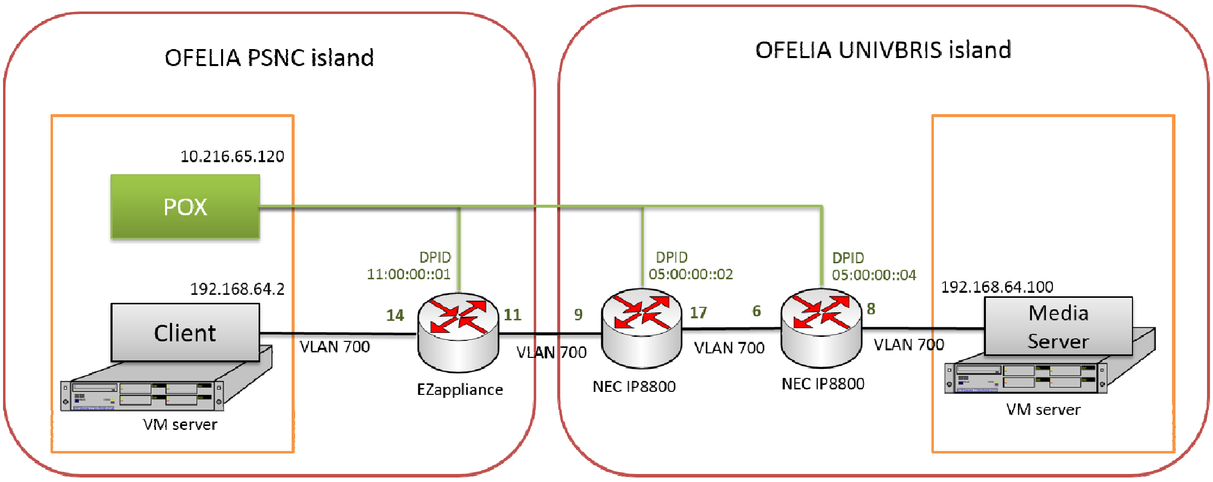


Figure 2-10 EZappliance integration slice and flowspace details

2) EZappliance integration in OFELIA

HAL prototype (xDPd with EZappliance device driver) configuration for EZappliance device contains OpenFlow datapath id of the switch (id is visible in OCF Expedient GUI), OpenFlow version 1.0, IP address of FlowVisor as an OF proxy controller of the switch, list of EZappliance data path ports and local address of EZ-Proxy component of EZappliance HSP:

```
config:{
openflow:{
logical-switches:{
dp0:{
dpid = "0x1100000000000001";
version = 1.0;
description="This is an PSNC-EZappliance switch";
controller-connections:{
main:{
remote-hostname="10.216.65.100";
remote-port=6633;
            };
   };
reconnect-time=1; #seconds
num-of-tables=1;
          ports = ("eth0","eth1","eth2","eth3","eth4","eth5","eth6","eth7",
"eth8","eth9","eth10","eth11","eth12","eth13","eth14")
        };
      };
    };
system:{
driver-extra-params="10.134.0.4"; #EZ Proxy IP address
    };
  };
```

FlowVisor installed as part of OCF software suite is reporting OpenFlow resources in PSNC Island composed of one OF switch (based on EZappliance platform) with 15 data plane switch:

```
root@alienOCFtemplate:~#  fvctl list-datapaths
 Password:
 Connected switches:
1 : 11:00:00:00:00:00:00:01
 root@alienOCFtemplate:~# fvctl list-datapath-info 11:00:00:00:00:00:00:01
 Password:
 {
   "connection": "/10.216.65.100:6633-->/10.216.65.10:39899",
   "current-flowmod-usage": {
     "fvadmin": 0
   },
   "dpid": "11:00:00:00:00:00:00:01",
   "num-ports": 24,
   "port-list": [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
   "port-names": ["eth0","eth1","eth2","eth3","eth4","eth5","eth6","eth7",
"eth8","eth9","eth10","eth11","eth12","eth13","eth14"]
}
```

2.2.2 Functional testing

The functional tests of HAL for EZappliance were done using streaming on demand application running in OpenFlow network. In order to run functional tests of HAL, an OFELIA slice was created which covered PSNC and UNIVBRIS OFELIA islands. The ALIEN slice included server resources (e.g. Virtual Machines for Client, Media Server, and OpenFlow controller), EZappliance ALIEN hardware platform and NEC switches.

1) OFELIA Resources involved in the testing

Functional testing performed with usage of EZappliance devices is demonstrating how HAL can be used to create a flexible streaming on demand service chain in OF networks with HAL-enabled network devices. Figure 2-11, illustrates the validation of HAL. Although these devices are programmable, they do not natively support the OpenFlow protocol. Through the addition of HAL they are able to communicate with the OpenFlow controller, and appear as any protocol-conforming switch. As a practical example, after the request for a video by an end user, the controller generates OpenFlow flow modifications (flowmods) to redirect web and video streaming requests from the clients to the media server. Finally, the video stream traffic from the media server is directed in the OpenFlow reactive manner, back to the client over the HAL-enabled network.

The following resources in each island were involved during tests:

- PSNC Island
  - HP Proliant DL360 G7 - with OpenFlow controller and Media Client VMs
  - IBM System x3550 M3 – with OCF
  - EZappliance – ALIEN network hardware platform
  - Juniper EX4200 – switch with access to GEANT network (towards UNIVBRIS)

- UNIVBRIS Island
  - 2x NEC IP8800 – OpenFlow switches
  - DELL poweredge 1950 server –with media server VM
  - Extreme Networks X450 – switch with access to GEANT network (towards PSNC)

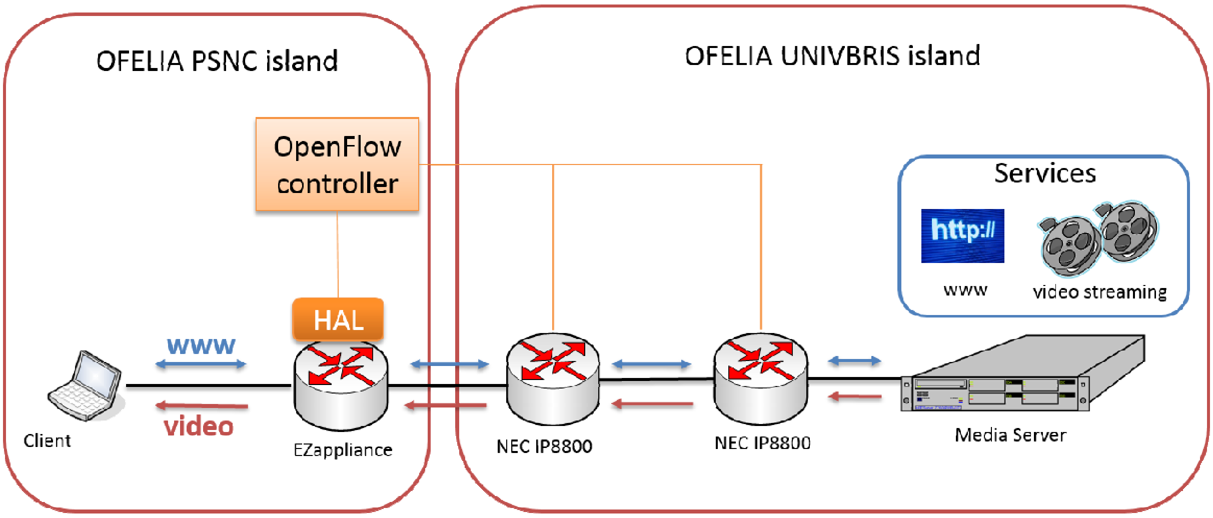


Figure 2-11 EZappliance functional testing overview

2) Testing workflow

The functional testing can be represented as a workflow with clearly defined steps which must be accomplished by HAL-enable EZappliance platforms:

- Step 0 (proactive web-access)
    - OF controller installs flow entries for web-access transport service (allows any client connected to EZappliance nodes to access web server)
    - Flow entries are statically defined in OpenFlow controller application code (predefined data plane topology knowledge)
    - Web-access transport service is bidirectional and must support ARP, ICMP and TCP traffic

- Step 1 (interacting with web server)
    - User can view and navigate on web server page(s)

- Step 2 (user requests a movie stream)
    - Step 2.0) User clicks on “Play” button within a video page
    - Step 2.1) Video player generates RTSP request for video streaming server (destination TCP port 554)
    - Step 2.2) RTSP request is forwarded to OF controller in packet-in event
    - Step 2.3) OF controller ignores RTSP request; streaming client cannot connect

- Step 3 (user requests a network configuration)
    - Step 3.0) User clicks on “Configure network” button and HTML page requests TCP session with 10.0.0.200 (not existing IP address which is used within a “signalling” to OF controller, that OF controller must take an action)
    - Step 3.1) Client PC sends ARP request for 10.0.0.200
    - Step 3.2) ARP request is forwarded to OF controller by packet-in event
    - Step 3.3) OF controller installs flow entries enabling RTSP sessions between client and video streaming server

- Step 4 (user requests a movie stream)
    - Step 4.0) Users clicks on “Play” button
    - Step 4.1) Video player generates RTSP request for video streaming server (destination TCP port 554)
    - Step 4.2) RTSP request is forwarded by network devices to video streaming server
    - Step 4.3) video streaming server sends RTSP response to client
    - Step 4.4) RTSP response is forwarded to client and RTSP session is established

- Step 5 (video stream is sent)
    - Step 5.0) Basing on RTSP session, video streaming server starts sending RTP messages carrying video content
    - Step 5.1) First network node generates packet-in with that RTP packet
    - Step 5.2) OF controller recognizes destination IPv4 and destination UDP port, and sends proper flow entries in a reactive mode to network devices
    - Step 5.3) video streaming packets are sent through network to client and video is displayed

- Step 6 (user stops video)
    - Step 6.0) User clicks on “Stop” button
    - Step 6.1) Video player generates RTSP request for video streaming server (destination TCP port 554)
    - Step 6.2) RTSP request is forwarded by network devices to video streaming server
    - Step 6.3) video streaming server sends RTSP response to client
    - Step 6.4) RTSP response is forwarded to client and RTSP session is ended

- Step 7 (user clears network)
  - Step 7.0) User clicks on “Deconfigure network” button and HTML page requests TCP session with 10.0.0.201 (not existing address)
  - Step 7.1) Client PC sends ARP request for 10.0.0.201 (not existing IP address which is used within a “signalling” to OF controller that OF controller must take an action)
  - Step 7.2) ARP request is forwarded to OF controller by packet-in event
  - Step 7.3) OF controller uninstall flow entries for RTSP and RTP packets which sent in the data plane between client and video streaming server

OpenFlow entries installed proactively in EZappliance for Step 0

Flow entries for traffic from client to server:

- match to:
  - ARP ethertype, source IPv4 of client and destination IPv4 of web server within inspected ARP packet
  - ICMP IP protocol, source IPv4 of client and destination IPv4 of web server
  - source IPv4 of client, destination IPv4 of web server and destination TCP port 80

- action is output to EZappliance interface towards a server (as shown on Figure 2-12)

Flow entries for traffic from server to client:

- match to:
  - ARP ethertype and source IPv4 of web server and destination IPv4 of client within inspected ARP packet
  - ICMP IP protocol, source IPv4 of web server and destination IPv4 of client
  - source IPv4 of web server, destination IPv4 of client and source TCP port 80

- Action is output to EZappliance interface towards client (as shown on Figure 2-12)

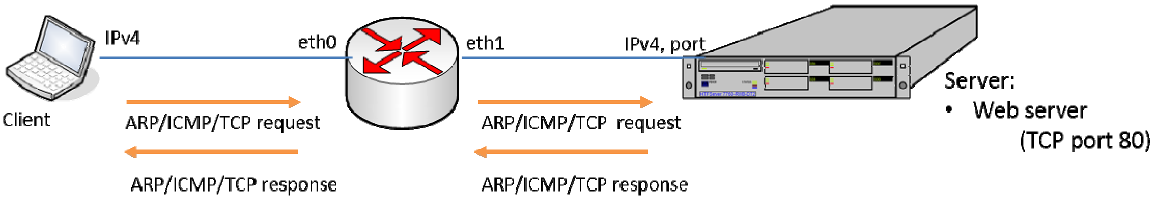


Figure 2-12 Flow entries – step 0 (proactive web-access)

OpenFlow entries installed proactively in EZappliance for Step 3

Flow entries for RTSP traffic from client to server:

- match to:
  - source IPv4 of client, destination IPv4 of web server and destination TCP port 554

- action is output to EZappliance interface towards server (as shown on Figure 2-13)

Flow entries for RTSP traffic from server to client:

- match to:

- source IPv4 of web server, destination IPv4 of client and source TCP port 554

- action is output to EZappliance interface towards client (as shown on Figure 2-13)

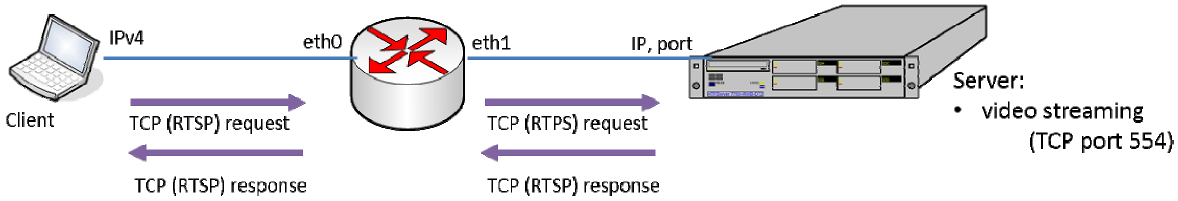


Figure 2-13 Flow entries – step 3 (user requests a network configuration)

<u>OpenFlow entries installed reactively in EZappliance in Step 5</u>

Flow entries for RTP traffic from server to client:

- match to:
    - source IPv4 of web server, destination IPv4 of client and destination UDP port (destination UDP port is recognized by OF controller during packet-in of the first non-matched UDP packet containing RTP payload sent from server)

- action is output to EZappliance interface towards client (as shown on Figure 2-14)

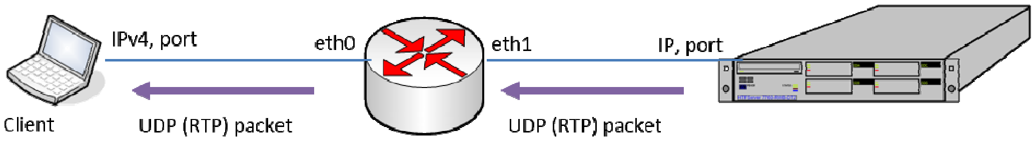


Figure 2-14 Flow entries – step 5 (video stream is sent)

3) Testing results

EZappliance integration validation started with launching POX controller with ‘pox-alien-ez-integration’ module performing testing workflow. The listing bellow presents the output of POX controller (“in red” is shown OF switch DPID connected and in “in blue” it is shown what flowmods are installed in an OF switch):

```
damianparniewicz@ALIENController:~/pox$  ./pox.py  openflow.of_01  --port=6653
   pox-alien-ez-integration
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
  WARNING:version:POX requires Python 2.7. You're running 2.6.
  WARNING:version:If you run into problems, try using Python 2.7 or PyPy.
  INFO:alien-ez-pox:ALIEN POX EZappliance integration validation started!
  INFO:core:POX 0.2.0 (carp) is up.
  INFO:openflow.of_01:[00-00-00-00-00-02|1280 1] connected
  INFO:alien-ez-pox:Switch 00-00-00-00-00-02|1280 has come up.
  INFO:alien-ez-pox:Clearing all flows from 00-00-00-00-00-02|1280.
```

```
INFO:alien-ez-pox:arp installed
  INFO:alien-ez-pox:icmp installed
  INFO:alien-ez-pox:web installed
  INFO:alien-ez-pox:rtsp installed
  INFO:alien-ez-pox:rtsp installed
INFO:openflow.of_01:[00-00-00-00-00-04|1280 2] connected
  INFO:alien-ez-pox:Switch 00-00-00-00-00-04|1280 has come up.
  INFO:alien-ez-pox:Clearing all flows from 00-00-00-00-00-04|1280.
INFO:alien-ez-pox:arp installed
  INFO:alien-ez-pox:icmp installed
  INFO:alien-ez-pox:web installed
  INFO:alien-ez-pox:rtsp installed
  INFO:alien-ez-pox:rtsp installed
INFO:openflow.of_01:[00-00-00-00-00-01|4352 3] connected
  INFO:alien-ez-pox:Switch 00-00-00-00-00-01|4352 has come up.
  INFO:alien-ez-pox:Clearing all flows from 00-00-00-00-00-01|4352.
INFO:alien-ez-pox:arp installed
  INFO:alien-ez-pox:icmp installed
  INFO:alien-ez-pox:web installed
  INFO:alien-ez-pox:rtsp installed
  INFO:alien-ez-pox:rtsp installed
```

The following log presents one of OpenFlow flowmods which was handled by EZappliance HSP. This concrete flowmod enable switching of ICMP messages coming from client to port 11 of EZappliance. The flowmod is translated into EZappliance memory entries and sent to EZ-Proxy for further processing. The log is coloured to show matching and action fields name (in blue) and values (in red) in handler OpenFlow message as well within a binary entry sent to EZchip memory structure.

```
  [rofl][ctl] ctid:0x0 Flow-Mod message received
<cofmsg version:1 type:14 length:80 xid:0x15 >
<cofmsg_flow_mod >
<command: -ADD- >
<table-id:0 >
<cookie:0x0>
<idle-timeout:0 >
<hard-timeout:0 >
<priority:0x1>
<buffer-id:0xffffffff >
<flags:0x0>
<out-port:0xffff >
<matches: >
<cofmatch ofp-version:1 >
<coxmatches #matches:5 >
<coxmatch oxm-id: 0x80000a02 >
<cmemory: data:0x202fb1e0 datalen:6 >
```

```
                0000: 80 00 0a 02 08 00
<eth-type: 0x800>
<coxmatch oxm-id: 0x80000c02 >
<cmemory: data:0x202fb1e0 datalen:6 >
                0000: 80 00 0c 02 02 bc
<coxmatch_ofb_vlan_vid >
<vlan-vid: 0xbc/0xffff >
<coxmatch oxm-id: 0xffff0108 >
<cmemory: data:0x21ba97a0 datalen:12 >
                  0000: ff ff 01 08 c0 a8 40 0b   ff ff ff ff
<nw-src: 192.168.64.11/255.255.255.255>
<coxmatch oxm-id: 0xffff0308 >
<cmemory: data:0x21ba97a0 datalen:12 >
                   0000: ff ff 03 08 c0 a8 40 64   ff ff ff ff
<nw-dst: 192.168.64.100/255.255.255.255><actions: >
<cofactions ofp-version:1 #actions:1 >
<cofaction ofp-version:1 type:0 length:8  >
<cofaction_output port: 0xb max-len: 0x0 >
  [AFA] driver_of1x_process_flow_mod_add (dpid: 1, table_id: 0, buffer_id: -1,
   check_overlap: 0, reset_counts: 0)
  [Pipeline-imp] plaftorm_of1x_add_entry_hook (new_entry: 0x202fbff0)
     EZ action: forward to port11
  [EZ-CORBA] Calling set_ez_struct method
    -> key is:  0:0x0  1:0x1  2:0x0  3:0x0  4:0x0  5:0x0  6:0x0  7:0x0  8:0x0  9:0x0
   10:0x0 11:0x0 12:0x0 13:0x0
14:0x0  15:0xbc  16:0x12  17:0x0  18:0x8  19:0xb  20:0x421:0xa8  22:0xc0  23:0x64
   24:0x40
25:0xa826:0xc0  27:0x0  28:0x0  29:0x0  30:0x0  31:0x0  32:0x00  33:0x00  34:0x00
   35:0x0
36:0x0 37:0x0
-> mask is: 0:0x0 1:0xff 2:0x0 3:0x0 4:0x0 5:0x0 6:0x0 7:0x0 8:0x0 9:0x0 10:0x0
   11:0x0 12:0x0
13:0x014:0x0  15:0xff  16:0x1f  17:0xff  18:0xff  19:0xff  20:0xff  21:0xff  22:0xff
   23:0xff 24:0xff
                    25:0xff  26:0xff  27:0x0  28:0x0  29:0x0  30:0x0  31:0x0  32:0x1
   33:0x0 34:0x0 35:0x0 36:0x0
37:0x0
    ->result is: 0:0x3 1:0x2 2:0xa 3:0x0 4:0x0 5:0x0 6:0x0 7:0x0
```

Finally, both client and media server were interconnected through OF switches in Bristol and EZappliance device in PSNC. The following listing presents ping from 192.168.64.2 (Client node IP located in PSNC) towards 192.168.64.100 (Media server node located in UNIVBRIS):

```
  damianparniewicz@alien-client:~$ ping 192.168.64.100
  PING 192.168.64.100 (192.168.64.100) 56(84) bytes of data.
```

```
64 bytes from 192.168.64.100: icmp_req=1 ttl=64 time=866 ms
64 bytes from 192.168.64.100: icmp_req=2 ttl=64 time=39.0 ms
64 bytes from 192.168.64.100: icmp_req=3 ttl=64 time=39.1 ms
64 bytes from 192.168.64.100: icmp_req=4 ttl=64 time=39.1 ms
64 bytes from 192.168.64.100: icmp_req=5 ttl=64 time=39.1 ms
64 bytes from 192.168.64.100: icmp_req=6 ttl=64 time=39.0 ms
64 bytes from 192.168.64.100: icmp_req=7 ttl=64 time=39.0 ms
```

2.2.3 Conclusions

Validation of EZappliance platform was performed in OFELIA test bed with resources from UNIVBRIS and PSNC Islands. The tests were performed in a slice created with usage of OCF tool. The slice was composed basing on resources from one EZappliance device controlled by HAL instance, two standard OpenFlow switches and two servers where VMs were deployed.

During validation of EZappliance platform and corresponding developments the following OpenFlow functionalities were tested and demonstrated successfully:

- OpenFlow 1.0
- Flow entry match to:
    - Ethernet type (recognize IPv4 and ARP)
    - Source IPv4 address (also carried within ARP when OF1.0 control is used)
    - Destination IPv4 address (also carried within ARP when OF1.0 control is used)
    - IP protocol (recognize TCP, UPD and ICMP)
    - Source TCP port
    - Destination TCP port
    - Destination UDP port

- Supported actions:
    - Forward to a port
    - Drop

- Flow entry add
- Flow entry remove
- Packet-in
- Packet-out

Streaming on demand in OpenFlow networks build upon EZappliances was demonstrated earlier during FIA2014 and TNC2014 conferences. All experience gained during these demonstrations preparation was used during tests of HAL for EZappliance.

Only one problem was related to establishing Q-in-Q mechanism used between PSNC and UNIVBRIS Island which was chosen during integration design phase for allowing PSNC to participate in multiple OFELIA VLAN-based slices. Due to incompatibility of switches in PSNC and UNIVBRIS Islands, VLAN swapping mechanism was used as a way of using separate VLANs in GEANT between PSNC and UNIVBRIS and in the OFELIA experiment.

2.3 Integration of L0 switch

In this section the integration of the Layer0 switches inside the UNIVBRIS OFELIA Island is demonstrated. Since UNIVBRIS was initially part of OFELIA many of the components had already been installed and most of the focus was to replace the software parts that were developed as part of the ALIEN project. The goal was to show the functionality provided by HAL and demonstrate how this can be used in the context of an existing OFELIA island.

2.3.1 Description of the integration

The following picture details the connectivity and the configuration of the aforementioned equipment. The optical switch datapaths based on the HAL developed in the context of ALIEN are installed on a Dell Poweredge 1950 server. The connectivity between the components inside UNIVBRIS Island is detailed in the picture below. Below there is a list with the physical equipment utilized for the UNIVBRIS OFELIA Island:

- 3 ADVA FSP3000 nodes
- 4 NEC IP8800/S3640-24T2XW OpenFlow 1.0 packet switches
- 2 XEN servers dedicated for VM creation (Dell Poweredge 1950)
- 1 OpenVPN server (connected to OFELIA's general hub)
- 2 servers for running OCF (Dell Poweredge 1950)

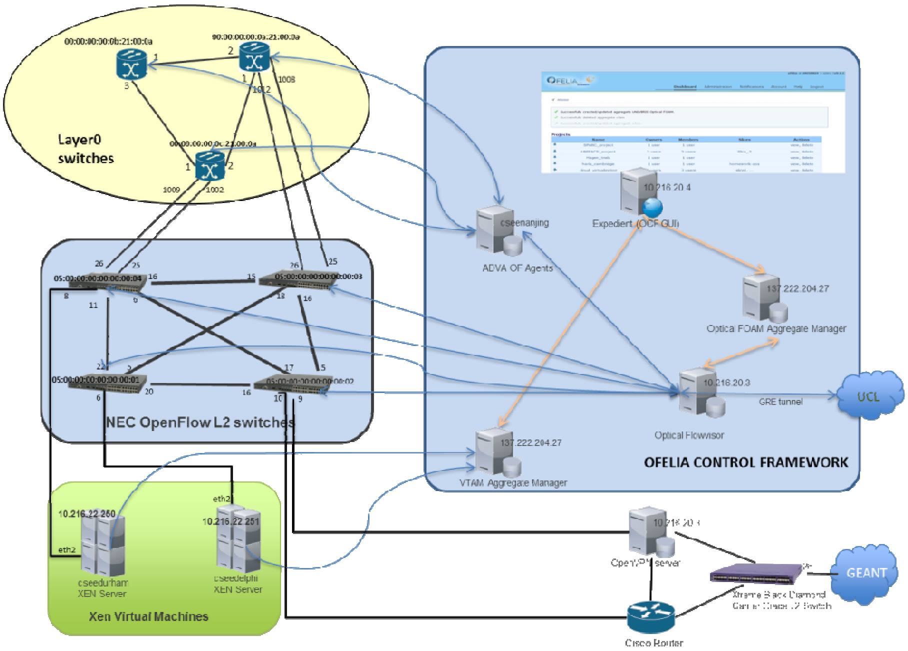
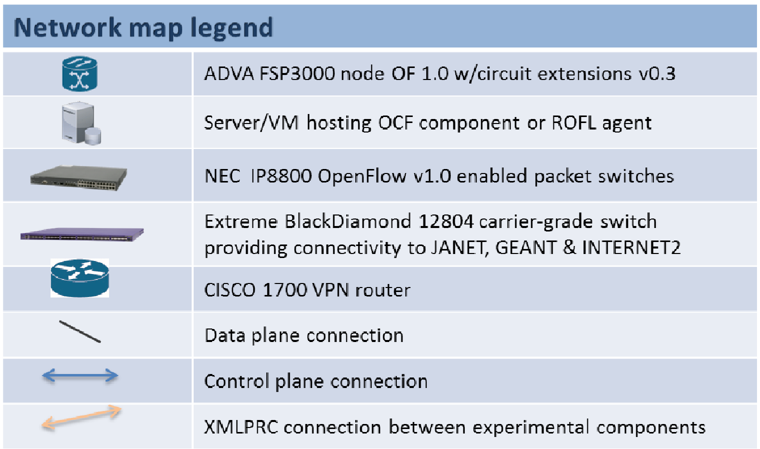


Figure 2-15 UNIVBRIS OFELIA Island network map



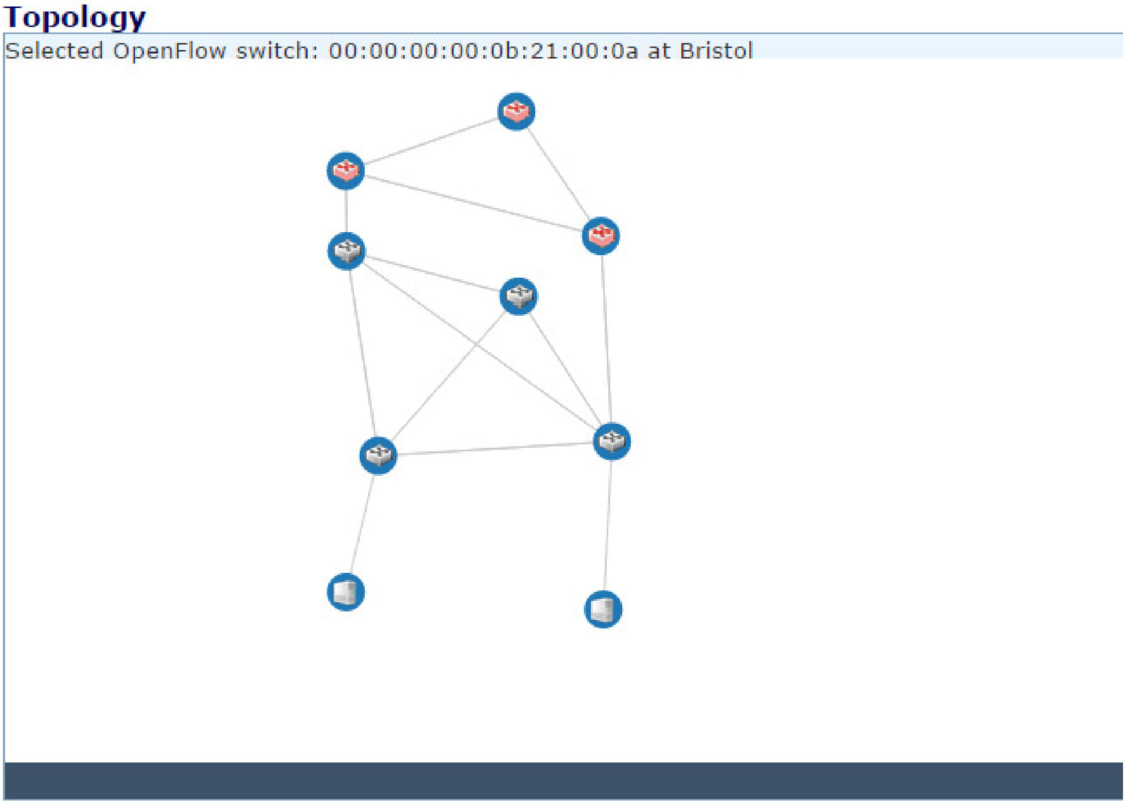
Network map legend	
	ADVA FSP3000 node OF 1.0 w/circuit extensions v0.3
	Server/VM hosting OCF component or ROFL agent
	NEC IP8800 OpenFlow v1.0 enabled packet switches
	Extreme BlackDiamond 12804 carrier-grade switch providing connectivity to JANET, GEANT & INTERNET2
	CISCO 1700 VPN router
╲	Data plane connection
⟷	Control plane connection
⟷	XMLPRC connection between experimental components

Figure 2-16 UNIVBRIS OFELIA Island network map legend

The ADVA FSP3000 OpenFlow enabled devices are part of the UNIVBRIS OFELIA test bed. The island also consists of 4 OpenFlow enabled L2 switches and 2 dedicated servers for running virtual machines. The virtual machines are used as end hosts and they are used to run experiments over the slices created through the Ofelia Control Framework. Furthermore we have deployed a number of components in order to allow experimentation of multiple users over the same resources. The additional resources for management and control of the test bed consist of:

- A server running the OFELIA control framework web interface
- A server running the Aggregate Managers (AM) of the island (Optical FOAM & VTAM)
- A server running the extended Optical Flowvisor
- A server running OpenVPN

The topology of the described island is also shown in the capture below which is taken from OFELIA’s expedient GUI.



Topology
Selected OpenFlow switch: 00:00:00:00:0b:21:00:0a at Bristol

Figure 2-17 UNIVBRIS OFELIA Island topology

Configuration of the equipment

The ROFL OpenFlow agents of the optical nodes are running on a server that points to the management interface (SNMP channel) of the optical devices. The datapath implementation acts like an agent for these devices which adds the OpenFlow capability to them. It is necessary to have multiple instances of the agent running to control multiple devices. Since this test bed has 3 optical nodes it is necessary to execute the agent 3 times (for each one of them) and use a different configuration each time. This is achieved by simply running the agent with a different configuration file that causes the agent to connect to a different device each time. The most interesting fields in that file are the host, which is the IP of the device's management interface and openflow field, which is the IP of the OpenFlow controller the devices is pointing to.

```
<config>
<host>10.0.34.10:161</host>
<community>private</community>
<trap>0.0.0.0:1620</trap>
<poll>0</poll>
<cpreload>10</cpreload>
<openflow>tcp:10.0.34.133</openflow>
</config>
```

The agents of optical switches are pointing to a special purpose controller called Flowvisor. This entity can act as a proxy between the device and multiple users' controllers and slice the resources of it acting as a virtualization engine for the user. For the purposes of OFELIA project [OFELIA] the original version of Flowvisor was extended in a way that is able to slice both in packet and the optical domains [OFV]. As it was shown in the previous network map (Figure 2-15), the L2 OpenFlow enabled are pointing directly to the Flowvisor. In the following snippet Flowvisor's fvctl tool is used to list the DPIDs of the OpenFlow resources visible to the Optical Flowvisor in UNIVBRIS Island. The test bed comprises of 4 OpenFlow enabled L2 switches and also with DPIDs 05:00:00:00:00:00:00:0X as well as a GEPON device with DPID 12:00:00:00:00:00:00:01 that appears as a packet switch and it is described in section 2.5 of this deliverable.

```
cseerio $ ./fvctl --passwd-file=fvadmin.passlistDevices
Device 0: 05:00:00:00:00:00:00:03
Device 1: 05:00:00:00:00:00:00:04
Device 2: 05:00:00:00:00:00:00:02
Device 3: 05:00:00:00:00:00:00:01
Device 4: 12:00:00:00:00:00:00:01
cseerio $
```

The circuit switch devices visible to UNIVBRIS extended Optical FlowVisor are listed using a different argument with the fvctl tool.

```
cseerio $ ./fvctl --passwd-file=fvadmin.passlistCDevices
Device 0: 00:00:00:00:0b:21:00:0a
Device 1: 00:00:00:00:0c:21:00:0a
Device 2: 00:00:00:00:0a:21:00:0a
cseerio $
```

Control configuration

The resources of UNIVBRIS OFELIA Island become available through the OFELIA Control Framework software stack [OCF]. The Optical Flowvisor is connected to the Optical FOAM that manages the users and the slices over the test bed's OpenFlow resources. This aggregate manager (AM) is the equivalent of VTAM, which is responsible for controlling the computational resources (virtual machines) of the island. The initial configuration of island imposed that all software components, like Flowvisor and Aggregate managers, to be under the OFELIA VPN and have an IP in the assigned range (10.216.20.0/22). However, since we are exposing our resources not only to OFELIA islands but to a broader range of SFA compatible test beds in the context of Fed4FIRE project [F4F], both these aggregate managers are accessible using a public IP.

Given the credentials, the aggregate managers can be added to a peering island using the following URLs:

```
FOAM: https://137.222.204.27:3626/core/legacyexpedientapi/xmlrpc/
VTAM: https://137.222.204.27:8445/xmlrpc/plugin/
```

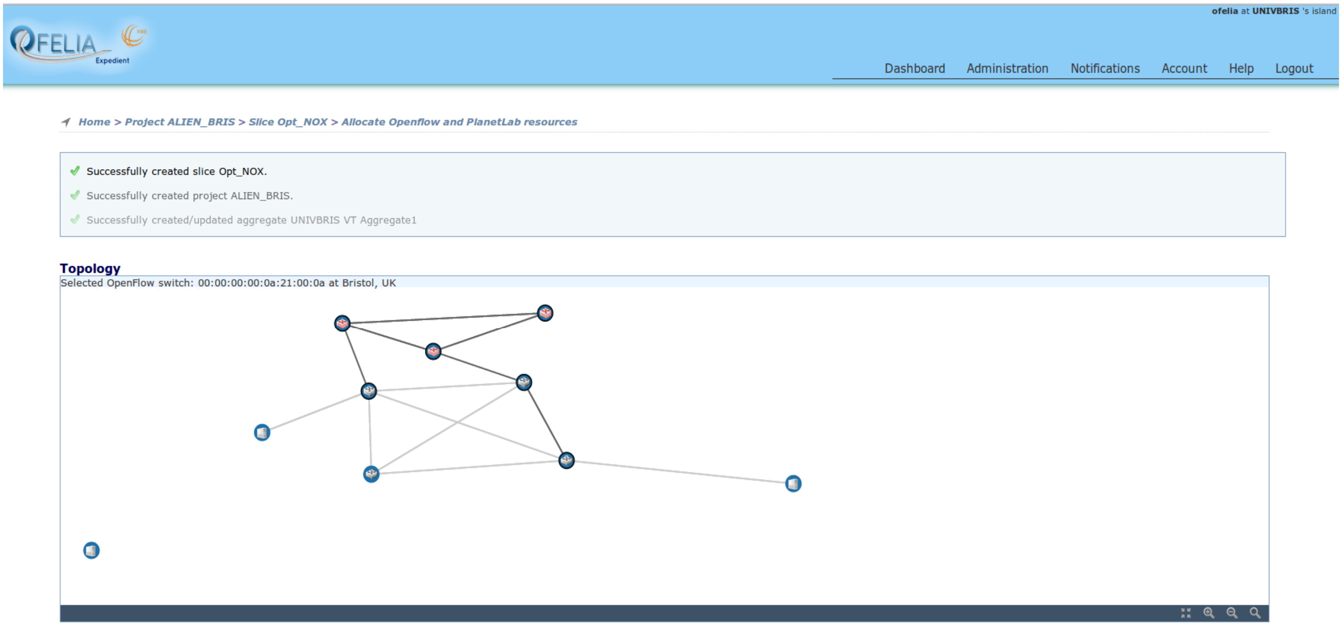
The experimenter will need to have an Ofelia account and a working VPN, in order to login to the expedient web interface and create a slice to conduct his/her experiment. The expedient's URL for UNIVBRIS Island is https://10.216.20.4/. The virtual machines created inside an OFELIA slice get an IP in the assigned range 10.216.20.0/22 which makes them accessible to the user inside the OFELIA VPN.

2.3.2 Functional testing

In the section below it is described the initial setup in OFELIA Control Framework and the steps followed to run the experiment.

1) OFELIA Resources involved in the testing

The Layer0 ADVA optical nodes were tested in the context of UNIVBRIS Island. It was created a project for testing the ALIEN equipment and a slice containing all the three optical nodes provided by UNIVBRIS and also three of the L2 OpenFlow-enabled switches. The screenshot below shows the network devices connected to each other and also the connections between the network and computational resources.



Home > Project ALIEN_BRIS > Slice Opt_NOX > Allocate Openflow and PlanetLab resources

✓ Successfully created slice Opt_NOX.

✓ Successfully created project ALIEN_BRIS.

✓ Successfully created/updated aggregate UNIVBRIS VT Aggregate1

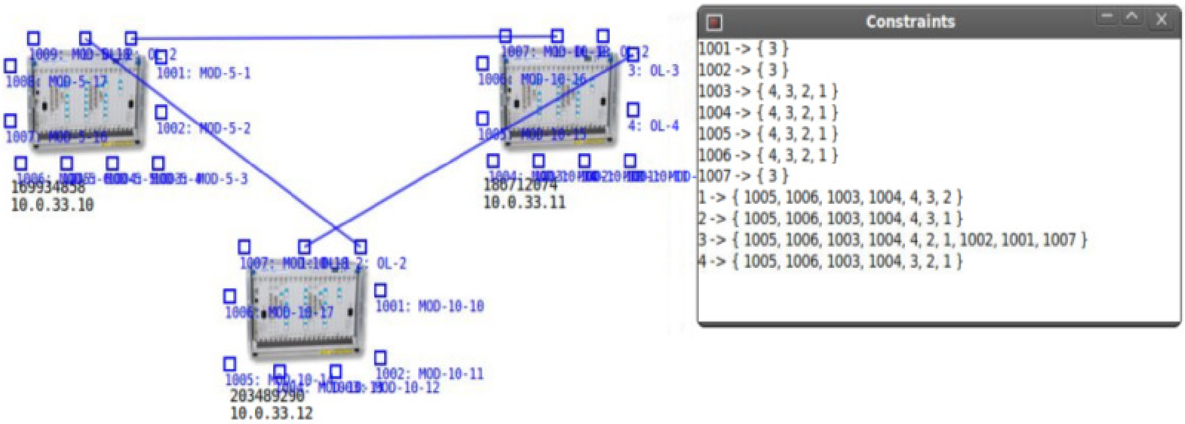
Topology

Selected OpenFlow switch: 00:00:00:00:0a:21:00:0a at Bristol, UK

Figure 2-18 Allocated slice for verifying UNIVBRIS ALIEN equipment

The lines in bold show the allocated links between the optical nodes and the L2 packet switches. The ADVA switches are connected in a ring topology, while 2 virtual machines are attached which act as a source/sink to demonstrate the flow establishment. The purpose of this test is to show the basic functionality of setting up a flow between the 2 VMs.

The extended hybrid OpenFlow controller (NOX) [OPEXP] can handle both packet and optical switches. It is responsible for the resource discovery and provisioning the path between the two end nodes. The developed OpenFlow agents are responsible for handling the OpenFlow messages sent by the controller and translating to commands interpretable by the device itself. In Figure 2-19 there can be seen the discovered optical switches, the physical connections between them and the switching constrains imposed for one of the ADVA ROADMs. So, for example, from port 1001 it is only possible to go to port 3, while from port 1003 can switch to ports 1, 2, 3 and 4.



1001 -> { 3 }
1002 -> { 3 }
1003 -> { 4, 3, 2, 1 }
1004 -> { 4, 3, 2, 1 }
1005 -> { 4, 3, 2, 1 }
1006 -> { 4, 3, 2, 1 }
1007 -> { 3 }
1 -> { 1005, 1006, 1003, 1004, 4, 3, 2 }
2 -> { 1005, 1006, 1003, 1004, 4, 3, 1 }
3 -> { 1005, 1006, 1003, 1004, 4, 2, 1, 1002, 1001, 1007 }
4 -> { 1005, 1006, 1003, 1004, 3, 2, 1 }

Figure 2-19 Optical node switches and switching constraints

2) Testing workflow and results

Below the steps required to run the experiment are provided in detail. It is assumed that the user has already reserved a slice for running the experiment with the appropriate flowspace. The slice given to NOX controller can control packets with VLAN 400 and wavelength 193.9mm.

- Step 0 (data plane application)
  - User login to the VMs hosted in servers provided by OFELIA
  - Setup ping between the VMs
  - VMs are not accessible from each other (ping unreachable)

- Step 1(control plane – OpenFlow)
  - Start NOX controller with *switch* and *lightpath* applications running on top of it
  - NOX exchanges OpenFlow messages with the optical devices and the packet switches through Optical FlowVisor (proxy)
  - Initially sending HELLO messages and subsequently FEATURES_REQUEST to discover the capabilities of the network elements.
  - L2 switch application installs rules in the packet switches to allow path setup between the end nodes.
  - The rules are applied to the switches using FLOW_MOD messages
  - Lightpath application sets up a path across the ADVA ROADMs
  - Cross connections are created by sending CFLOW_MOD and OOE_POWER_EQUALIZATION messages to the deployed agents

- Step 2 (data plane application)
  - VMs are able to ping each other since there is a valid path between them

- Step 3 (control plane – OpenFlow)
  - Sending CFLOW_MESSAGES to the ADVA ROADMs to delete the cross-connections and tear-down the lightpath

- Step 4 (data plane application)
  - Ping application cannot reach the target since there no established path now

In the experiment, the slice was started and the devices were pointed to the IP of the extended OpenFlow controller. The developed OpenFlow agents allowed the controller (NOX) to discover the optical nodes of the network while the packet ones were discovered by the hybrid OpenFlow controller [OPEXP] since they were OpenFlow compatible. Upon the start of the slice and the HELLO messages exchanged between the two entities, the controller sent a FEATURES_REQUEST message to the controlled devices. The devices replied by sending a FEATURES_REPLY message to the controller. The previous Figure 2-19 shows the GUI of an application running on top of the extended NOX controller called lightpath which presents the discovered optical nodes and the connections between them. Through the experimental messages developed [D3.3] the switching constraints of the optical node were also get, as shown in the right part of the figure.

A simple application (ping) was used to test the datapath between the end nodes of the slice. In the beginning of the experiment there was no valid path between the two end nodes. In order to create a valid datapath between the nodes the OpenFlow controller needs to install the rules in the packet switches and perform the cross connections in the optical ones. Thus, in the beginning there was no datapath between the VMs and packets failed to reach their destination. Subsequently, it was observed that the two end nodes were accessible and could ping each other. It should be noted that the lightpath across the optical domain appeared as a simple pipe between the L2 packet switches. Finally the lightpath created was tear-down and communication between the endpoints was interrupted.

2.3.3 Conclusions

During validation of ADVA FSP3000 optical nodes the following functions were successfully tested:

- OpenFlow 1.0 with circuit switch extensions (v0.3)
- Resource and peering features discovery
- Flow entry add
- Flow entry tear down
- Optical node switching constrains
- Optical packet integration

Lessons learned from integration in real scenarios:

- The ECHO_REQUEST – ECHO_REPLY interval was longer in ROFL library which caused the controller to disconnect and discover the optical switch continuously. This value was changed to meet the NOX requirements.

2.4 Integration of NetFPGA

During Alien project PUT created a new OFELIA island. Basing on developed under previous projects direct fibre to PSNC, it was decided to connect to Ofelia datapath via this medium through PSNC Island. OFELIA Control Framework was realized via public part of Internet. The goal was to connect to other laboratories and be able to be a part of much bigger experiments than the ones that can be realized in standalone laboratory. OCF offers many advanced and practically tested and verified functionalities and mechanisms useful for researchers and scientists. Becoming a part of Ofelia extends up the possibilities of PUT. The aim of this achievement in Alien project was also to show, that the developed stack is able to be controlled by OCF.

2.4.1 Description of the integration

Connection to OFELIA control structures has been realized by deploying VPN connections and running router, which connects IP network designated for PUT Island (with address spaces 10.216.66.0/24) with OCF maintained by OFELIA. Data-plane has been realized by VLAN 602, which connects PUT laboratory with PSNC Island and further OFELIA structures. By those two actions an independent and complete Island with ID 18 has been created at PUT facilities. Detailed topology (including IP addresses of particular functional modules) is shown in Figure 2-20.

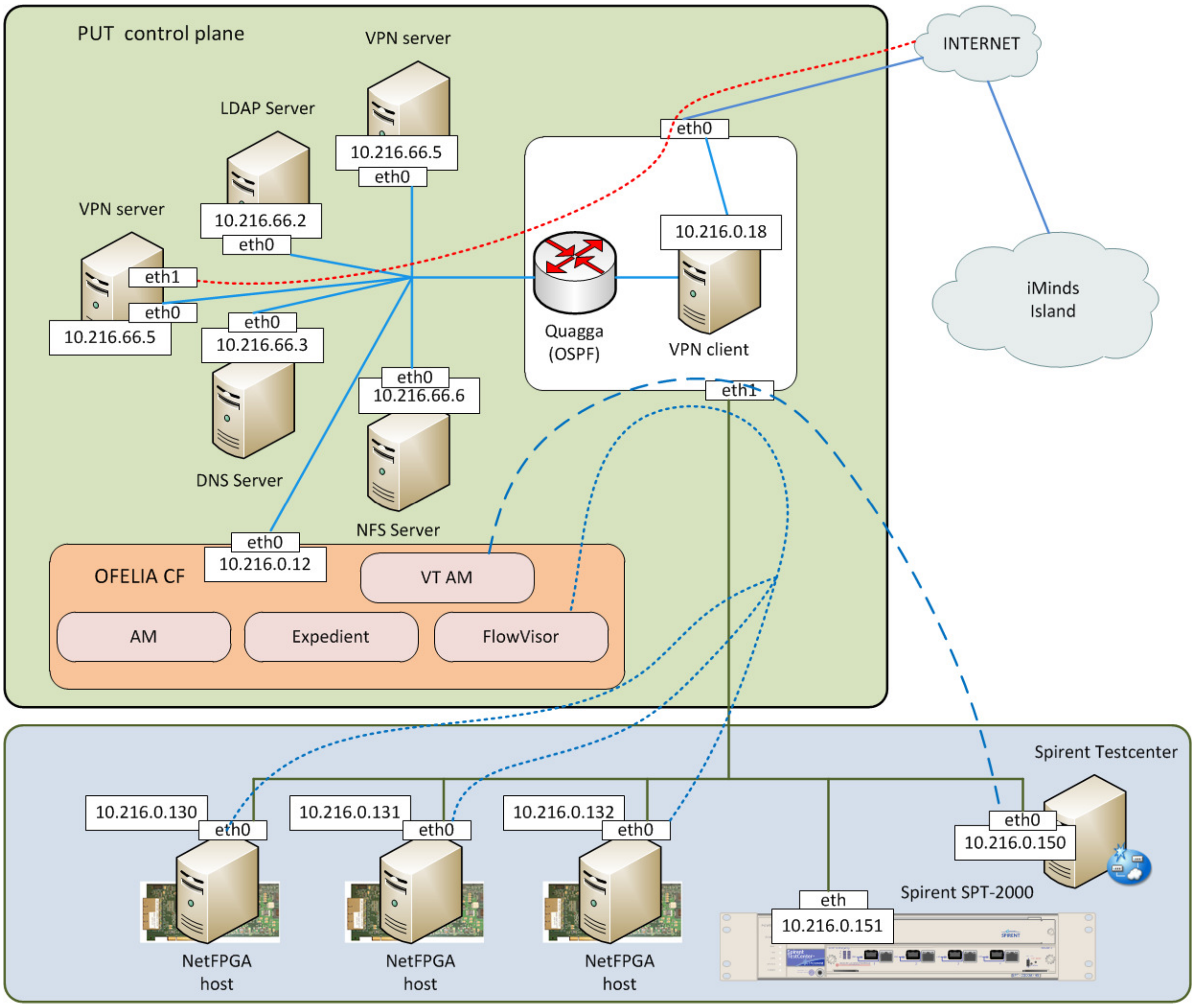


Figure 2-20 Control plane configuration

1) PUT Island

The primary integration and validation of NetFPGA developments have been performed using the following resources:

- 3 hosts with NetFPGA cards exposed as OF switches
- 1 VM with OpenFlow controller
- 1 VM with Spirent Test Center
- Spirent SPT-2000 (chassis with controller acts as packet generator)

- 2 virtual and 2 real hosts

Configuration of the equipment

In the test it was used Spirent network testing environment which consists of hardware platform (SPT-2000) and software application (Spirent TestCenter). Tested network was composed of three hosts, each of them contained one NetFPGA card - these sets (containing HAL) act as three OF switches. These switches were controlled by NOX OpenFlow controller via HAL. Spirent TestCenter allows generating flows of packets directed to particular destination. On the other hand, this application can analyze received traffic in terms of routing correctness and packets latency. It is a very useful tool to validate HAL implementation. In presented configuration one interface of SPT-2000 acts as full configured packet generator. Packets from this interface are transferred through (one, two or three - depending on destination address and flows are determined by NOX controller) NetFPGA switches and data paths to remaining three interfaces of SPT-2000.

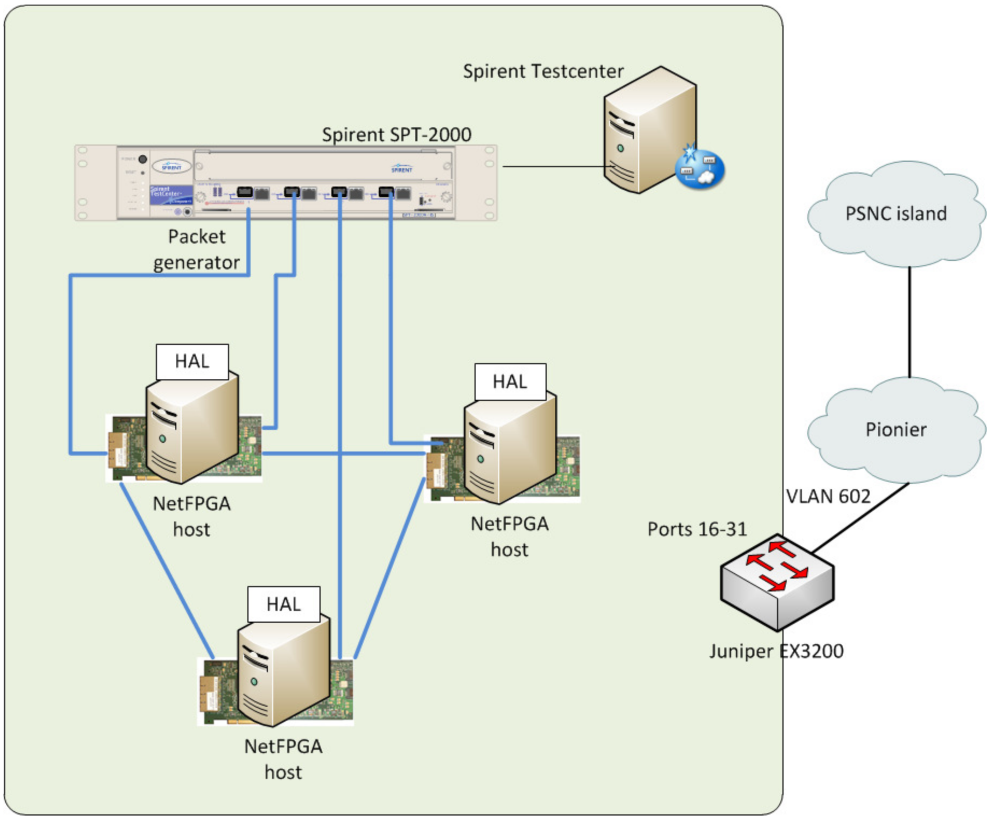


Figure 2-21 Data Plane configuration

Control configuration

The validation tests of HAL implementation in NetFPGA cards were realized in the network environment presented in Figure 2-21. This network realizes transport functions for control plane and is integrated with global OFELIA control network. The configuration of particular network nodes in control network complies the OFELIA recommendations. All IP addresses are from the pool of addresses provided by OFELIA administration assigned to the PUT OFELIA Island.

QUAGGA daemon configuration file:

```
! -*- ospf -*-
!
hostname ospfdPUT
!
interface tap1
ip ospf authentication message-digest
ip ospf message-digest-key 1 md5 xxxxxx
ip ospf cost 5
interface tap2
ip ospf authentication message-digest
ip ospf message-digest-key 1 md5 xxxxxx
ip ospf cost 15
!
router ospf
router-id 10.216.0.18
log-adjacency-changes detail
network 10.216.0.0/23 area 0
network 10.216.2.0/23 area 0
network 10.216.66.0/24 area 18
passive-interface eth0
passive-interface eth1
area 0 range 10.216.0.0/22
area 18 range 10.216.66.0/24
area 0 authentication message-digest
area 18 authentication message-digest
!
!
!log stdout
```

2) NetFPGA integration in OFELIA

Due to earlier references from OFELIA the PUT Island was set up from dedicated elements. They are available at the links and addresses listed below:

```
dns.put.fp7-ofelia.eu              10.216.66.3    DNS
gw.put.fp7-ofelia.eu               10.216.66.129  default gw to Ofelia
ocf.put.fp7-ofelia.eu              10.216.66.12   Local OCF2
```

As the devices connected to OFELIA and the physical background of the island, it was decided to expose several devices and allow other OFELIA users to use them via IP connections. All of used nodes have DNS and IP configuration, access to them is granted by user/password combination or LDAP functionality. On each host with hardware platform there is running xDPd which is connected to OCF. Names of hosts, which describe installed and controlled devices and their IP addresses are listed below:

```
ez1.put.fp7-ofelia.eu              10.216.66.131  xDPd for EzChip 1
```

```
ez2.put.fp7-ofelia.eu                   10.216.66.132   xDPd for EzChip 2
netfpga1.put.fp7-ofelia.eu              10.216.66.133   xDPd for NetFPGA1
netfpga2.put.fp7-ofelia.eu              10.216.66.201   xDPd for NetFPGA2
netfpga3.put.fp7-ofelia.eu              10.216.66.202   xDPd for NetFPGA3
netfpga4. put.fp7-ofelia.eu             10.216.66.203   xDPd for NetFPGA4
netfpga5.put.fp7-ofelia.eu              10.216.66.204   xDPd for NetFPGA5
```

In some cases one host can have more than one IP address while such a host serves more than one hardware platform - in PUT laboratory netfpga2 and netfpga3 are placed in the same PC. All the equipments are visible in OFELIA Expedient as independent switches. It was decided to hide in DPID (DataPathID) some information, which makes it easier to maintain logical structure of investigated slice of network. Each DPID contains 16 signs and starts with value 0x12, which denotes in hex 18 - number assigned as ID of PUT Island. The next 12 digits are reserved for IP address of host which handle xDPd for this hardware platform. (IP address has to be presented in form 000.000.000.000 with leading zeros in each of its octet). Last two digits are used to distinguish hardware platforms hosted in the same machine - in most cases it would be value 01, but in some cases (for example for machine with two NetFPGA cards) it could be 02 (or even more). Example of a DPID created with these rules looks like this: 12:01:02:16:06:61:34:02. Configuration file for xDPd maintaining one of two NetFPGA cards placed in host with IP 010.216.066.201 is presented below:

```
#Node C
config:{
   openflow:{
     logical-switches:{
             #Name of the switch dp0
             dp0:{
                     #Most complex configuration
                     dpid = "0x1201021606620102"; #hexadecimal
                     version = 1.0;
                     description="Switch C";

                     #Controller connection(s)
                     controller-connections:{
                             main:{
                             remote-hostname="10.216.66.12";
                             remote-port = 6633;
                             };
                     };

                     #Tables and MA
                     num-of-tables=1;

#Physical ports attached to this logical switch. This is mandatory
#The order and position in the array dictates the number of
# 1 -> veth0, 2 -> veth2, 3 -> veth4, 4 -> veth6
ports = ("nf2c4", "nf2c5", "nf2c6", "nf2c7");
}; };        };       };
```

As it is easy to see, the DPID in this file contains island ID, IP address and number of card (in this case - 02). It is also visible that ports assigned to this switch are named starting not from 0, but from 4.

2.4.2 Functional testing

The functional testing performed with the usage of OF switches, based on NetFPGA cards with HAL, demonstrates how these switches can be used as transport network for packets generated and received by Spirent testing environment. A very important feature of SPT-2000 is the fully configured hardware packet generator and receiver. It is possible to generate exact number of packets (or generate packets with configured speed) with a given value of fields. Moreover, Spirent device can be aware of which interface generated packet should appear and which fields of a packet must be modified by network devices.

1) OFELIA Resources involved in the testing

Many tests were run, starting from basic test for connectivity, throughput and time for realizing particular procedures. Most of them will be extended on other platforms as experiment under task T5.3 and described in next deliverable.

Basic test consisted on setting up a network with three switches (connections between them are marked with green in Figure 2-22) and connecting to them two nodes (with blue connections). Virtual machines were tested, which were provided by OCF and physical hosts located in PUT lab. In both cases, source and sink of test traffic were connected to the test network with proper VLAN configuration or physical connections respectively. To test the integration with OCF this portal was used to validate the proper availability of nodes in L2 segment created by three OF switches (by clearing and checking ARP tables in both hosts), their connectivity in IP domain (with ping) and throughput of connections with scp (secure copy via SSH connection).

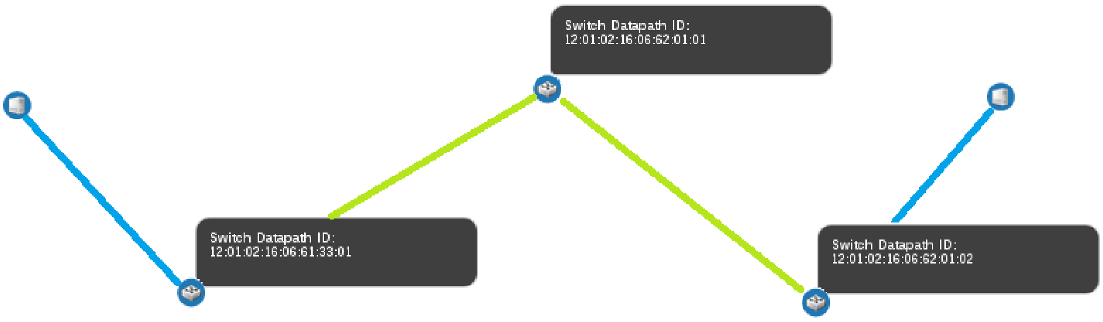


Figure 2-22 Logical topology of nodes (physical and virtual) used during availability tests seen by Ofelia Expedient

2) Testing workflow and results

Basic functional test of the PUT implementation consisted on the validation of the proper serving of ARP and ICMP requests and replies in both cases. It was performed in the topology presented in Figure 2-23. There was an OpenFlow switch, a host with OpenFlow controller (actually it was the same PC, those two modules were available at the same IP address) and two host (ICMP source and destination). It was also measured the time in different cases, where different cases had different state in control structures (as ARP table in hosts and flowtable in switch).

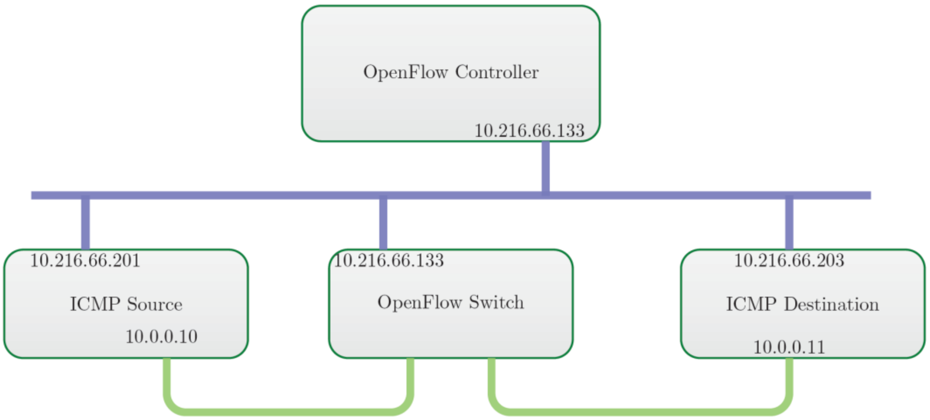


Figure 2-23 Topology for functional test

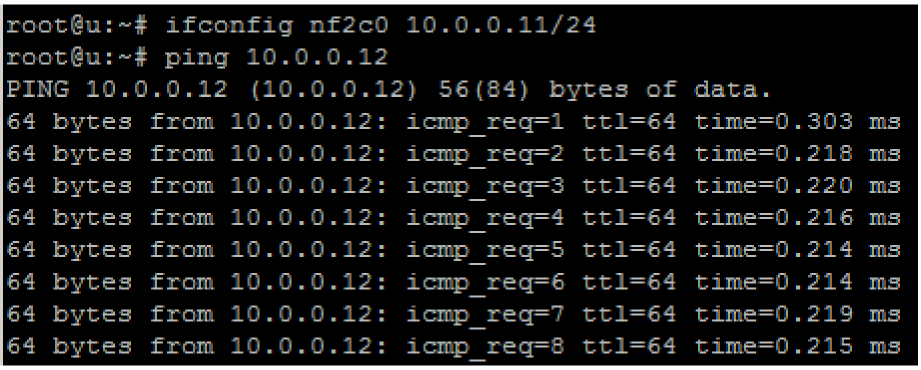
Important steps of test:

- Step 1: Prepare physical (connections) and logical (MAC/IP configuration) topology (as it shown in Figure 2-23).

As hardware platform NetFPGA card was used, but it worked as a switch, so its MAC addresses were not important, because all interfaces worked transparently in L2, MAC addresses of ICMP source and destination had to be different (this was very important, because L2 switches distinguish destinations of frames by their MAC addresses, they assume unique MACs and do not work properly with the same MACS in different hosts). Also setting up controller was realized in this step.

- Step 2: Perform simple PING test with default behaviour of OpenFlow switch.

In this case the typical behaviour was observed: first one or several packets of burst reported much bigger delay than next ones. The cause of these mechanisms comes from common treatment of time of real trip and time which is necessary to fill all control information (as ARP tables - MAC-IP assignment in both source and destination of ICMP packets, and also flowmods installation). In the log presented at Figure 2-24, there is a visible difference between the first and the rest of the packets. The difference between the first and the next packets depends on the speed of host PCs, PC which operates controller and several network mechanisms, it can be different in different laboratory. Results of time measurement in different cases are presented at the end of this section.



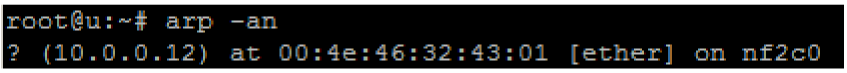
```
root@u:~# ifconfig nf2c0 10.0.0.11/24
root@u:~# ping 10.0.0.12
PING 10.0.0.12 (10.0.0.12) 56(84) bytes of data.
64 bytes from 10.0.0.12: icmp_req=1 ttl=64 time=0.303 ms
64 bytes from 10.0.0.12: icmp_req=2 ttl=64 time=0.218 ms
64 bytes from 10.0.0.12: icmp_req=3 ttl=64 time=0.220 ms
64 bytes from 10.0.0.12: icmp_req=4 ttl=64 time=0.216 ms
64 bytes from 10.0.0.12: icmp_req=5 ttl=64 time=0.214 ms
64 bytes from 10.0.0.12: icmp_req=6 ttl=64 time=0.214 ms
64 bytes from 10.0.0.12: icmp_req=7 ttl=64 time=0.219 ms
64 bytes from 10.0.0.12: icmp_req=8 ttl=64 time=0.215 ms
```

Figure 2-24 Typical result of PINGing remote host - there is visible significant difference in time for first packet and the rest of them

In this step time when there were no ARP in hosts and no Flows in switch (first packet) was measured, as well as the case with full information - second and further packets. Results of time measurement in all different cases are presented at the end of this section.

- Step 3: Perform simple PING test with no ARP information but with full flow tables.

It is possible to artificially generate a situation where hosts have no information about destination MAC addresses, but flows in the switch are installed (for example from previous experiments). After this step ARP table in host contained all necessary information to send ICMP packets.



```
root@u:~# arp -an
? (10.0.0.12) at 00:4e:46:32:43:01 [ether] on nf2c0
```

Figure 2-25 Content of local ARP table in host after step 3

- Step 4: Perform simple PING test with ARP info, but no flows in switch.

It is possible to generate situation where hosts know proper destination MACs (no arp process is required in such a case), but in the switch there is no flows and they have to be installed. This step is complementary with step 3.

Results of test for the four different cases are presented in Table 2-1.

ARP	FlowMod	av. time (us)
no	no	310
no	yes	300
yes	no	298
yes	yes	216

Table 2-1 Times of pinging in four different states of control tables

2.4.3 Conclusions

During the validation of NetFPGA ALIEN platform the following OpenFlow actions and HAL functions were tested and demonstrated successfully:

- Flow entry add
- Flow entry remove
- Packet-in
- Packet-out
- Flow entry searching

During the tests it was found that it is very important to plan tests and experiments from early stage to last one, where the last one consists on analysing of the results and measurements. Several times there had to repeat some experiments to gather all necessary measurements and statistics. It is good to have prepared scripts or even dedicated software which manages tests (sends out and receives control packets), with such a tool it is easier to perform the same tests in different topologies. Also professional tools (e.g. Spirent Test Center) are very useful, because they offer advanced and convenient functionality (for example - saving test configuration to file). Technical details are also very important. Users have to be very careful with the configuration of IP addresses; in one of the tests, a simple mistake was made (with IP prefix length), but this mistake was very hard to found as its results were strange and caused improper network behaviour. In one case it was necessary to modify settings of one virtual machine from OCF (the amount of RAM was not sufficient) to obtain proper performance. In some cases problems were found which were solved only by restarting investigated mechanism (it can suggest some hidden bug, which was not caught with precision and it is considered to require closer look at).

2.5 Integration of GEPON

2.5.1 Description of the integration

The GEPON deployment has added one switch with four ports and three VMs to the Bristol OFELIA Island. The switch represents a virtualization of a distributed switching system consisting of the OLT and three ONU.

Instead of being an island of its own the test equipment at UCL has been added to the Bristol Island.

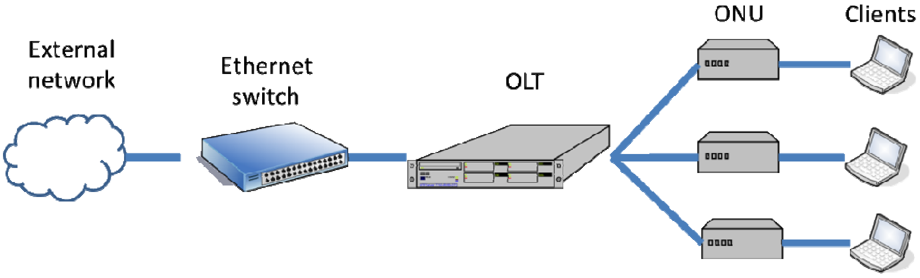


Figure 2-26 Usual layout of the GEPON without an OpenFlow deployment

Configuration of the equipment

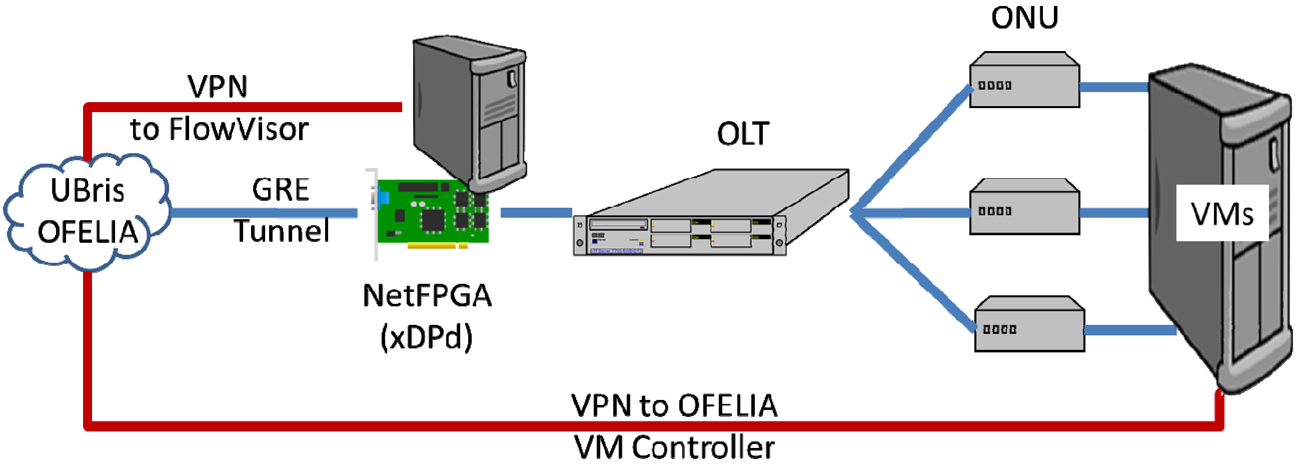


Figure 2-27 Schema of the configuration of test bed for UCL testing in OFELIA

The GEPON installation has been aided by two custom PCs constructed for the project as shown in Figure 2-27. The PC on the left (referred to as the front-end as it faces the network) is an OpenFlow switch, this can be a software switch or a hardware switch working using the NetFPGA card installed in this machine. In the tests a software switch was used. This PC sits in front of the OLT. It runs an OpenFlow switch and the xCPd software previously described in [D3.3]; the xCPd is a control path element, only similar to the xDPd (data path element) in the rofl underling it. The job of this software is to make the combination of OLT, OpenFlow Switch and 3xONU appear instead as a single four port switch. The control plane connection to Bristol is achieved using a VPN and this carries control commands between xCPd and the FlowVisor within the Bristol OFELIA Island. As far as the FlowVisor is concerned, xCPd is an OpenFlow switch. The original intent was that the data plane connection was carried over a GRE tunnel, however, logistical difficulties have postponed this and a second VPN carries the data plane to the Bristol OFELIA Island.

The second PC sits behind the three active ONU. It has a four port NIC which connects to three ONU. The fourth port was used only in the local OFtest setting in WP3. Another VPN connection to the Bristol OFELIA Island allows control messages for VMs to be sent.

Control configuration

There are several configuration aspects that should be mentioned.

The control plane for OpenFlow is carried over a VPN to the Bristol Island, configured using openvpn. The data plane to Bristol is carried over a GRE tunnel as previously mentioned. The control plane for the VMs to the OFELIA Control Framework in Bristol is carried over a second VPN residing on the access facing PC (on the right in the figure).

As described in [D3.3] xCPd presents the ports of the OLT and n ONUs as if they were the ports of a large n+1 port switch. xCPd uses a common configuration file with xDPd in order to simplify configuration. This file is as follows:

```
#Configuration for UCL GEPON hardware testbed
```

```
config:{

   openflow:{
     logical-switches:{
             #Name of the switch dp0
             dp0:{
                     dpid = "0x12:00:00:00:00:00:00:01"; #Must be hexadecimal
                     version = 1.0;
                     description="UCL GEPON virtualised switch";

                     #Controller
                     mode="active";
master-controller-ip="127.0.0.1";  # connection to xcpd
master-controller-port=16633;
                     reconnect-time=1; #seconds

                     #Tables and MA
                     num-of-tables=1;

                     #These are the real ports on the proxy machine.
                     ports = ("tap0", "eth4");

             };
     };
   };
xcpd:{
   higher-controller-ip="10.216.20.3";  # Bristol
higher-controller-port=6633;
upward-mode="active";
virtual-ports:{
            dp0: {
port1:{
physical= "tap0";
mac= "00:00:00:00:10:01";
                };
port2:{
physical= "eth4";
vlan=10;
mac= "00:00:00:00:10:02";
                };
port3:{
physical= "eth4";
vlan=11;
mac= "00:00:00:00:10:03";
                };
port4:{
```

```
physical= "eth4";
vlan=12;
mac= "00:00:00:00:10:04";
                };
            };
        };
    };
 };
```

2.5.2 Functional testing

The functional testing of the GEPON device over OFELIA used a simple firewall application based upon the python based pox controller. The first task was to ensure that the connection to the Bristol OFELIA Control Framework was up and running. This was divided in three parts.

One part was ensuring that an OpenFlow connection was possible between xCPd and the FlowVisor at Bristol; it was achieved with a VPN between Bristol and UCL. Another part was ensuring that a data connection could be made between the front-end PC and the OFELIA island at Bristol, which was intended to be achieved with a GRE tunnel but is currently a VPN. Finally, it should be ensured that VMs could be started on the UCL OFELIA Control Framework running on the rightmost PC in the diagram; that one was achieved with a third VPN and installation of the OCF OXA software that controls the start up and shut down of VMs. For technical reasons (due to software incompatibilities) the OXA agent was, itself, installed within a VM.

1) OFELIA Resources involved in the testing

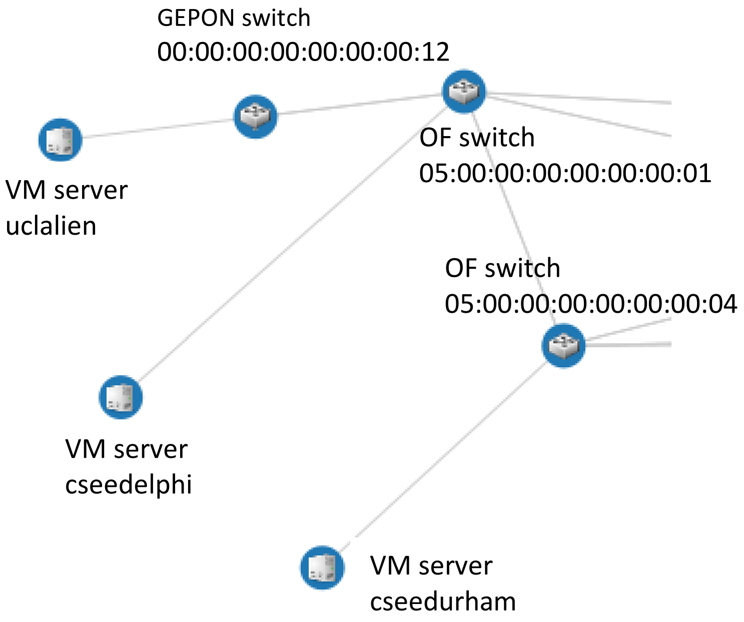
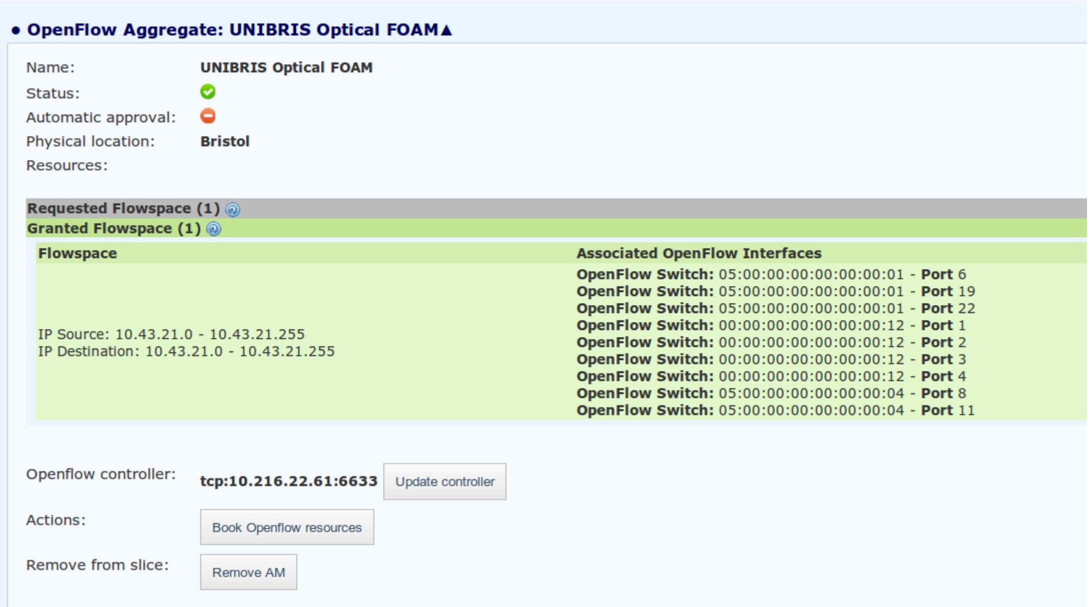


Figure 2-28 OFELIA resources involved in the UCL GEPON testing

The figure above shows the section of the OFELIA island at Bristol where the UCL GEPON attaches to the Bristol Island. For the tests reported here, two other switches were used and four VMs, one at UCL and three at Bristol. The GEPON switch had a VM server sitting behind it that could create VMs that link to any of the ONU facing ports of the GEPON.



• **OpenFlow Aggregate: UNIBRIS Optical FOAM▲**

Name:	**UNIBRIS Optical FOAM**
Status:	✓
Automatic approval:	⊖
Physical location:	**Bristol**
Resources:	

Requested Flowspace (1)

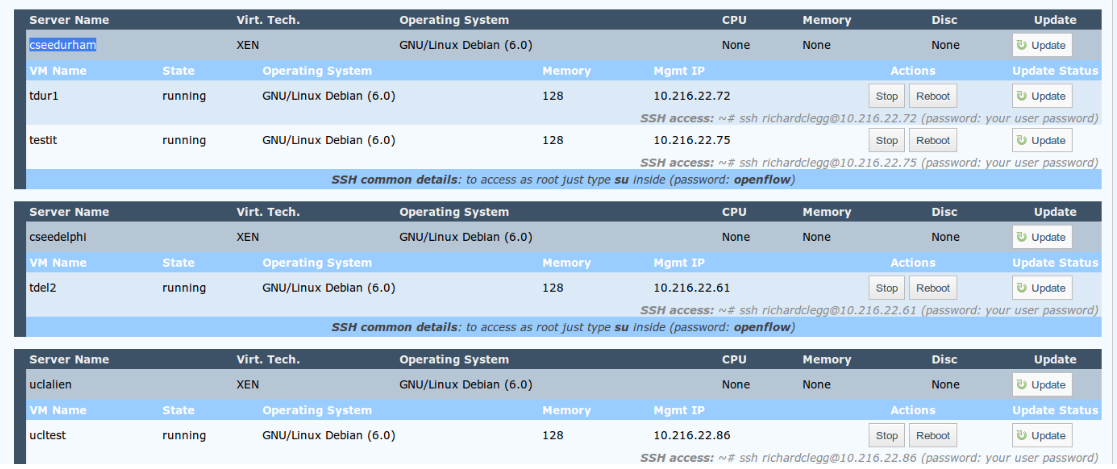
Granted Flowspace (1)

Flowspace	**Associated OpenFlow Interfaces**
IP Source: 10.43.21.0 - 10.43.21.255 IP Destination: 10.43.21.0 - 10.43.21.255	**OpenFlow Switch:** 05:00:00:00:00:00:00:01 - **Port** 6 **OpenFlow Switch:** 05:00:00:00:00:00:00:01 - **Port** 19 **OpenFlow Switch:** 05:00:00:00:00:00:00:01 - **Port** 22 **OpenFlow Switch:** 00:00:00:00:00:00:00:12 - **Port** 1 **OpenFlow Switch:** 00:00:00:00:00:00:00:12 - **Port** 2 **OpenFlow Switch:** 00:00:00:00:00:00:00:12 - **Port** 3 **OpenFlow Switch:** 00:00:00:00:00:00:00:12 - **Port** 4 **OpenFlow Switch:** 05:00:00:00:00:00:00:04 - **Port** 8 **OpenFlow Switch:** 05:00:00:00:00:00:00:04 - **Port** 11

Openflow controller:	**tcp:10.216.22.61:6633** Update controller
Actions:	Book Openflow resources
Remove from slice:	Remove AM

Figure 2-29 OFELIA resources booking page –UCL's GEPON slice

The above Figure 2-29 shows the OFELIA resources booked. Because, as mentioned in deliverable [D3.3] the GEPON cannot easily deal with VLANs the slice is a slice of IP space – in specific, the network 10.43.21.0/24. The OF controller is on 10.215.22.61 port 6633 as will be seen in the next Figure 2-30, this is the VM server in the Bristol test bed known as cseedelphi.



Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc	Update
cseedurham	XEN	GNU/Linux Debian (6.0)	None	None	None	Update

VM Name	State	Operating System	Memory	Mgmt IP	Actions	Update Status
tdur1	running	GNU/Linux Debian (6.0)	128	10.216.22.72	Stop Reboot	Update
testit	running	GNU/Linux Debian (6.0)	128	10.216.22.75	Stop Reboot	Update

SSH access: ~# ssh richardclegg@10.216.22.72 (password: your user password)

SSH access: ~# ssh richardclegg@10.216.22.75 (password: your user password)

***SSH common details**: to access as root just type **su** inside (password: **openflow**)*

Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc	Update
cseedelphi	XEN	GNU/Linux Debian (6.0)	None	None	None	Update

VM Name	State	Operating System	Memory	Mgmt IP	Actions	Update Status
tdel2	running	GNU/Linux Debian (6.0)	128	10.216.22.61	Stop Reboot	Update

SSH access: ~# ssh richardclegg@10.216.22.61 (password: your user password)

***SSH common details**: to access as root just type **su** inside (password: **openflow**)*

Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc	Update
uclalien	XEN	GNU/Linux Debian (6.0)	None	None	None	Update

VM Name	State	Operating System	Memory	Mgmt IP	Actions	Update Status
ucltest	running	GNU/Linux Debian (6.0)	128	10.216.22.86	Stop Reboot	Update

SSH access: ~# ssh richardclegg@10.216.22.86 (password: your user password)

Figure 2-30 UCL's VM visibility in the OFELIA-UNIVBRIS Control Framework

This figure shows the created VMs as seen by the OFELIA Control Framework. One VM is created at UCL and three at Bristol. Once logged in these machines can also be assigned internal IP addresses with which they can refer to each other.

VM name	VM host	eth1 IP	MGMT IP
tdur1	(Bristol cseedurham server)	10.43.21.1	10.216.22.72
testit	(Bristol cseedurham server)	10.43.21.4	10.216.22.75
tdel2	(Bristol cseedelphi server)	10.43.21.2	10.216.22.61
ucltest	(UCL uclalien server)	10.43.21.3	10.216.22.86

Table 2-2 VMs and internal IP addresses in the UCL GEPON testing

So, for example, viewed from ucltest the interface used for experiments was:

```
richardclegg@ucltest:~/code$ sudo ifconfig eth1
eth1 Link encap:Ethernet HWaddr 02:06:00:00:00:4f
inet addr:10.43.21.3 Bcast:10.43.21.255 Mask:255.255.255.0
inet6 addr: fe80::6:ff:fe00:4f/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:6292 errors:0 dropped:0 overruns:0 frame:0
TX packets:7293 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:685077 (669.0 KiB) TX bytes:578232 (564.6 KiB)
```

2) Testing workflows and results

The test OpenFlow application was a very simple routing application based on the pox l3_learning forwarding component that also designated one or more switches as firewalls that could reject traffic based upon match rules and placed FlowMods to drop that traffic.

The test environment was set up as follows:

- Step 0:
  - Set the GEPON to forward VLANs appropriately (VLAN 10 to ONU1, VLAN 11 to ONU2, VLAN 12 to ONU 3).
- Step 1:
  - Ensure that all three VPNs are connected to Bristol.
- Step 2:
  - Ensure xCPd is running on the front-facing host and is connected to the underlying OF switch (in this case this involves starting an underlying software switch).
- Step 3:
  - Start all four VMs using the OFELIA expedient interface.
- Step 4:
  - Ensure the VMs have the correct IP addresses and each are in the same subnet.
- Step 5:
  - Start the OFC firewall application.

The OFC firewall application that was written for this deliverable has a simple rules based system. In the tests described here the only traffic blocked is that destined for port 80 and going either to or from 10.43.21.4. All other traffic should be allowed.

The first test was a ping test to establish connectivity between the UCL VM and the Bristol Island.

10.43.21.3 (UCL) did a ping 10.43.21.4 (Bristol) and vice versa.

```
richardclegg@ucltest:~/code$ ping 10.43.21.4
PING 10.43.21.4 (10.43.21.4) 56(84) bytes of data.
64 bytes from 10.43.21.4: icmp_req=1 ttl=64 time=175 ms
64 bytes from 10.43.21.4: icmp_req=2 ttl=64 time=9.90 ms
64 bytes from 10.43.21.4: icmp_req=3 ttl=64 time=9.05 ms
64 bytes from 10.43.21.4: icmp_req=4 ttl=64 time=8.79 ms
64 bytes from 10.43.21.4: icmp_req=5 ttl=64 time=8.51 ms

richardclegg@testit:~$ ping 10.43.21.3
PING 10.43.21.3 (10.43.21.3) 56(84) bytes of data.
64 bytes from 10.43.21.3: icmp_req=1 ttl=64 time=351 ms
64 bytes from 10.43.21.3: icmp_req=2 ttl=64 time=10.0 ms
64 bytes from 10.43.21.3: icmp_req=3 ttl=64 time=6.89 ms
```

The initial long connection time (175 ms and 351 ms respectively) was because the ping first requires set up via ARP, so an initial ARP request was sent to the OFC, triggered and ARP flood and ARP response before rules were put in place to allow the ping response (this is standard for the l3 learning switch module).

TCP dump from the UCL test bed ingress VPN shows the packets arriving:

```
10.43.21.3 > 10.43.21.4: ICMP echo request, id 1683, seq 11, length 64
22:43:25.915484 02:06:00:00:00:4f > 02:06:00:00:00:15, ethertype IPv4 (0x0800),
   length 98: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1),
   length 84)
10.43.21.3 > 10.43.21.4: ICMP echo request, id 1683, seq 12, length 64
22:43:26.539587 00:00:00:00:00:00 > 01:23:20:00:00:01, ethertype LLDP (0x88cc),
   length 166: LLDP, length 152
Chassis ID TLV (1), length 7
Subtype MAC address (4): 00:00:00:00:00:00
Port ID TLV (2), length 3
Subtype Port component (2):
Time to Live TLV (3), length 2: TTL 112s
System Description TLV (6), length 8
Unknown TLV (101), length 254
[|LLDP]
```

```
22:43:26.916673 02:06:00:00:00:4f > 02:06:00:00:00:15, ethertype IPv4 (0x0800),
   length 98: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1),
   length 84)
10.43.21.3 > 10.43.21.4: ICMP echo request, id 1683, seq 13, length 64
22:43:27.917622 02:06:00:00:00:4f > 02:06:00:00:00:15, ethertype IPv4 (0x0800),
   length 98: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1),
   length 84)
```

Note the additional LLDP packets – these are ignored by the routing module.

The OFC can be made to verbosely report its status:

```
INFO:forwarding.firewall:360287970189639684  8  ARP  request  10.43.21.4  =>
   10.43.21.3
INFO:forwarding.firewall:360287970189639684 8 flooding ARP request 10.43.21.4
   => 10.43.21.3
INFO:forwarding.firewall:360287970189639681  22  ARP  request  10.43.21.4  =>
   10.43.21.3
INFO:forwarding.firewall:360287970189639681 22 learned 10.43.21.4
INFO:forwarding.firewall:360287970189639681 22 flooding ARP request 10.43.21.4
   => 10.43.21.3
INFO:forwarding.firewall:18 1 ARP request 10.43.21.4 => 10.43.21.3
INFO:forwarding.firewall:18 1 learned 10.43.21.4
INFO:forwarding.firewall:18 1 flooding ARP request 10.43.21.4 => 10.43.21.3
INFO:forwarding.firewall:18 2 ARP reply 10.43.21.3 => 10.43.21.4
INFO:forwarding.firewall:18 2 learned 10.43.21.3
```

The long numbers `360287970189639684` and `360287970189639681` represent the switches at Bristol with dpids 05:00:00:00:00:00:00:04and 05:00:00:00:00:00:00:01 respectively. The second number is the port of that switch. 18 is the dpid of the UCL GEPON. Having established that the VMs can ping each other then the next stage was to test the firewall's effectiveness. If correctly configured then traffic to and from port 80 should be accepted unless it was originated from or was going to 10.43.21.4.

The test of traffic to port 80 from 10.43.21.1 (Bristol) to 10.43.21.3 (UCL) was to use the nc command on 10.43.21.1 to send traffic on port 80 while, at the other end the nc command was used in listen mode to listen to port 80.

At the server 10.43.21.1 (Bristol) the nc command sent. The order of the test is as follows:

- Step 0:
  - Test ping from 10.43.21.1 to 10.43.21.3 to establish connectivity.

- Step 1:
  - On 10.43.21.3 use nc to listen on port 80.

- Step 2:

- On 10.43.21.1 use nc to connect to port 80 on 10.43.21.3 and type a message which should be received.

```
richardclegg@tdur1:~$ ping 10.43.21.3
PING 10.43.21.3 (10.43.21.3) 56(84) bytes of data.
64 bytes from 10.43.21.3: icmp_req=1 ttl=64 time=339 ms
64 bytes from 10.43.21.3: icmp_req=2 ttl=64 time=9.64 ms
^C
--- 10.43.21.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 9.644/174.323/339.002/164.679 ms
richardclegg@tdur1:~$ nc 10.43.21.3 80
hello
there
from 10.43.21.1
```

The test as seen from 10.43.21.3 is shown below:

```
richardclegg@ucltest:~/code$ sudo nc -l -p 80
hello
there
from 10.43.21.1
```

As can be seen, the message was successfully sent on port 80. The next test was to ensure that the same message fails if sent to or from 10.43.21.4.

```
richardclegg@testit:~$ ping 10.43.21.3
PING 10.43.21.3 (10.43.21.3) 56(84) bytes of data.
64 bytes from 10.43.21.3: icmp_req=1 ttl=64 time=339 ms
64 bytes from 10.43.21.3: icmp_req=2 ttl=64 time=9.64 ms
^C
--- 10.43.21.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 9.644/174.323/339.002/164.679 ms
richardclegg@testit:~$ nc 10.43.21.3 80
hello
there
from 10.43.21.4
```

When viewed from 10.43.21.3 then no traffic was seen by nc. The POX firewall controller also reported that a drop rule had been installed.

```
richardclegg@ucltest:~/code$ sudo nc -l -p 80
```

This shows convincingly that the switch can provide connectivity as part of the test bed and can match against rules and act upon those matches to forbid or allow traffic.

2.5.3 Conclusions

The tests in this chapter have shown that the GEPON system can be presented to the OFELIA framework as if it were a single OpenFlow switch with one port for each ONU and one for the OLT. As it currently stands the framework has a lot of software path components that have performance implications, a software OpenFlow switch outside the OLT and a VPN providing connectivity to the network. The tests used here have demonstrated that the framework can handle the following functionalities:

- PacketIn
- PacketOut (used for ARP replies)
- Matching on ports and IP addresses
- Insertion of FlowMod rules to provide routing.

2.6 Integration of Cavium

The xDPd/ROFL Cavium first implementation integrated the ALIEN HAL design to the Split Data Plane Dell research platform. But as the SDP has a single 10G internal interface, this limited the possible testing scenario and it was decided to integrate a simple scenario on a single flow. In this integration phase for simplicity reasons it was decided to connect the Split Data Plane with the ALIEN HAL / xDPd to OFELIA without creating a new island with its full control framework, and to use the EHU Island instead.

2.6.1 Description of the integration

The minimum requirement to interact with OFELIA EHU Island was to use a Layer2 OpenVPN TAP directly with EHU and the Force test bed to create the data plane pipe and an OFELIA OpenVPN to iMinds as an OFELIA experimenter for the Control plane pipe.

The test bed equipment is listed below:

- Dell Server PowerEdge C6100 Server: running OF controllers and admin tools
- PowerConnect 7024 + SDP module
- IXIA 400T 1 Port 1Gbps: generated traffic to EHU
- NEC OpenFlow Switch on EHU side
- 1 Virtual Machine on EHU side

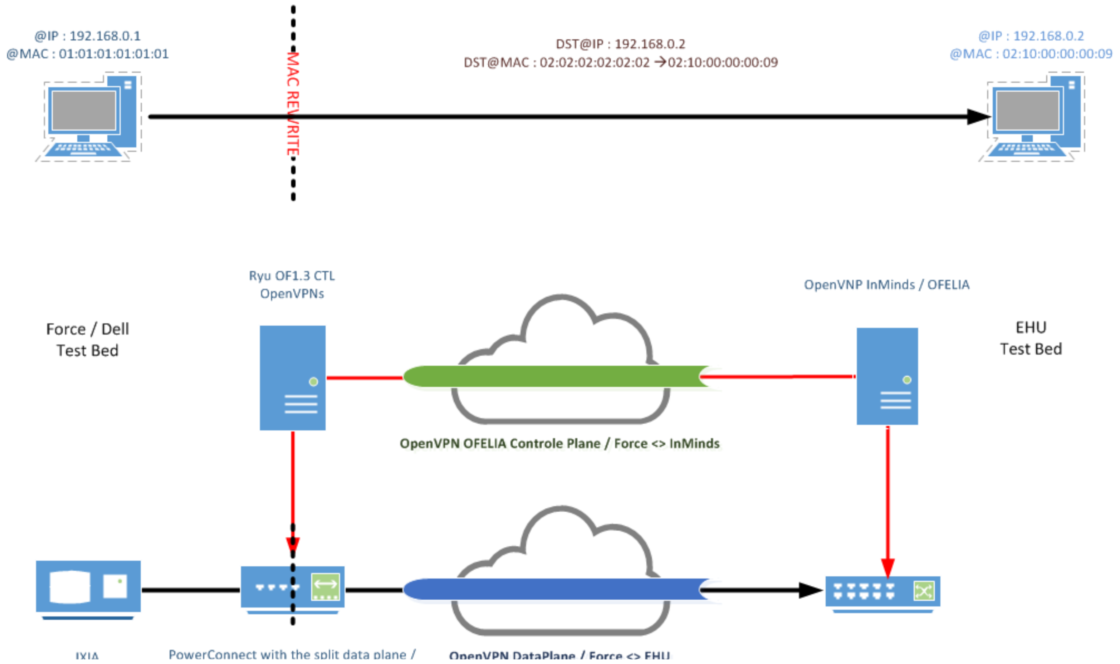


Figure 2-31 SDP Test bed OFELIA interconnection EHU Island diagram

The Figure 2-31 above gives a description of the topology used to interconnect the SDP test bed setup and the EHU Island. The IXIA is replacing a host on the SDP side and allows emulating any type of traffic.

The IXIA is sending Ethernet / IP / UDP packets to the EHU VM Host through the data plane pipe.

- MAC: SRC 01:01:01:01:01:01 / DST 02:02:02:02:02:02
- IP : SRC 192.168.0.1 / DST 192.168.0.2
- UDP

The PowerConnect is not controlled through OpenFlow and behaves as a normal learning switch, which needed to configure MAC Access-List redirection to force the frames to reach the Octeon Split Data Plane module via the internal XAUI interface.

Here is the configuration of the MAC ACL redirect:

```
mac access-list extended ToOcteon
permit any any redirect Te1/1/2
exit
mac access-list extended To1
permit  0101.0101.0101  FFFF.FFFF.FFFF  0210.0000.0009  FFFF.FFFF.FFFF  redirect
   Gi1/0/2
```

About the Split Data Plane configuration The Octeon CN5260 has four MIPS cores and it was configured to have Linux running xDPd with the configuration below on 1 core and the 3 remaining cores running as SA mode ODPD.

xDPd config file:

```
config:{

openflow:{
logical-switches:{
                        #Name of the switch dp0
dp0:{
                                #Most complex configuration
dpid = "0x100"; #Must be hexadecimal
version = 1.3;
description="This is a switch";

                                #Controller connection(s)
controller-connections:{
main:{
remote-hostname="192.168.1.67";
remote-port = 6653;
                                        };
                                };

                                #Reconnect behaviour
reconnect-time=1; #seconds

                                #Packet-in rate limiting
pirl-enabled=TRUE;
pirl-rate=400; #MAX PKT_IN events/s

                                #Tables and MA
num-of-tables=1;

#Physical ports attached to this logical switch. This is mandatory
ports = ("xaui0");
                        };
                };
        };
};
```

The EHU NEC switch needs to be configured with the dynamically allocated IP address from the OFELIA OpenVPN.

Control configuration

RYU controller was configured to run OFCTL_REST inside the Force test bed and the simple curl command line was used to configure new rules on the SDP / xDPd.

RYU also configured the EHU / NEC switch with a simple rule to allow the manipulated SDP's traffic to reach the EHU Virtual Machine.

The EHU NEC Switch was configured with the OFELIA IP dynamically allocated when connecting the OFELIA OpenVPN. It is not the perfect setup for a permanent testing environment; further improvements are possible and could be considered in the future.

Here is a sample of the CURL REST command line used:

```
curl    -d    '{"dpid":    "256","priority":"24001","duration_sec":    0,"match":
   {"in_port":"1"},"actions":[{"type":"SET_FIELD","field":"eth_dst","value":"00
   :04:00:00:00:01"},{"type":"OUTPUT","port":"4294967288"}]}'
   http://127.0.0.1:8080/stats/flowentry/add
```

2.6.2 Functional testing

For the functional testing the major openflow controllers were tried: Ryu, OpenDaylight and POX and OFTest with OF version 1.3.

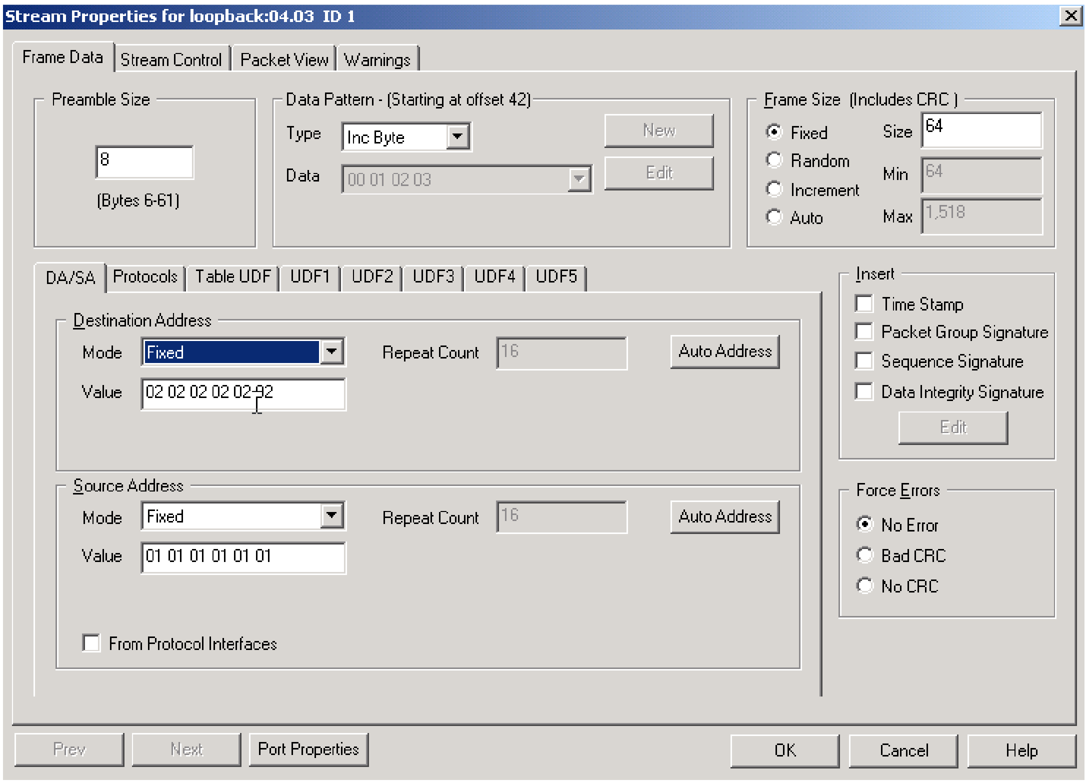
1) OFELIA Resources involved in the testing

As the OFELIA resources involved in this Cavium testing are the same as the ones explained in the previous section 2.1.1, the reader is referred to it.

2) Testing workflow and results

The workflow was minimized to a single step for testing integration, which consisted in the IXIA equipment generating a flow to MAC destination (02:02:02:02:02) and the SDP changing it to another one (02:10:00:00:00:09).

The screenshots below were taken using layer 2 access-list redirect to check the modified packets. Those screenshots present the IXIA generating and capturing input and output of the Split data plane, sent to the EHU virtual machine through the openvpn and the EHU NEC openflow switch.



Stream Properties for loopback:04.03 ID 1

Frame Data | Stream Control | Packet View | Warnings

Preamble Size: 8 (Bytes 6-61)

Data Pattern - (Starting at offset 42)
Type: Inc Byte
Data: 00 01 02 03

Frame Size (Includes CRC)
● Fixed ○ Random ○ Increment ○ Auto
Size: 64
Min: 64
Max: 1,518

DA/SA | Protocols | Table UDF | UDF1 | UDF2 | UDF3 | UDF4 | UDF5

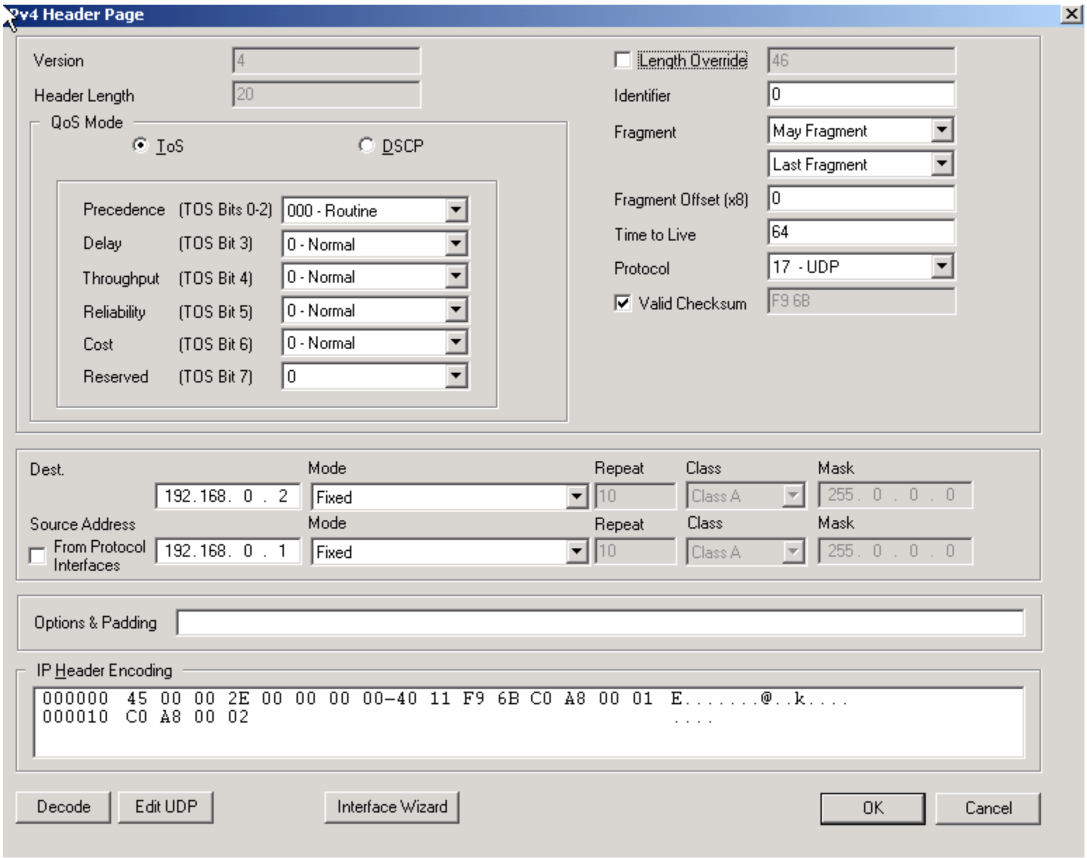
Destination Address
Mode: Fixed
Value: 02 02 02 02 02 02
Repeat Count: 16

Source Address
Mode: Fixed
Value: 01 01 01 01 01 01
Repeat Count: 16
□ From Protocol Interfaces

Insert
□ Time Stamp
□ Packet Group Signature
□ Sequence Signature
□ Data Integrity Signature

Force Errors
● No Error
○ Bad CRC
○ No CRC

Figure 2-32 IXIA screenshot stream generated properties (1Gbps line-rate traffic)



IPv4 Header Page

Version: 4

Header Length: 20

QoS Mode: ● ToS ○ DSCP

Precedence (TOS Bits 0-2): 000 - Routine

Delay (TOS Bit 3): 0 - Normal

Throughput (TOS Bit 4): 0 - Normal

Reliability (TOS Bit 5): 0 - Normal

Cost (TOS Bit 6): 0 - Normal

Reserved (TOS Bit 7): 0

□ Length Override: 46

Identifier: 0

Fragment: May Fragment / Last Fragment

Fragment Offset (x8): 0

Time to Live: 64

Protocol: 17 - UDP

☑ Valid Checksum: F9 6B

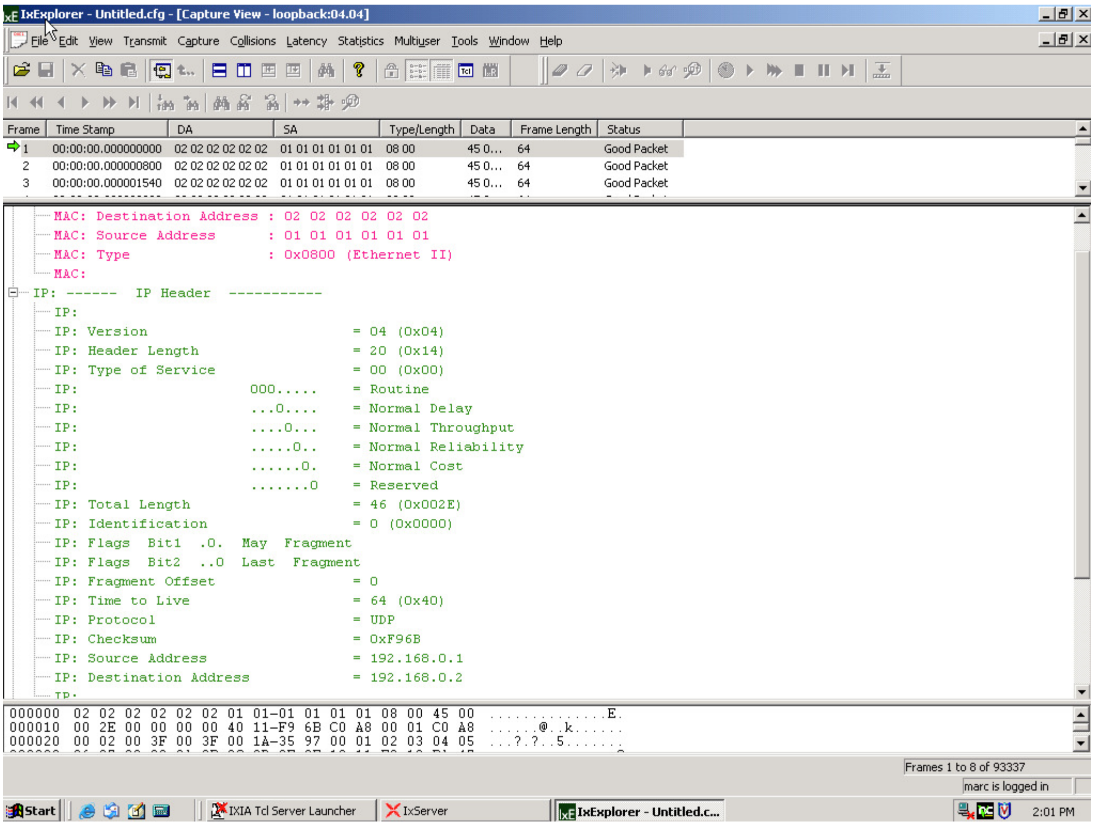
	Address	Mode	Repeat	Class	Mask
Dest.	192.168. 0 . 2	Fixed	10	Class A	255 . 0 . 0 . 0
Source Address (□ From Protocol Interfaces)	192.168. 0 . 1	Fixed	10	Class A	255 . 0 . 0 . 0

Options & Padding:

IP Header Encoding

```
000000  45 00 00 2E 00 00 00 00-40 11 F9 6B C0 A8 00 01  E.......@..k....
000010  C0 A8 00 02                                       ....
```

Figure 2-33 IXIA screenshot packet header configuration window



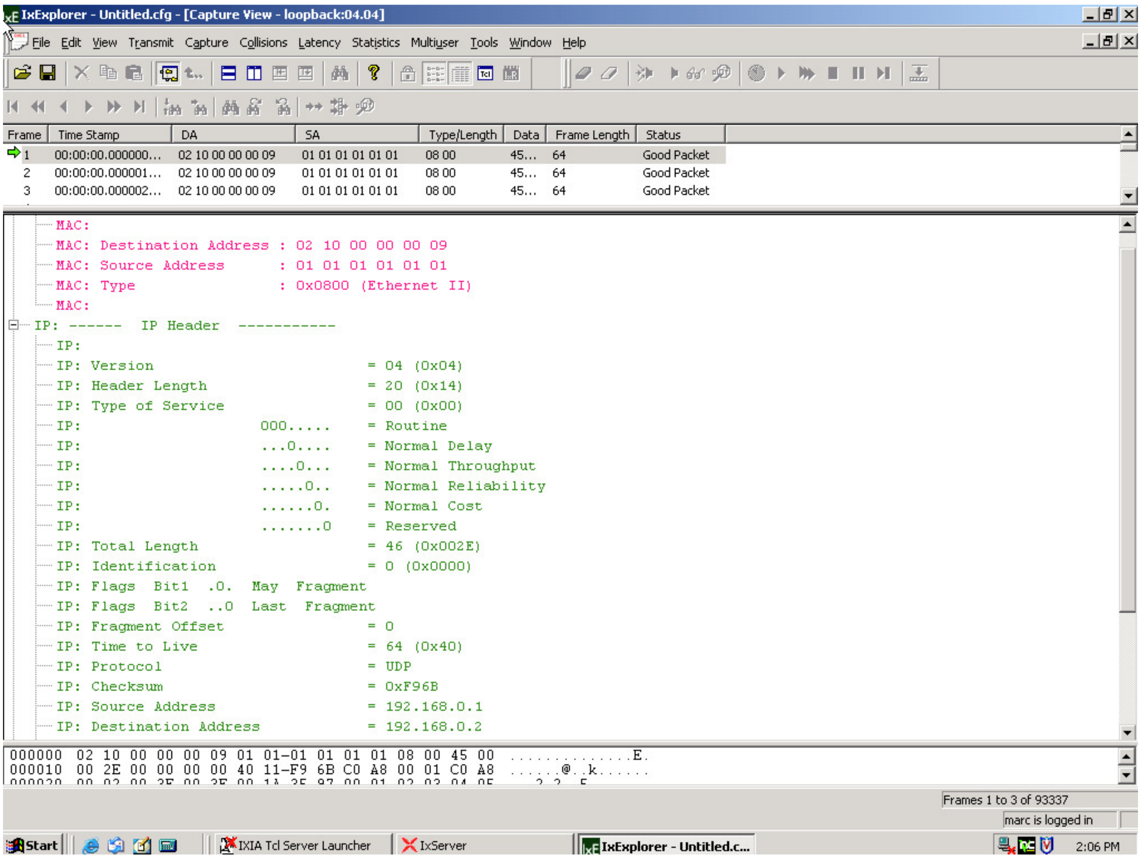
IxExplorer - Untitled.cfg - [Capture View - loopback:04.04]
File Edit View Transmit Capture Collisions Latency Statistics Multiuser Tools Window Help

Frame	Time Stamp	DA	SA	Type/Length	Data	Frame Length	Status
1	00:00:00.000000000	02 02 02 02 02 02	01 01 01 01 01 01	08 00	45 0...	64	Good Packet
2	00:00:00.000000800	02 02 02 02 02 02	01 01 01 01 01 01	08 00	45 0...	64	Good Packet
3	00:00:00.000001540	02 02 02 02 02 02	01 01 01 01 01 01	08 00	45 0...	64	Good Packet

```
  MAC: Destination Address : 02 02 02 02 02 02
  MAC: Source Address      : 01 01 01 01 01 01
  MAC: Type                : 0x0800 (Ethernet II)
  MAC:
IP: ------  IP Header  -----------
  IP:
  IP: Version                          = 04 (0x04)
  IP: Header Length                    = 20 (0x14)
  IP: Type of Service                  = 00 (0x00)
  IP:                       000.....   = Routine
  IP:                       ...0....   = Normal Delay
  IP:                       ....0...   = Normal Throughput
  IP:                       .....0..   = Normal Reliability
  IP:                       ......0.   = Normal Cost
  IP:                       .......0   = Reserved
  IP: Total Length                     = 46 (0x002E)
  IP: Identification                   = 0 (0x0000)
  IP: Flags  Bit1  .0.  May  Fragment
  IP: Flags  Bit2  ..0  Last  Fragment
  IP: Fragment Offset                  = 0
  IP: Time to Live                     = 64 (0x40)
  IP: Protocol                         = UDP
  IP: Checksum                         = 0xF96B
  IP: Source Address                   = 192.168.0.1
  IP: Destination Address              = 192.168.0.2
  IP:
```

```
000000  02 02 02 02 02 02 01 01-01 01 01 01 08 00 45 00  ..............E.
000010  00 2E 00 00 00 00 40 11-F9 6B C0 A8 00 01 C0 A8  ......@..k......
000020  00 02 00 3F 00 3F 00 1A-35 97 00 01 02 03 04 05  ...?.?..5.......
```

Figure 2-34 IXIA screenshot packet captured without MAC rewrite rule active



IxExplorer - Untitled.cfg - [Capture View - loopback:04.04]

File Edit View Transmit Capture Collisions Latency Statistics Multiuser Tools Window Help

Frame	Time Stamp	DA	SA	Type/Length	Data	Frame Length	Status
1	00:00:00.000000...	02 10 00 00 00 09	01 01 01 01 01 01	08 00	45...	64	Good Packet
2	00:00:00.000001...	02 10 00 00 00 09	01 01 01 01 01 01	08 00	45...	64	Good Packet
3	00:00:00.000002...	02 10 00 00 00 09	01 01 01 01 01 01	08 00	45...	64	Good Packet

```
  MAC:
  MAC: Destination Address : 02 10 00 00 00 09
  MAC: Source Address      : 01 01 01 01 01 01
  MAC: Type                : 0x0800 (Ethernet II)
  MAC:
IP: ------  IP Header  -----------
  IP:
  IP: Version                          = 04 (0x04)
  IP: Header Length                    = 20 (0x14)
  IP: Type of Service                  = 00 (0x00)
  IP:                       000.....   = Routine
  IP:                       ...0....   = Normal Delay
  IP:                       ....0...   = Normal Throughput
  IP:                       .....0..   = Normal Reliability
  IP:                       ......0.   = Normal Cost
  IP:                       .......0   = Reserved
  IP: Total Length                     = 46 (0x002E)
  IP: Identification                   = 0 (0x0000)
  IP: Flags  Bit1  .0.  May  Fragment
  IP: Flags  Bit2  ..0  Last  Fragment
  IP: Fragment Offset                  = 0
  IP: Time to Live                     = 64 (0x40)
  IP: Protocol                         = UDP
  IP: Checksum                         = 0xF96B
  IP: Source Address                   = 192.168.0.1
  IP: Destination Address              = 192.168.0.2
```

```
000000  02 10 00 00 00 09 01 01-01 01 01 01 08 00 45 00  ..............E.
000010  00 2E 00 00 00 00 40 11-F9 6B C0 A8 00 01 C0 A8  ......@..k......
```

Frames 1 to 3 of 93337

marc is logged in

Figure 2-35 IXIA screenshot packet captured with MAC rewrite rule active

In addition of interconnecting to EHU Island, simple testing was conducted using 1Gbps IXIA port sending line rate traffic for rewriting through the SDP/Octeon. The Split Data Plane module was able to maintain this throughput for hours.

On the xDPd configuration file above the packet rate input was fixed to 150 kpps. The IXIA was injecting 1Gbps line rate traffic of 64Byte UDP packets at 1.488.096 pps. The xDPd with a single rewriting MAC destination address openflow rule was able to keep up with a packet rate of 1.406.00 pps which is a bit less than the line rate throughput.

A bash script was used to generate REST FlowMod adding, matching random source mac address rules. It could be observed the packet rate decreasing rapidly with the number of rules. With 10 active rules matching random mac addresses the rate went to 741.000pps, which is half of the maximum throughput. The actual matching algorithm used in xDPd was doing a one by one match check and this was the expected performance with this type of algorithm.

See below a log scale graph.

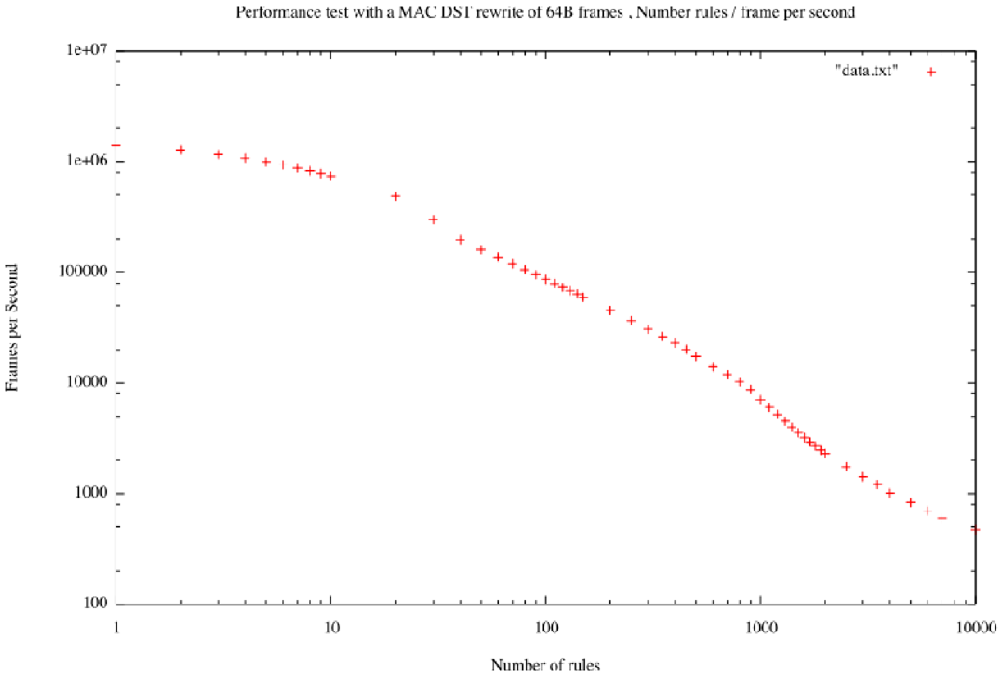


Figure 2-36 scale graph of packet per second throughput and number of active rules

2.6.3 Conclusions

The actions tested in this experimentation are listed below:

- OpenFLow controller test with the SDP / xDPd:
  - Opendaylight Hydrogen version with OpenFlow 1.0 and 1.3
  - Ryu version 3.9 with OpenFlow 1.0 and 1.3
  - POX version 0.2.0 with OpenFlow 1.0

- Flow entry match to :
  - IP protocol (recognize TCP, UPD and ICMP)
  - Source TCP port
  - Destination TCP port
  - Source UDP
  - Destination UDP port

- Supported actions:
  - Forward back port_in
  - Destination or Source MAC address rewrite
  - Dropping

- Flow entry add
- Flow entry remove
- Packet-in
- Packet-out

It was not possible to test VLAN tagging due to the limitation of the Split Data Plane architecture as all the packets needed to be redirected by the Host Switch BCM chipset who was dropping or striping out the VLAN tags.

The ALIEN HAL xDPd implementation testing helped to remove bugs and get a stable version of the Cavium development at least for simple rules packets manipulation with OF 1.0 to 1.3. As an example, the list bellow presents the main bugs fixed:

- It was detected that the SDP / xDPd was not recognized by OpenDaylight, due to a badly formed description field.
- Also, it was noticed that full line rate traffic hardware generated by the IXIA was crashing xDPd.
- The first linux kernel was compiled with unnecessary features and in some condition was crashing. A new kernel lighter and stable has been compiled.
- Finally, it was observed that the openflow get stats dumping the content of a flow table larger than 500 active rules, took too much time and reached the controller to timeout.

3 CONET experiment

3.1 Description of experiment

The main goal of the CONET experiment is to test and demonstrate the ALIEN HAL concept by testing and validating the developments done in other work packages concerning the hardware agnostic and hardware specific parts of the HAL for non-OpenFlow capable devices.

The experiment combines CCN with OpenFlow using the existing work from Content NETwork (CONET) project [CONET] as described in [D5.1]. To run the CONET experiment and test at European level, the OFELIA experimental facility (FIRE facility) is used and both computational resources and OpenFlow devices are provisioned on it. An OFELIA slice, the ALIEN slice (VLAN 700), is created (see Figure 3-1 CONET experiment deployment at OFELIA: resources and topology) involving resources from several OFELIA islands: EHU, i2Cat, iMinds, UNIVBRIS and PSNC. The ALIEN slice includes computational resources (e.g. end-nodes, CONET nodes and OpenFlow controller), ALIEN hardware platforms (i.e. DOCSIS ALIEN) and OpenFlow devices from the pool of resources provided by OFELIA.

The CONET experiment allows validating the HAL-enabled devices, like the DOCSIS ALIEN device, by using a third party application (i.e. CCn) adapted to the OpenFlow environment, i.e. CONET (an application that generates both data plane and control plane traffic). As a result, the DOCSIS ALIEN device included in the tests is validated. The OpenFlow requirements imposed by CONET to any ALIEN device included in this scenario are listed in [D5.1].

The CONET scenario includes a content client, content server and a cache server. These CCN end nodes are deployed in different OFELIA islands to demonstrate the proper integration of ALIEN islands (currently part of OFELIA) in the experiment.

3.2 OFELIA resources involved

The CONET experiment involves resources from several OFELIA islands. Some of these islands were already OFELIA islands (from the original FP7 project), such as i2Cat, iMinds and UNIVBRIS islands, and some others have become part of OFELIA as a result of the ALIEN project, such as EHU, PSNC and PUT Islands. Therefore, the experiment spreads across five OFELIA islands.

The ALIEN slice has been created to deploy the scenario for CONET testing and one of the tricky parts of defining an inter-island slice is to select a (free) VLAN identifier that was not assigned before. The VLAN 700 was finally assigned to the ALIEN slice. After manual approval of the corresponding flowspace at each OFELIA island (involving people not included in the ALIEN project) the scenario was ready for testing.

Figure 3-1 shows the details of the ALIEN slice, including the deployed resources and topology of CONET scenario. The topology is relevant to properly define the inter-island connection and the flowspaces (including DPIDs, physical ports and VLAN identifier, i.e. VLAN 700) to be requested to each island through the OCF. The resources from each OFELIA island are listed below.

- EHU Island:
    - 1 DOCSIS ALIEN device exposed as OF1.0 switch:
        - DPID 10:00:00:00:00:00:00:02,VLAN 700, ports 12 and 21.
    - 1 NEC IP8800 OF1.0 switch:
        - DPID 10:00:00:00:00:00:00:01, VLAN 700, ports 5, 6 and 19.
            - Port 19 is connected to i2Cat Island.
    - 2VMs deployed from 2 different computation resources:
        - 2 CONET content-clients to request CCN content.

- i2Cat Island:
    - 4 OF switches:
        - DPID 00:10:00:00:00:00:00:01, VLAN 700, ports 2, 3 and 11.
            - Port 11 is connected to iMinds Island.
        - DPID 00:10:00:00:00:00:00:02, VLAN 700, ports 1, 4, 12 and 13.
            - Port 13 is connected to EHU Island.
        - DPID 00:10:00:00:00:00:00:03, VLAN 700, ports 1 and 9.
            - Port 9 is connected to UNIVBRIS Island.
        - DPID 00:10:00:00:00:00:00:04, VLAN 700, ports 2 and 12.
    - 2 VMs deployed from 2 different computation resources:
        - 1 CONET cache server to cache CCN content.
        - 1 CONET server 1 to provide CCN content.
    - 1 VM OpenFlow CONET controller (10.216.12.121).

- iMinds Island:
    - 1 OF switches:
        - DPID 01:00:00:00:00:00:00:FF, VLAN 700, ports 3 and 5.
            - Port 3 is connected to i2Cat Island.
            - Port 5 is connected to UNIVBRIS Island.

- UNIVBRIS Island:
    - 3 OF switches:
        - DPID 05:00:00:00:00:00:00:02, VLAN 700, ports 5, 9 and 17.
            - Port 9 is connected to PSNC Island.
        - DPID 05:00:00:00:00:00:00:03, VLAN 700, ports 7 and 16.
            - Port 7 is connected to i2Cat Island.
        - DPID 05:00:00:00:00:00:00:04, VLAN 700, ports 6 and 9.
            - Port 9 is connected to iMinds Island.

- PSNC Island:
    - 1 OF switches:

  - DPID 11:00:00:00:00:00:00:01, VLAN 700, ports 11 and 17.
    - Port 11 is connected to UNIVBRIS Island.
- 1 VMs deployed from computation resources:
  - 1 CONET server 2 to provide CCN content.

As previously listed in the i2Cat Island, the CONET controller (10.216.12.121) is defined as the OpenFlow controller for the ALIEN slice (VLAN 700).

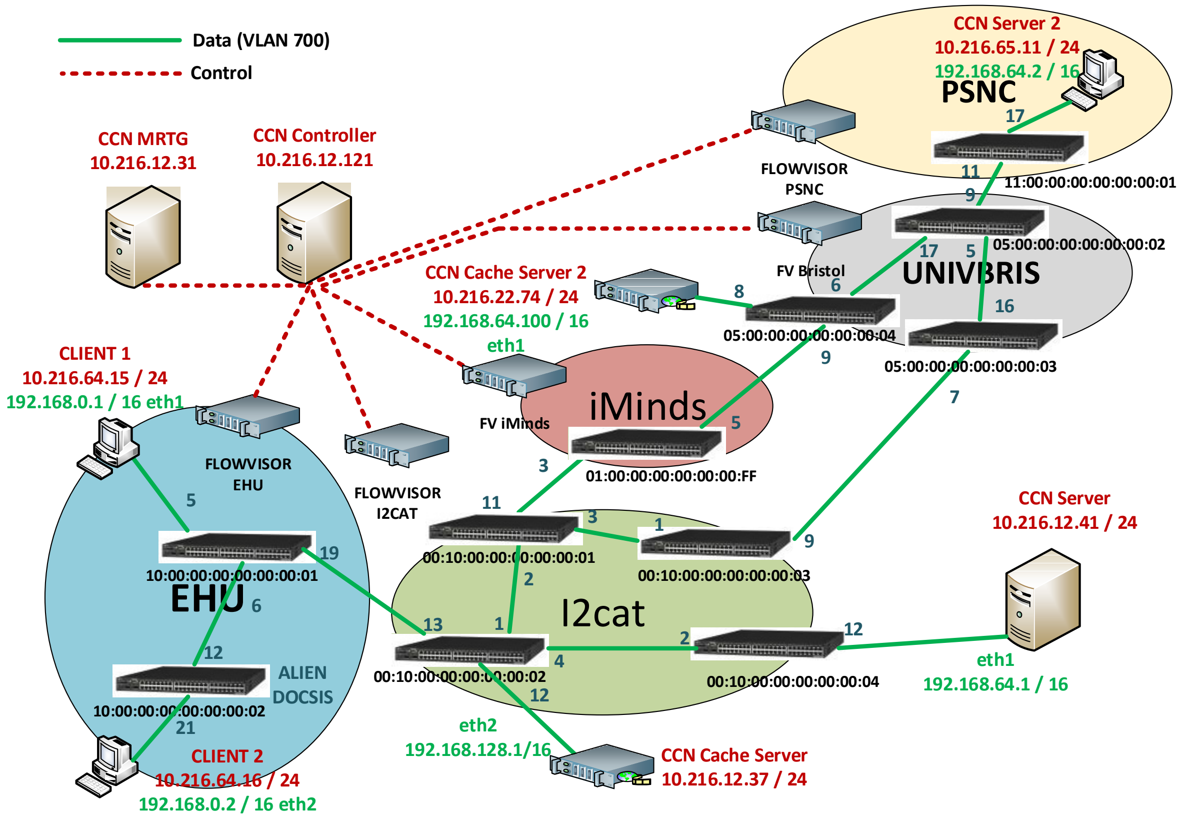


Figure 3-1 CONET experiment deployment at OFELIA: resources and topology

3.3 Workflow of CONET tests

The CONET experiment is a third party application that is used to validate the integration of the ALIEN islands in OFELIA and the implementation of the HAL for ALIEN devices (i.e. DOCSIS ALIEN). In order to explain what has been tested, the CONET experiment is represented as a couple of workflows with several steps defined to better understand the tests. The first workflow describes the general workflow of any CONET deployment, focusing on the control plane (i.e. OpenFlow messages involved). Once the CONET control workflow is clear, the second workflow describes the test performed to better understand the results and graphics presented in the next section 3.4 Results from CONET tests.

The following workflow describes the CONET application and the steps involved at control plane when a piece of content is requested from a client to a server and it is also cached by the cache server:

- Step 0 (CONET controller application starts)
  - The CONET controller deletes all the flow entries of all the DPIDs.

- Step 1 (CONET client requests some content to the CONET server: ARP request -> packet-in)
  - The CONET client is properly configured with information about content and CONET servers.
  - The CONET client requests a specific content. The IP address of the CONET server with this content is obtained.
  - The CONET client sends and ARP to obtain the MAC address associated with the server.
  - The ARP request packet generates a packet-in message (OF) to the CONET controller.

- Step 2 (CONET cache server configuration ->flowmod)
  - The CONET traffic (i.e. UDP packets) from the CONET cache server to the CONET client is configured by flowmod messages.
  - The CONET traffic from the CONET server to the CONET client is configured by flowmod messages installed on the DPIDs with a CONET cache server associated to them (in the path from the server to the client). The flow entry duplicates all the packets from the server to the client and rewrites the destination MAC address of the duplicated packet with the MAC address of the CONET cache server.

- Step 3 (ARP reply ->flowmod)
  - The CONET controller enables the ARP response packets from the CONET server to the CONET client by installing a flowmod message.

- Step 4 (IP traffic between the CONET server and CONET client is enabled)
  - The CONET client starts sending content requests to the CONET server.
  - The first content request generates a packet-in message (OF) to the CONET controller.
  - The CONET controller sends a flowmod message to enable the IP traffic (MAC addresses and EtherType) from the CONET client to the CONET server.
  - The CONET server starts sending content chunks to the CONET client.
  - The first content chunk generates a packet-in message (OF) to the CONET controller.
  - The CONET controller sends a flowmod message to enable the IP traffic (MAC addresses and EtherType) from the CONET server to the CONET client.

- Step 5 (CONET cache server receives a chunk)
  - While the packets from a content chunk pass through the DPID with a cache server, the chunk is cached.
  - Once the complete chunk is cached, the CONET cache server sends a message to the CONET controller to notify which chunk has been cached.

- Step 6 (CONET controller installs a flow entry for the cached chunk ->flowmod)
  - Each time a chunk is cached at the CONET cache server, a new flow entry is installed in the DPID associated to that cache server to redirect all the client requests for the same chunk.
  - The CONET controller sends a flowmod message to redirect the UDP packets from clients to servers with the UDP ports associated to the cached chunk (the identifier of each chunk is coded as UDP ports).

- Step 7 (Cached content is provided from the CONET cache server)
  - Each time a CONET client request a content (i.e. chunks) that has been already cached, the cached chunks are provided from the CONET cache server. In fact, the chunk request from client to server is redirected to the cache server.

The following workflow describes the actual test performed to validate and demonstrate the CONET scenario. This workflow presents a high level view to better explain the output results, which focuses on the data plane, since the graphics are related to the actual traffic (i.e. data plane) transmitted from the server (or cache server if the content has been already cached) to the client.

The test performed distinguishes two phases. In the first phase, the client requests a content, which is provided by the server. While the content is going from the server to the client, the cache server is caching the same content. In the second phase, the client requests the same content again, and this time, the cache server provides the content.

- Step 0 (CONET client and server are configured)
  - The CONET client and the CONET server must be properly configured.
  - The CONET client must configure the server associated with each piece of content.

- Step 1 (CONET client requests some content to the CONET server)
  - The CONET client requests a specific piece of content (i.e. executes a command at the terminal).
  - The client's configuration resolves which CONET server is associated to that content.
  - The CONET client sends the request to the appropriate server. Each content chunk is requested independently and generates a different request packet.

- Step 2 (CONET server provides the content to the CONET client)
  - The CONET server receives the client's requests (one per chunk) and sends the content chunks back to the CONET client.
  - The whole content is sent in chunks from the server to the client.

- Step 3 (CONET cache server caches the content)
  - While the content chunks pass through the DPID with a CONET cache server associated, the content chunks are duplicated and the copy is sent to the cache server.
  - Actually, step 2 and step 3 are simultaneous steps.

- Step 4 (CONET client requests the same content again)
  - Once the complete content is received at the client, the CONET client requests the same content again (i.e. executes the same command at the terminal).
  - Based on the client's configuration, the CONET server is resolved.
  - The CONET client sends the request to the associated server.
  - The requests from the client (one per chunk) reach the first CONET cache server in the path to the server and do not progress further (i.e. the server is not aware of any new request).

- Step 5 (CONET cache server provides de content to the CONET client)
  - The CONET cache server receives the client's requests and sends the content chunks back to the CONET client.
  - The chunks already cached are sent from the cache server to the client.
  - If a certain chunk has not been already cached, the client's request progresses until the server and the server provides the chunk.

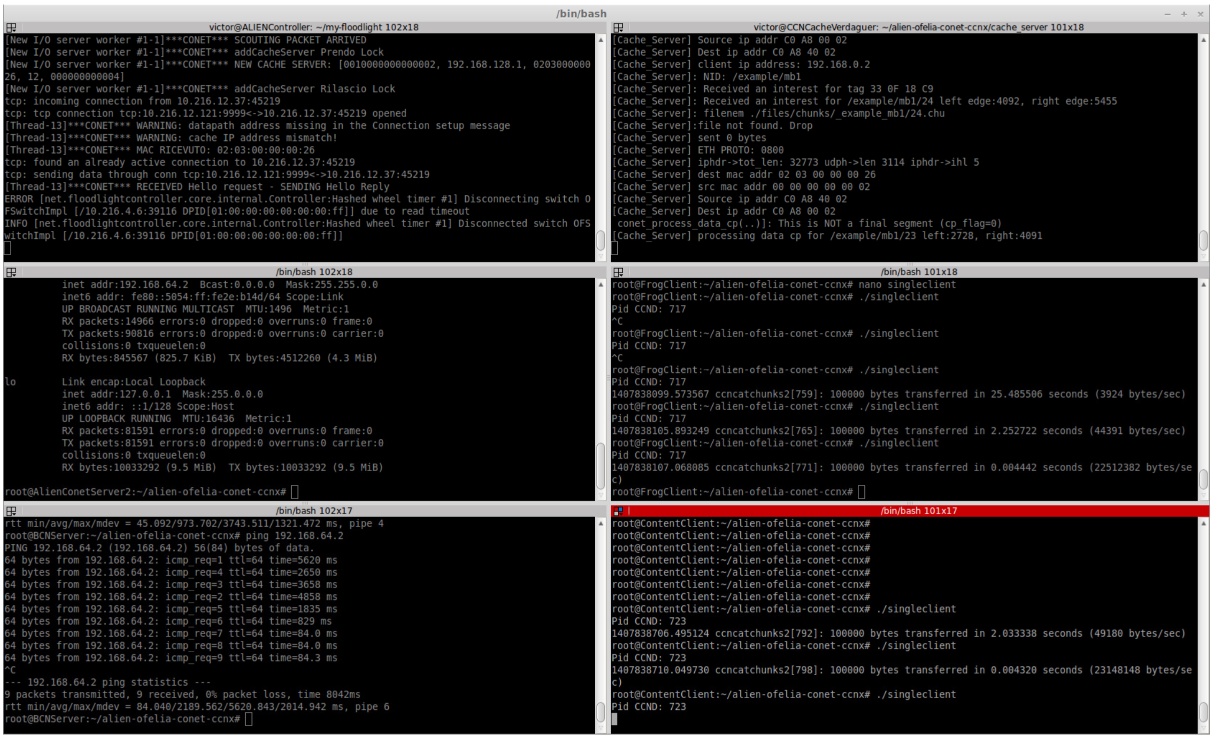
The next section shows the results (i.e. screenshots, graphics and output from terminals) obtained from performing the test described in the second workflow.

3.4 Results from CONET tests

This section presents the results from the test performed with the CONET experiment. The test workflow is described in the previous section 3.3. In a nutshell, a client behind the DOCSIS ALIEN in the EHU Island requests a content (100 Kbyte file) to a server located in the PSNC Island. The content is provided by the server at the PSNC Island, while a cache server located in the i2Cat Island caches this content. The same client in the EHU Island requests the same content, and this time, the content is provided by the cache server in the i2Cat Island.

The following Figure 3-2 shows the set of terminals needed to launch the CONET test, once all the resources are properly deployed and configured. All the terminals are explained below:

- On the top-left corner, the CONET controller is executed in the i2Cat Island (10.216.12.121).
- On the top-right corner, the CONET cache server is executed in the i2Cat Island.
- The terminal on the left center is the CONET server located in the PSNC Island (192.168.64.2).
- The terminal on the right center is the CONET client 1 in the EHU Island connected to the NEC OpenFlow switch.
- On the bottom-left corner, the CONET server located in the i2Cat Island is sending a ping to the CONET server located in the PSNC Island (84.0 ms).
- On the bottom-right corner, the CONET client behind the DOCSIS ALIEN in the EHU Island executes the test. The same content (100 Kbyte file) is requested two times, the first one it lasts 2.033338 seconds (PSNC server to EHU client), whereas the second time it lasts 0.004320 seconds (i2Cat cache server to EHU client).


/bin/bash
victor@ALIENController: ~/my-floodlight 102x18
victor@CCNCacheVerdaguer: ~/alien-ofelia-conet-ccnx/cache_server 101x18
/bin/bash 102x18
/bin/bash 101x18
/bin/bash 102x17
/bin/bash 101x17

```
[New I/O server worker #1-1]***CONET*** SCOUTING PACKET ARRIVED
[New I/O server worker #1-1]***CONET*** addCacheServer Prendo Lock
[New I/O server worker #1-1]***CONET*** NEW CACHE SERVER: [0010000000000002, 192.168.128.1, 0203000000
26, 12, 000000000004]
[New I/O server worker #1-1]***CONET*** addCacheServer Rilascio Lock
tcp: incoming connection from 10.216.12.37:45219
tcp: tcp connection tcp:10.216.12.121:9999<->10.216.12.37:45219 opened
[Thread-13]***CONET*** WARNING: datapath address missing in the Connection setup message
[Thread-13]***CONET*** WARNING: cache IP address mismatch!
[Thread-13]***CONET*** MAC RICEVUTO: 02:03:00:00:00:26
tcp: found an already active connection to 10.216.12.37:45219
tcp: sending data through conn tcp:10.216.12.121:9999<->10.216.12.37:45219
[Thread-13]***CONET*** RECEIVED Hello request - SENDING Hello Reply
ERROR [net.floodlightcontroller.core.internal.Controller:Hashed wheel timer #1] Disconnecting switch O
FSwitchImpl [/10.216.4.6:39116 DPID[01:00:00:00:00:00:00:ff]] due to read timeout
INFO [net.floodlightcontroller.core.internal.Controller:Hashed wheel timer #1] Disconnected switch OFS
witchImpl [/10.216.4.6:39116 DPID[01:00:00:00:00:00:00:ff]]
```

```
[Cache_Server] Source ip addr C0 A8 00 02
[Cache_Server] Dest ip addr C0 A8 40 02
[Cache_Server] client ip address: 192.168.0.2
[Cache_Server]: NID: /example/mb1
[Cache_Server]: Received an interest for tag 33 0F 18 C9
[Cache_Server]: Received an interest for /example/mb1/24 left edge:4092, right edge:5455
[Cache_Server]: filenem ./files/chunks/_example_mb1/24.chu
[Cache_Server]:file not found. Drop
[Cache_Server] sent 0 bytes
[Cache_Server] ETH PROTO: 0800
[Cache_Server] iphdr->tot_len: 32773 udph->len 3114 iphdr->ihl 5
[Cache_Server] dest mac addr 02 03 00 00 00 26
[Cache_Server] src mac addr 00 00 00 00 00 02
[Cache_Server] Source ip addr C0 A8 40 02
[Cache_Server] Dest ip addr C0 A8 00 02
 conet_process_data_cp(..)]: This is NOT a final segment (cp_flag=0)
[Cache_Server] processing data cp for /example/mb1/23 left:2728, right:4091
```

```
          inet addr:192.168.64.2  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::5054:ff:fe2e:b14d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1496  Metric:1
          RX packets:14966 errors:0 dropped:0 overruns:0 frame:0
          TX packets:90816 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:845567 (825.7 KiB)  TX bytes:4512260 (4.3 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:81591 errors:0 dropped:0 overruns:0 frame:0
          TX packets:81591 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10033292 (9.5 MiB)  TX bytes:10033292 (9.5 MiB)

root@AlienConetServer2:~/alien-ofelia-conet-ccnx#
```

```
root@FrogClient:~/alien-ofelia-conet-ccnx# nano singleclient
root@FrogClient:~/alien-ofelia-conet-ccnx# ./singleclient
Pid CCND: 717
^C
root@FrogClient:~/alien-ofelia-conet-ccnx# ./singleclient
Pid CCND: 717
^C
root@FrogClient:~/alien-ofelia-conet-ccnx# ./singleclient
Pid CCND: 717
1407838099.573567 ccncatchunks2[759]: 100000 bytes transferred in 25.485506 seconds (3924 bytes/sec)
root@FrogClient:~/alien-ofelia-conet-ccnx# ./singleclient
Pid CCND: 717
1407838105.893249 ccncatchunks2[765]: 100000 bytes transferred in 2.252722 seconds (44391 bytes/sec)
root@FrogClient:~/alien-ofelia-conet-ccnx# ./singleclient
Pid CCND: 717
1407838107.068085 ccncatchunks2[771]: 100000 bytes transferred in 0.004442 seconds (22512382 bytes/se
c)
root@FrogClient:~/alien-ofelia-conet-ccnx#
```

```
rtt min/avg/max/mdev = 45.092/973.702/3743.511/1321.472 ms, pipe 4
root@BCNServer:~/alien-ofelia-conet-ccnx# ping 192.168.64.2
PING 192.168.64.2 (192.168.64.2) 56(84) bytes of data.
64 bytes from 192.168.64.2: icmp_req=1 ttl=64 time=5620 ms
64 bytes from 192.168.64.2: icmp_req=4 ttl=64 time=2650 ms
64 bytes from 192.168.64.2: icmp_req=3 ttl=64 time=3658 ms
64 bytes from 192.168.64.2: icmp_req=2 ttl=64 time=4858 ms
64 bytes from 192.168.64.2: icmp_req=5 ttl=64 time=1835 ms
64 bytes from 192.168.64.2: icmp_req=6 ttl=64 time=829 ms
64 bytes from 192.168.64.2: icmp_req=7 ttl=64 time=84.0 ms
64 bytes from 192.168.64.2: icmp_req=8 ttl=64 time=84.0 ms
64 bytes from 192.168.64.2: icmp_req=9 ttl=64 time=84.3 ms
^C
--- 192.168.64.2 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8042ms
rtt min/avg/max/mdev = 84.040/2189.562/5620.843/2014.942 ms, pipe 6
root@BCNServer:~/alien-ofelia-conet-ccnx#
```

```
root@ContentClient:~/alien-ofelia-conet-ccnx#
root@ContentClient:~/alien-ofelia-conet-ccnx#
root@ContentClient:~/alien-ofelia-conet-ccnx#
root@ContentClient:~/alien-ofelia-conet-ccnx#
root@ContentClient:~/alien-ofelia-conet-ccnx#
root@ContentClient:~/alien-ofelia-conet-ccnx#
root@ContentClient:~/alien-ofelia-conet-ccnx#
root@ContentClient:~/alien-ofelia-conet-ccnx# ./singleclient
Pid CCND: 723
1407838706.495124 ccncatchunks2[792]: 100000 bytes transferred in 2.033338 seconds (49180 bytes/sec)
root@ContentClient:~/alien-ofelia-conet-ccnx# ./singleclient
Pid CCND: 723
1407838710.049730 ccncatchunks2[798]: 100000 bytes transferred in 0.004320 seconds (23148148 bytes/se
c)
root@ContentClient:~/alien-ofelia-conet-ccnx# ./singleclient
Pid CCND: 723
```

Figure 3-2 Screenshot of the terminals needed to perform the CONET experiment test

The following graphics show the data plane traffic (incoming traffic in green and outgoing in blue) in the three most relevant elements in this test: the CONET client at the EHU Island, the CONET server at the PSNC Island, and the CONET cache server at the i2Cat Island.

The next Figure 3-3 shows the data plane traffic entering the CONET client located at the EHU Island behind the DOCSIS ALIEN. The same content (100 Kbyte file) is requested two times, shown in green in the graphic.

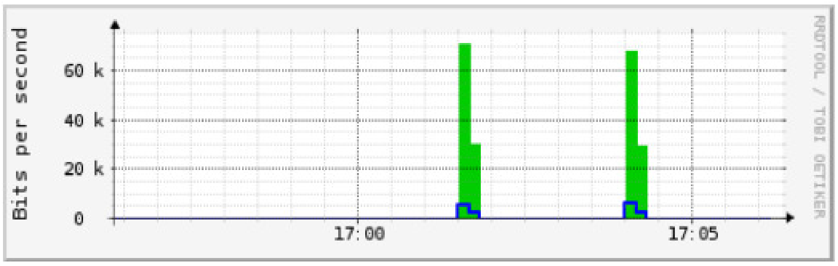


Figure 3-3 Data plane traffic of the CONET client at the EHU Island

The next Figure 3-4 shows the data plane traffic transmitted from the CONET server located at the PSNC Island. There is only one transmission of this file (100 Kbyte), shown in blue in the graphic.

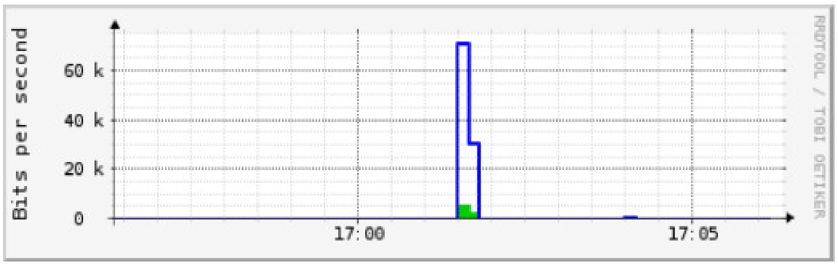


Figure 3-4 Data plane traffic of the CONET server at the PSNC Island

The next Figure 3-5 shows the data plane traffic to and from the CONET cache server located at the i2Cat Island. The first time the content is requested, the chunks enter the cache server (shown in green in the graphic) in order to be cached. The second time the same content is requested, the chunks are provided from the cache server to the CONET client (shown in blue in the graphic).

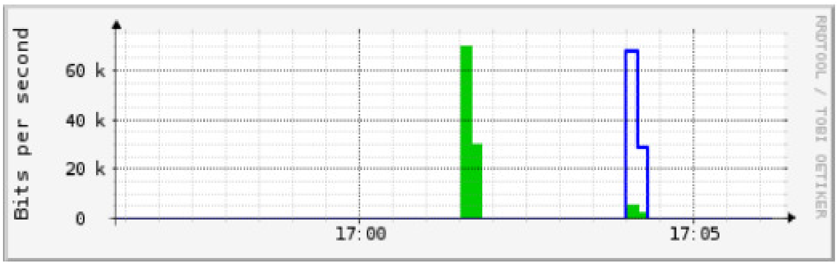
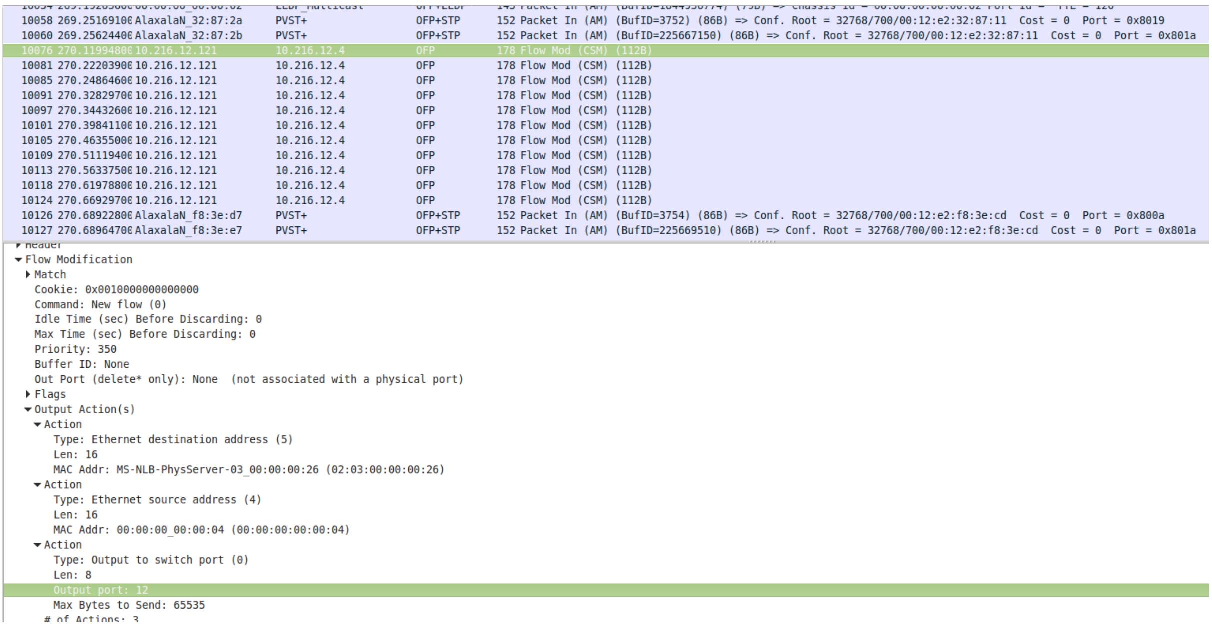


Figure 3-5 Data plane traffic of the CONET cache server at the i2Cat Island

The next figure shows a capture of the CONET control plane done with Wireshark. The OpenFlow messages have been analyzed to describe the CONET control plane workflow presented in the previous section 3.3. The ALIEN hardware must properly implement these OpenFlow messages in order to be part of the CONET scenario tested in this section.

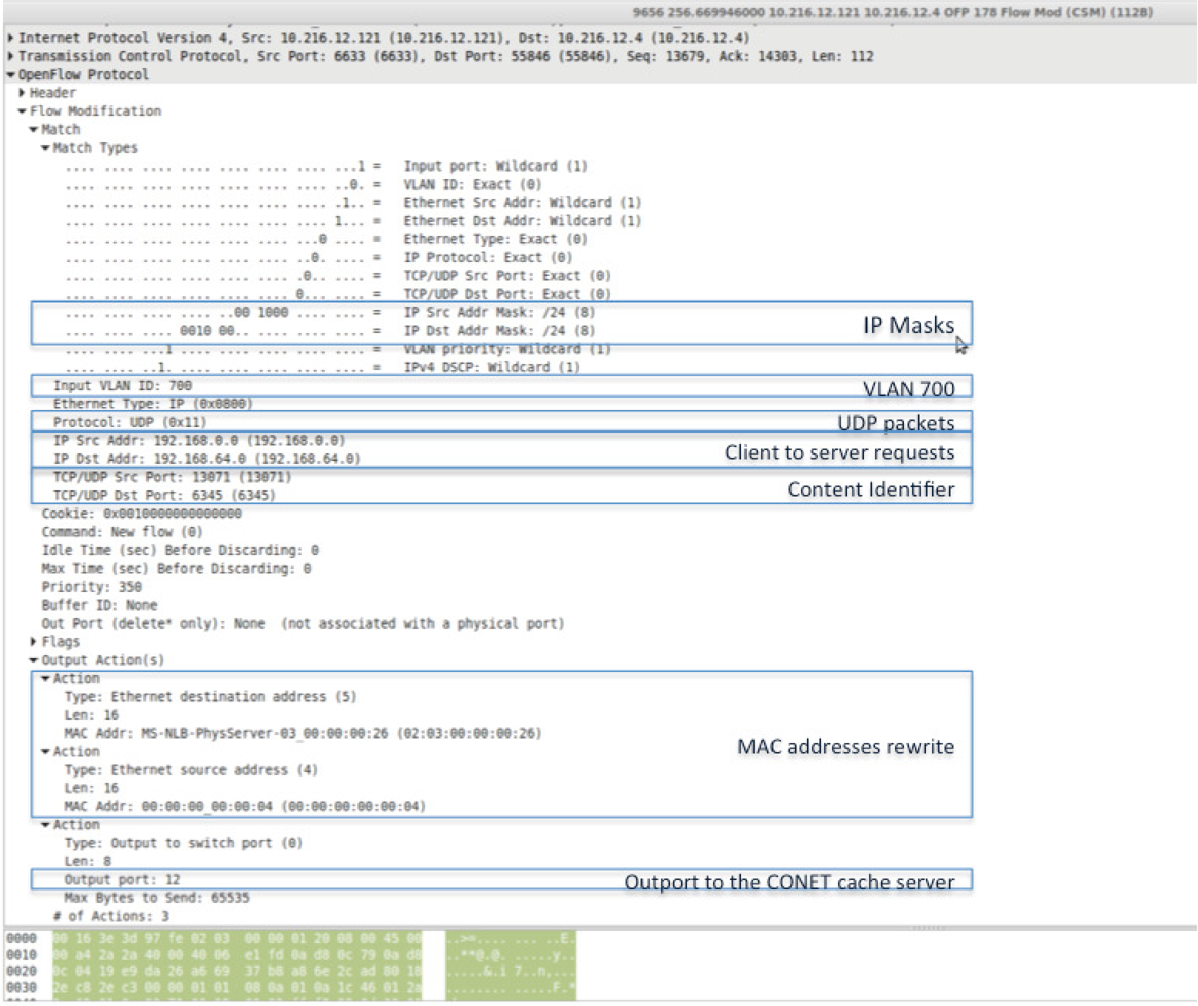


```
10058 269.2516910 AlaxalaN_32:87:2a   PVST+          OFP+STP   152 Packet In (AM) (BufID=3752) (86B) => Conf. Root = 32768/700/00:12:e2:32:87:11  Cost = 0  Port = 0x8019
10060 269.2562440 AlaxalaN_32:87:2b   PVST+          OFP+STP   152 Packet In (AM) (BufID=225667150) (86B) => Conf. Root = 32768/700/00:12:e2:32:87:11  Cost = 0  Port = 0x801a
10076 270.1199480 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10081 270.2220390 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10085 270.2486460 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10091 270.3282970 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10097 270.3443260 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10101 270.3984110 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10105 270.4635500 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10109 270.5111940 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10113 270.5633750 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10118 270.6197880 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10124 270.6692970 10.216.12.121       10.216.12.4    OFP       178 Flow Mod (CSM) (112B)
10126 270.6892280 AlaxalaN_f8:3e:d7   PVST+          OFP+STP   152 Packet In (AM) (BufID=3754) (86B) => Conf. Root = 32768/700/00:12:e2:f8:3e:cd  Cost = 0  Port = 0x800a
10127 270.6896470 AlaxalaN_f8:3e:e7   PVST+          OFP+STP   152 Packet In (AM) (BufID=225669510) (86B) => Conf. Root = 32768/700/00:12:e2:f8:3e:cd  Cost = 0  Port = 0x801a
```

```
▸ Header
▾ Flow Modification
  ▸ Match
    Cookie: 0x0010000000000000
    Command: New flow (0)
    Idle Time (sec) Before Discarding: 0
    Max Time (sec) Before Discarding: 0
    Priority: 350
    Buffer ID: None
    Out Port (delete* only): None  (not associated with a physical port)
  ▸ Flags
  ▾ Output Action(s)
    ▾ Action
        Type: Ethernet destination address (5)
        Len: 16
        MAC Addr: MS-NLB-PhysServer-03_00:00:00:26 (02:03:00:00:00:26)
    ▾ Action
        Type: Ethernet source address (4)
        Len: 16
        MAC Addr: 00:00:00_00:00:04 (00:00:00:00:00:04)
    ▾ Action
        Type: Output to switch port (0)
        Len: 8
        Output port: 12
        Max Bytes to Send: 65535
      # of Actions: 3
```

Figure 3-6 Screenshot of the CONET control plane traffic captured with Wireshark

The next Figure 3-7 shows one of the steps (i.e. Step 6) of the first workflow described in the previous section, when the CONET control plane traffic is detailed. In particular, this OpenFlow message is the flowmod that redirects to the CONET cache server all the requests from the CONET clients to the CONET servers once the CONET chunk has been already cached. The relevant information from this flowmod is highlighted in the figure.

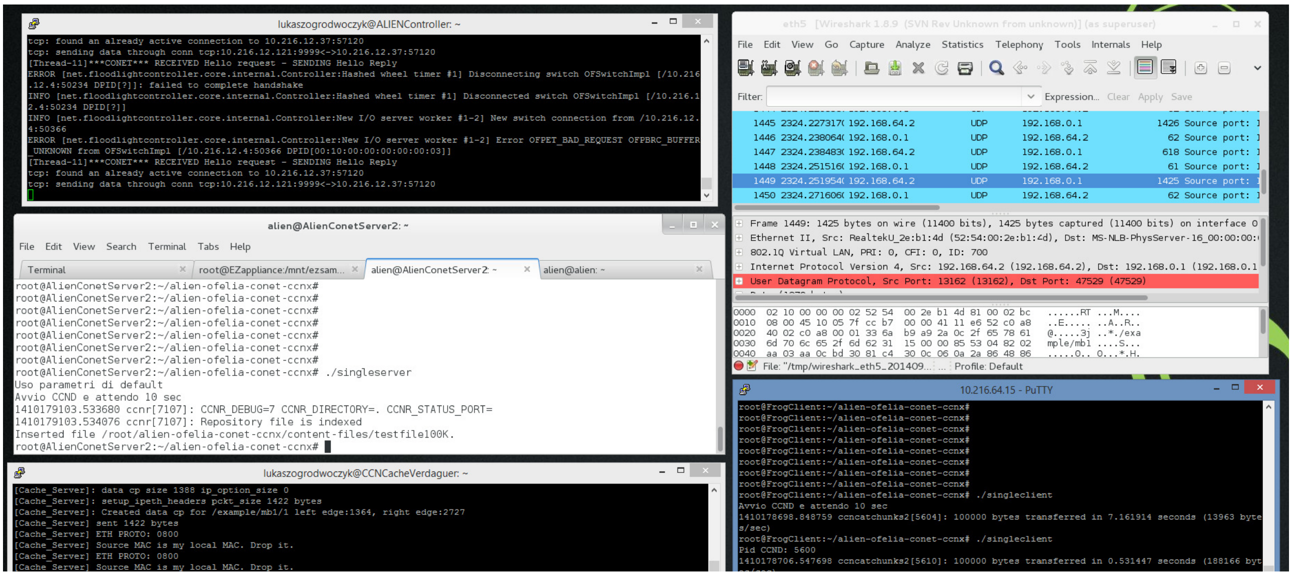


```
9656 256.669946000 10.216.12.121 10.216.12.4 OFP 178 Flow Mod (CSM) (112B)
▸ Internet Protocol Version 4, Src: 10.216.12.121 (10.216.12.121), Dst: 10.216.12.4 (10.216.12.4)
▸ Transmission Control Protocol, Src Port: 6633 (6633), Dst Port: 55846 (55846), Seq: 13679, Ack: 14303, Len: 112
▾ OpenFlow Protocol
  ▸ Header
  ▾ Flow Modification
    ▾ Match
      ▾ Match Types
          .... .... .... .... .... .... .... ...1 =  Input port: Wildcard (1)
          .... .... .... .... .... .... .... ..0. =  VLAN ID: Exact (0)
          .... .... .... .... .... .... .... .1.. =  Ethernet Src Addr: Wildcard (1)
          .... .... .... .... .... .... .... 1... =  Ethernet Dst Addr: Wildcard (1)
          .... .... .... .... .... .... ...0 .... =  Ethernet Type: Exact (0)
          .... .... .... .... .... .... ..0. .... =  IP Protocol: Exact (0)
          .... .... .... .... .... .... .0.. .... =  TCP/UDP Src Port: Exact (0)
          .... .... .... .... .... .... 0... .... =  TCP/UDP Dst Port: Exact (0)
          .... .... .... .... ..00 1000 .... .... =  IP Src Addr Mask: /24 (8)
          .... .... .... 0010 00.. .... .... .... =  IP Dst Addr Mask: /24 (8)
          .... .... ...1 .... .... .... .... .... =  VLAN priority: Wildcard (1)
          .... .... ..1. .... .... .... .... .... =  IPv4 DSCP: Wildcard (1)
        Input VLAN ID: 700
        Ethernet Type: IP (0x0800)
        Protocol: UDP (0x11)
        IP Src Addr: 192.168.0.0 (192.168.0.0)
        IP Dst Addr: 192.168.64.0 (192.168.64.0)
        TCP/UDP Src Port: 13071 (13071)
        TCP/UDP Dst Port: 6345 (6345)
      Cookie: 0x0010000000000000
      Command: New flow (0)
      Idle Time (sec) Before Discarding: 0
      Max Time (sec) Before Discarding: 0
      Priority: 350
      Buffer ID: None
      Out Port (delete* only): None  (not associated with a physical port)
    ▸ Flags
    ▾ Output Action(s)
      ▾ Action
          Type: Ethernet destination address (5)
          Len: 16
          MAC Addr: MS-NLB-PhysServer-03_00:00:00:26 (02:03:00:00:00:26)
      ▾ Action
          Type: Ethernet source address (4)
          Len: 16
          MAC Addr: 00:00:00_00:00:04 (00:00:00:00:00:04)
      ▾ Action
          Type: Output to switch port (0)
          Len: 8
          Output port: 12
          Max Bytes to Send: 65535
        # of Actions: 3
0000  00 16 3e 3d 97 fe 02 03  00 00 01 20 08 00 45 00   ..>=.... ... ..E.
0010  00 a4 2a 2a 40 00 40 06  e1 fd 0a d8 0c 79 0a d8   ..**@.@. .....y..
0020  0c 04 19 e9 da 26 a6 69  37 b8 a8 6e 2c ad 80 18   .....&.i 7..n,...
0030  2e c8 2e c3 00 00 01 01  08 0a 01 0a 1c 46 01 2a   ........ .....F.*
```

Figure 3-7 Screenshot of flowmod that redirects the content requests to the CONET cache server

The following figure shows sets of terminals to configure and monitor CONET application in PSNC Island:

- Top-left corner: CONET controller (Floodlight available at 10.216.12.121) – shows status of connection to OF switches and CONET cache server
- Left-center: CONET content server_2 (10.216.65.11) which index repository file and publish example content ("testfile100k") – this content is available under 192.168.64.2
- Bottom-left: CONET cache server (10.216.12.37)
- Top-right corner: Wireshark with captured data traffic between CONET content client_1 in UPV/EHU and content server_2 in PSNC (content request and UPD packets with content chunks)
- Bottom-right: CONET content client (10.216.65.15) in UPV/EHU Island – shows two transmissions of content to client_1 (UPV/EHU) – 192.168.0.1
  - First request: content send from server_2 in PSNC, completed in about 7.16 sec.
  - Second request: content send from cache server in i2Cat, completed in about 0.53 sec.



```
lukaszogrodwoczyk@ALIENController: ~
tcp: found an already active connection to 10.216.12.37:57120
tcp: sending data through conn tcp:10.216.12.121:9999<->10.216.12.37:57120
[Thread-11]***CONET*** RECEIVED Hello request - SENDING Hello Reply
ERROR [net.floodlightcontroller.core.internal.Controller:Hashed wheel timer #1] Disconnecting switch OFSwitchImpl [/10.216
.12.4:50234 DPID[?]]: failed to complete handshake
INFO [net.floodlightcontroller.core.internal.Controller:Hashed wheel timer #1] Disconnected switch OFSwitchImpl [/10.216.1
2.4:50234 DPID[?]]
INFO [net.floodlightcontroller.core.internal.Controller:New I/O server worker #1-2] New switch connection from /10.216.12.
4:50366
ERROR [net.floodlightcontroller.core.internal.Controller:New I/O server worker #1-2] Error OFPET_BAD_REQUEST OFPBRC_BUFFER
_UNKNOWN from OFSwitchImpl [/10.216.12.4:50366 DPID[00:10:00:00:00:00:00:03]]
[Thread-11]***CONET*** RECEIVED Hello request - SENDING Hello Reply
tcp: found an already active connection to 10.216.12.37:57120
tcp: sending data through conn tcp:10.216.12.121:9999<->10.216.12.37:57120
```

alien@AlienConetServer2: ~
File Edit View Search Terminal Tabs Help
Terminal
root@EZappliance:/mnt/ezsam...
alien@AlienConetServer2: ~
alien@alien: ~

```
root@AlienConetServer2:~/alien-ofelia-conet-ccnx#
root@AlienConetServer2:~/alien-ofelia-conet-ccnx#
root@AlienConetServer2:~/alien-ofelia-conet-ccnx#
root@AlienConetServer2:~/alien-ofelia-conet-ccnx#
root@AlienConetServer2:~/alien-ofelia-conet-ccnx#
root@AlienConetServer2:~/alien-ofelia-conet-ccnx#
root@AlienConetServer2:~/alien-ofelia-conet-ccnx#
root@AlienConetServer2:~/alien-ofelia-conet-ccnx# ./singleserver
Uso parametri di default
Avvio CCND e attendo 10 sec
1410179103.533680 ccnr[7107]: CCNR_DEBUG=7 CCNR_DIRECTORY=. CCNR_STATUS_PORT=
1410179103.534076 ccnr[7107]: Repository file is indexed
Inserted file /root/alien-ofelia-conet-ccnx/content-files/testfile100K.
root@AlienConetServer2:~/alien-ofelia-conet-ccnx#
```

```
lukaszogrodwoczyk@CCNCacheVerdaguer: ~
[Cache_Server]: data cp size 1388 ip_option_size 0
[Cache_Server]: setup_ipeth_headers pckt_size 1422 bytes
[Cache_Server]: Created data cp for /example/mb1/1 left edge:1364, right edge:2727
[Cache_Server] sent 1422 bytes
[Cache_Server] ETH PROTO: 0800
[Cache_Server] Source MAC is my local MAC. Drop it.
[Cache_Server] ETH PROTO: 0800
[Cache_Server] Source MAC is my local MAC. Drop it.
```

eth5 [Wireshark 1.8.9 (SVN Rev Unknown from unknown)] (as superuser)
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

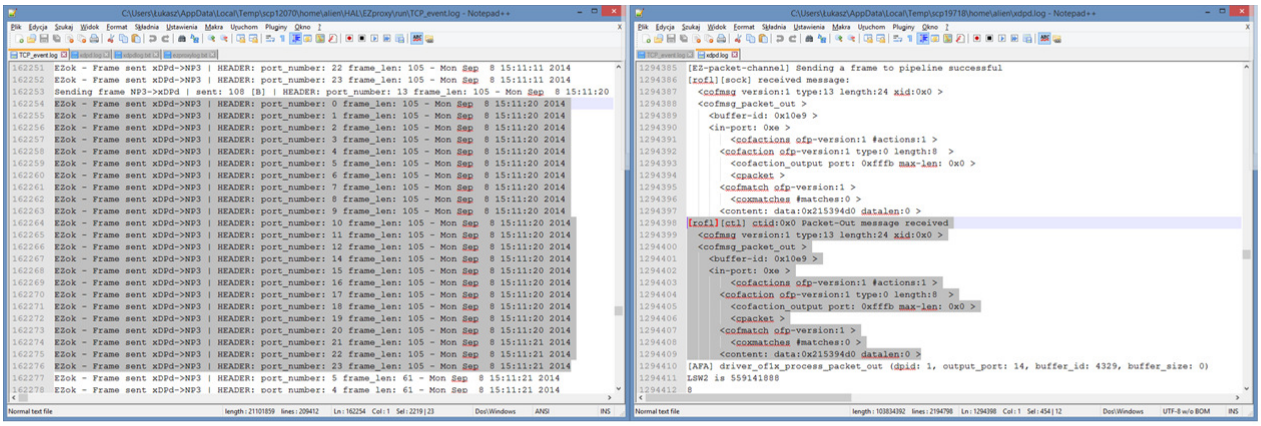
```
1445 2324.227317( 192.168.64.2   UDP   192.168.0.1    1426 Source port: 1
1446 2324.238064( 192.168.0.1    UDP   192.168.64.2     62 Source port: 1
1447 2324.238483( 192.168.64.2   UDP   192.168.0.1     618 Source port: 1
1448 2324.251516( 192.168.0.1    UDP   192.168.64.2     61 Source port: 1
1449 2324.251954( 192.168.64.2   UDP   192.168.0.1    1425 Source port: 1
1450 2324.271606( 192.168.0.1    UDP   192.168.64.2     62 Source port: 1

Frame 1449: 1425 bytes on wire (11400 bits), 1425 bytes captured (11400 bits) on interface 0
Ethernet II, Src: RealtekU_2e:b1:4d (52:54:00:2e:b1:4d), Dst: MS-NLB-PhysServer-16_00:00:00:
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 700
Internet Protocol Version 4, Src: 192.168.64.2 (192.168.64.2), Dst: 192.168.0.1 (192.168.0.1
User Datagram Protocol, Src Port: 13162 (13162), Dst Port: 47529 (47529)

0000  02 10 00 00 00 02 52 54  00 2e b1 4d 81 00 02 bc   ......RT ...M....
0010  08 00 45 10 05 7f cc b7  00 00 41 11 e6 52 c0 a8   ..E..... ..A..R..
0020  40 02 c0 a8 00 01 33 6a  b9 a9 2a 0c 2f 65 78 61   @.....3j ..*./exa
0030  6d 70 6c 65 2f 6d 62 31  15 00 00 85 53 04 82 02   mple/mb1 ....S...
0040  aa 03 aa 0c bd 30 81 c4  30 0c 06 0a 2a 86 48 86   .....0.. 0...*.H.
File: "/tmp/wireshark_eth5_201409... ... Profile: Default
```

```
10.216.64.15 - PuTTY
root@FrogClient:~/alien-ofelia-conet-ccnx#
root@FrogClient:~/alien-ofelia-conet-ccnx#
root@FrogClient:~/alien-ofelia-conet-ccnx#
root@FrogClient:~/alien-ofelia-conet-ccnx#
root@FrogClient:~/alien-ofelia-conet-ccnx#
root@FrogClient:~/alien-ofelia-conet-ccnx#
root@FrogClient:~/alien-ofelia-conet-ccnx#
root@FrogClient:~/alien-ofelia-conet-ccnx#
root@FrogClient:~/alien-ofelia-conet-ccnx# ./singleclient
Avvio CCND e attendo 10 sec
1410178698.848759 ccncatchunks2[5604]: 100000 bytes transferred in 7.161914 seconds (13963 byte
s/sec)
root@FrogClient:~/alien-ofelia-conet-ccnx# ./singleclient
Pid CCND: 5600
1410178706.547698 ccncatchunks2[5610]: 100000 bytes transferred in 0.531447 seconds (188166 byt
```

Figure 3-8 Screenshot of the terminals needed to perform the CONET experiment test (PSNC Island)

The following figure shows log files from:

- On the left: **xDPd-for-EZappliance** forwarding module with packet_out event marked (special port number 0xfffb)
- On the right: **EZproxy** with marked flood packets (from xDPd-for-EZappliance to NP-3 network processor) to all ports of EZapplaince (0-23) except incoming port no. 13



C:\Users\Łukasz\AppData\Local\Temp\scp12070\home\alien\HAL\EZproxy\run\TCP_event.log - Notepad++
Plik Edycja Szukaj Widok Format Składnia Ustawienia Makra Uruchom Pluginy Okno ?
TCP_event.log
xdpd.log
xdpdlog.txt
ezproxylog.txt
Normal text file
length : 21101859 lines : 209412
Ln : 162254 Col : 1 Sel : 2219 | 23
Dos\Windows
ANSI
INS

```
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 22 frame_len: 105 - Mon Sep  8 15:11:11 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 23 frame_len: 105 - Mon Sep  8 15:11:11 2014
Sending frame NP3->xDPd | sent: 108 [B] | HEADER: port_number: 13 frame_len: 105 - Mon Sep  8 15:11:20
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 0 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 1 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 2 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 3 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 4 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 5 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 6 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 7 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 8 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 9 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 10 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 11 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 12 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 14 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 15 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 16 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 17 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 18 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 19 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 20 frame_len: 105 - Mon Sep  8 15:11:20 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 21 frame_len: 105 - Mon Sep  8 15:11:21 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 22 frame_len: 105 - Mon Sep  8 15:11:21 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 23 frame_len: 105 - Mon Sep  8 15:11:21 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 5 frame_len: 61 - Mon Sep  8 15:11:21 2014
EZok - Frame sent xDPd->NP3 | HEADER: port_number: 4 frame_len: 61 - Mon Sep  8 15:11:21 2014
```

C:\Users\Łukasz\AppData\Local\Temp\scp19718\home\alien\xdpd.log - Notepad++
Plik Edycja Szukaj Widok Format Składnia Ustawienia Makra Uruchom Pluginy Okno ?
TCP_event.log
xdpd.log
Normal text file
length : 103834392 lines : 2194798
Ln : 1294398 Col : 1 Sel : 454 | 12
Dos\Windows
UTF-8 w/o BOM
INS

```
[EZ-packet-channel] Sending a frame to pipeline successful
[rofl][sock] received message:
  <cofmsg version:1 type:13 length:24 xid:0x0 >
  <cofmsg_packet_out >
    <buffer-id: 0x10e9 >
    <in-port: 0xe >
        <cofactions ofp-version:1 #actions:1 >
      <cofaction ofp-version:1 type:0 length:8  >
        <cofaction_output port: 0xfffb max-len: 0x0 >
        <cpacket >
      <cofmatch ofp-version:1 >
        <coxmatches #matches:0 >
      <content: data:0x215394d0 datalen:0 >
[rofl][ctl] ctid:0x0 Packet-Out message received
  <cofmsg version:1 type:13 length:24 xid:0x0 >
  <cofmsg_packet_out >
    <buffer-id: 0x10e9 >
    <in-port: 0xe >
        <cofactions ofp-version:1 #actions:1 >
      <cofaction ofp-version:1 type:0 length:8  >
        <cofaction_output port: 0xfffb max-len: 0x0 >
        <cpacket >
      <cofmatch ofp-version:1 >
        <coxmatches #matches:0 >
      <content: data:0x215394d0 datalen:0 >
[AFA] driver_of1x_process_packet_out (dpid: 1, output_port: 14, buffer_id: 4329, buffer_size: 0)
LSW2 is 559141888
8
```

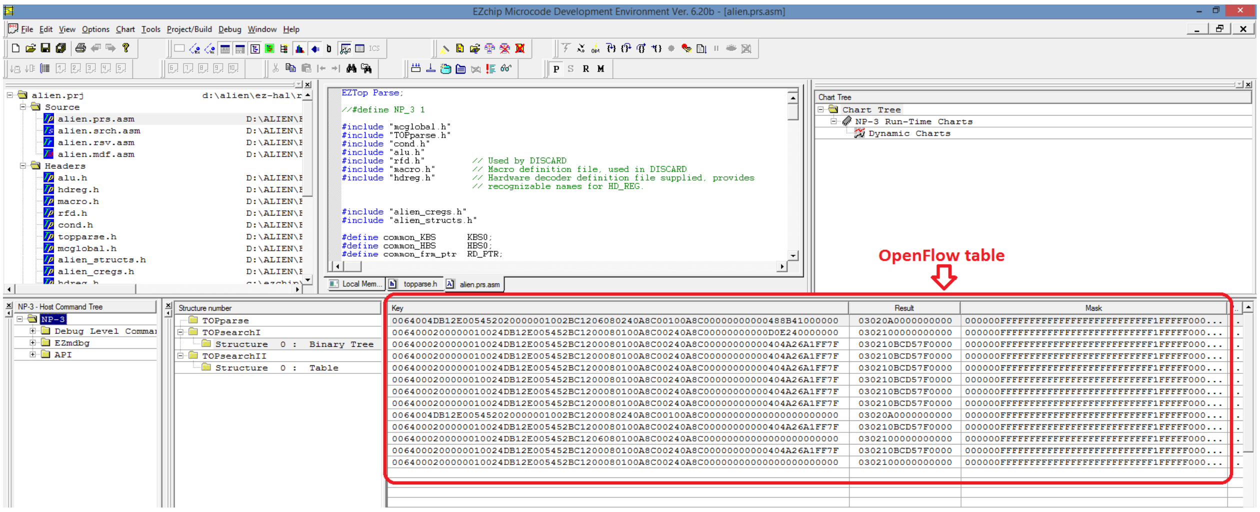
Figure 3-9 Screenshots of logs from xDPd-for-EZappliance and EZproxy modules

xDPd-for-EZappliance uses the following configuration during CONET experiment (*xdpd-ez-psnc.cfg* file).

```
config:{

   openflow:{
     logical-switches:{
                #Name of the switch dp0
```

```
               dp0:{
                        #Most complex configuration
                        dpid = "0x1100000000000001"; #Must be hexadecimal
                        version = 1.0;
                        description="This is an PSNC-EZappliance switch";

                        #Controller
                        controller-connections:{
                                main:{
                                        remote-hostname="10.216.12.121";
                                        remote-port=6633;
                                };
                        };
                        reconnect-time=1; #seconds

                        #Tables and MA
                        num-of-tables=1;

                        #Physical ports attached to this logical switch. This is
    mandatory
                        #The order and position in the array dictates the number of
                        # 1 -> eth1, 2 -> eth2, 3 -> eth3
                        ports    =    ("eth0",    "eth1",    "eth2",    "eth3",
    "eth4","eth5","eth6","eth7","eth8","eth9","eth10","eth11","eth12","eth13","e
    th14","eth15","eth16","eth17","eth18","eth19","eth20","eth21","eth22","eth23
    ");

               };
      };
    };

    system:{
                   logging-level="DEBUG";
      driver-extra-params="10.134.0.4"; #EZ Proxy IP address
    };
};
```

The following figure shows Microcode Development Environment (MDE) used to monitor status of search structures, registers, memories etc. inside NP-3 network processor into EZappliance.



```
EZTop Parse;

//#define NP_3 1

#include "mcglobal.h"
#include "TOPparse.h"
#include "cond.h"
#include "alu.h"
#include "rfd.h"          // Used by DISCARD
#include "macro.h"        // Macro definition file, used in DISCARD
#include "hdreg.h"        // Hardware decoder definition file supplied, provides
                          // recognizable names for HD_REG.


#include "alien_cregs.h"
#include "alien_structs.h"

#define common_KBS      KBS0;
#define common_HBS      HBS0;
#define common_frm_ptr  RD_PTR;
```

Key	Result	Mask
0064004DB12E005452020000001002BC1206080240A8C00100A8C000000000000488B41000000	03020A0000000000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C000000000000D0E240000000	0302100000000000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C0000000000000404A26A1FF7F	030210BCD57F0000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C0000000000000404A26A1FF7F	030210BCD57F0000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C0000000000000404A26A1FF7F	030210BCD57F0000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C0000000000000404A26A1FF7F	030210BCD57F0000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C0000000000000404A26A1FF7F	030210BCD57F0000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C0000000000000404A26A1FF7F	030210BCD57F0000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064004DB12E005452020000001002BC1200080240A8C00100A8C000000000000000000000000	03020A0000000000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C0000000000000404A26A1FF7F	030210BCD57F0000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1206080100A8C00240A8C000000000000000000000000	0302100000000000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C0000000000000404A26A1FF7F	030210BCD57F0000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...
0064000200000010024DB12E005452BC1200080100A8C00240A8C000000000000000000000000	0302100000000000	000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF1FFFFF000...

Figure 3-10 Screenshot of MDE with OpenFlow table

In current view OpenFlow table of EZappliance in PSNC Island is marked with flow entries captured during CONET experiment. OpenFlow table is implemented as search structure (binary tree) into TOPsearch I of NP-3 network processor [D3.2][D3.3].

3.5 Conclusions

As a general conclusion, the expectations where largely accomplished. The HAL modified hardware was successfully integrated and the CONET software ran without any problems. As usual, experimentation with real applications bring into sight problems that were not anticipated or discovered previously. In that sense, here are reflected as a hint for future developers some of the lessons learnt in the integration of the results in CONET, which were not discovered on local tests:

- In some situations Packet-out was not properly working and this was not detected on local hosts.
- In the case of ARP matching internal IPs, the difference between OF1.0 and OF1.2 behaviour was undetected.
- In one case, the flooding mechanism was not correctly implemented.
- In the ALHINP code, xDPd helpers (RGH and AGH) updated to latest versions due to problems in VLAN matching.
- Due to the fact that OFELIA project was finished, the support by the original project partners was in a best effort basis, so iterations took days. In any case, a warm thank you for the iMinds, Bristol and i2Cat OFELIA teams whose help was instrumental to get the project results tested in OFELIA.
- The allocation of a system wide free VLAN for interisland SLICEs (for OFELIA experiments and for CONET) is manual and error prone.
- The flowspaces need to be manually approved, and this is very time consuming as every change for the resources in charge in one island implies re-approval of the whole setup. Changes in one (controller IP, flowspace, vlan) affect all.
- When running experiments it was needed to take into account actual version of CONET limitations in the number of chunks, and adapt the experiments.

4 Summary

This document presents the results of the experiments performed in OFELIA with the results of the CONET experiment. For this, the following aspects have been presented:

- The deployment of developed OpenFlow-capable HAL prototypes for ALIEN hardware platforms as part of OFELIA environment
- The correct creation of OFELIA slices which makes use of resources from ALIEN hardware platforms, standard OpenFlow switches and virtualization servers
- The successful deployment of Content-Centric Networking in the form of CONET software in the ALIEN slice
- The description of the experiments executed showing the outputs of integration tests performed within OFELIA slices validating proper work of HAL-controlled platforms in OFELIA test bed as well as the deployment and verification of the CONET application over OFELIA.

For purpose of the validation of ALIEN hardware platforms, new OFELIA islands were created from the scratch in EHU, PSNC and PUT premises. In each island, the ALIEN hardware was installed and proper HAL software [D3.3] was deployed (see details in Table 4-1 Deployment of HAL-controlled hardware in OFELIA islands). Exception is UNIVBRIS Island, which already existed in the OFELIA. However, UNIVBRIS Island had to be modified in order to use ALIEN HAL prototypes for L0 switch as well; it was extended to cover GEPON system localized in UCL laboratory. Each ALIEN platform was taken under management of standard OCF software and only L0 switch was using modified version of OCF taking into account optical constrains of L0 switch. Different technologies like OpenVPN, VLAN and GRE were used to establish control plane and data plane connections to OFELIA islands (see details in Table 4-1 Deployment of HAL-controlled hardware in OFELIA islands).

ALIEN platform	HAL prototype software	Home island	OFELIA management software	OFELIA control network	Data Plane connectivity to OFELIA islands
NetFPGA	xDPd-for-NetFPGA	PUT Island (*new*)	Standard OCF	OpenVPN to iMinds	VLAN to PSNC
EZappliance	xDPd-for-EZappliance	PSNC Island (*new*)	Standard OCF	OpenVPN to iMinds	VLAN to Bristol and PUT
Cavium Octeon	xDPd for Octeon	EHU Island, device in Dell	Standard OCF	OpenVPN to EHU	VLAN to EHU
L0 switch	ADVA-ROFL-DP	UNIVBRIS Island	"Optical" OCF	OpenVPN to iMinds	VLAN to iMinds, i2Cat, PSNC

DOCSIS	ALHINP	EHU Island (*new*)	Standard OCF	VLAN to i2Cat over 1Gbps dedicated line	VLAN to i2Cat over 1Gbps dedicated line
GEPON	xCPd	UNIVBRIS Island, device located in UCL	Standard OCF	OpenVPN to UNIVBRIS	VPN tunnel

Table 4-1 Deployment of HAL-controlled hardware in OFELIA islands

A set of slices were created in OFELIA test bed in order to perform integration tests (please see Table 4-2). Two types of integration tests were performed:

- HAL developments integration (the validation of each single ALIEN platform with HAL in OFELIA)
- CCN integration (the validation of ALIEN platforms with HAL controlled by CONET in OFELIA)

Validation of each single ALIEN platform was performed in a simple slices set of resources from 1-2 islands. The requirement for this test assumed that, besides ALIEN platform, standard OFELIA resources (like standard OpenFlow switches and virtualization servers) would be used. Various test methods were used (see details on Table 4-2) but most popular was usage of the learning switch exercised with the ping tool. HAL developments integration tests were performed by partner responsible for a given ALIEN platform and its HAL prototype with support of team responsible from other participating OFELIA island. As a matter of fact the intended original schema of a punctual feedback to the development phase was replaced by a much more continuous process, as no architectural problems were faced, but only implementation related.

Integration type	Integration tests for	Number of islands	ALIEN platforms location	Standard OF switches location	Tests executed
HAL developments integration	NetFPGA	1	PUT Island	---	---
	EZappliance	2	PSNC Island	UNIVBRIS Island	Learning switch, ping, www, video
	Cavium Octeon	--	DELL as part of EHU Island	EHU Island	Ethernet Frame modification
	L0 switch	1	UNIVBRIS Island	UNIVBRIS Island	Learning switch, ping
	DOCSIS	2	EHU Island	i2Cat Island	Learning switch, ping, ssh, video
	GEPON	1	UCL as part of UNIVBRIS Island	UNIVBRIS Island	--
CCN integration	CONET	5	EHU, PSNC Island	i2Cat, iMinds, UNIVBRIS, PSNC Islands	Information retrieval and caching, information from cache

Table 4-2 ALIEN integrations overview

Successful results of single ALIEN platform integration tests were crucial milestone for performing CCN integration. CCN integration validates that CONET is operational in an OFELIA slice composed of HAL-controlled platforms. CONET introduced a new set of OpenFlow requirements [D5.1] which have been validated at a further level than the simple validation performed during HAL developments tests. The final, real world, experiment which involved a CCN application was also performed in a more complex slice composed of five islands (see details on Table 4‑2) as a joint work of many ALIEN partners. In D5.3 the results of additional experiments over the CONET platform will be shown, for example the implementation of QoS mechanism in the DOCSIS access network.

<THIS PAGE IS INTENTIONALLY LEFT BLANK>

5 References

[AFA-MGNT] https://www.codebasin.net/redmine/projects/rofl-core/repository/revisions/devel/entry/src/rofl/datapath/afa/fwd_module.h

[AFA-OPER] https://www.codebasin.net/redmine/projects/rofl-core/repository/revisions/devel/entry/src/rofl/datapath/afa/openflow/openflow12/of12_fwd_module.h

[AFA-NOTIF] https://www.codebasin.net/redmine/projects/rofl-core/repository/revisions/devel/entry/src/rofl/datapath/afa/cmm.h, https://www.codebasin.net/redmine/projects/rofl-core/repository/revisions/devel/entry/src/rofl/datapath/afa/openflow/openflow12/of12_cmm.h

[ALHINP] Victor Fuentes, Jon Matias, Alaitz Mendiola, Maider Huarte, Juanjo Unzilla and Eduardo Jacob, "Integrating complex legacy systems under OpenFlow control: The DOCSIS use case", EWSDN 2014, Budapest, Hungary, 2014

[CONET] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi and A. Detti, "Supporting Information-Centric Functionality in Software Defined Networks", SDN'12 Workshop on Software Defined Networks (ICC 2012), Ottawa, Canada, 2012

[D3.3] ALIEN project deliverable, "Final prototypes of hardware specific parts", July 2014

[D5.1] ALIEN project deliverable, "Usage Scenario description and requirements", February 2013

[F4F] http://www.fed4fire.eu/

[OCF] https://github.com/fp7-ofelia/ocf

[OFADD] Extensions to the OpenFlow Protocol in support of Circuit Switching, http://archive.openflow.org/wk/images/8/81/OpenFlow_Circuit_Switch_Specification_v0.3.pdf

[OFELIA] http://www.fp7-ofelia.eu/

[OFV] Azodolmolky, S.; Nejabati, R.; Shuping Peng; Hammad, A; Channegowda, M.P.; Efstathiou, N.; Autenrieth, A; Kaczmarek, P.; Simeonidou, D., "Optical FlowVisor: An OpenFlow-based optical network virtualization approach," Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference, vol., no., pp.1, 3, 4-8 March 2012

[OPEXP] M. Channegowda, R. Nejabati, M. Rashidi Fard, S. Peng, N. Amaya, G. Zervas, D. Simeonidou, R. Vilalta, R. Casellas, R. Martínez, R. Muñoz, L. Liu, T. Tsuritani, I. Morita, A. Autenrieth, J.P. Elbers, P. Kostecki, and P. Kaczmarek, "Experimental demonstration of an OpenFlow based software-defined optical network employing packet, fixed and flexible DWDM grid technologies on an international multi-domain testbed," Opt. Express 21, 5487-5498 (2013)

[PIP-COUNTER] https://www.codebasin.net/redmine/projects/rofl-core/repository/revisions/devel/entry/src/rofl/datapath/pipeline/platform/atomic_operations.h

[PIP-LOCK] https://www.codebasin.net/redmine/projects/rofl-core/repository/revisions/devel/entry/src/rofl/datapath/pipeline/platform/lock.h

[PIP-MEM] https://www.codebasin.net/redmine/projects/rofl-core/repository/revisions/devel/entry/src/rofl/datapath/pipeline/platform/memory.h

[PIP-PACKET] https://www.codebasin.net/redmine/projects/rofl-core/repository/revisions/devel/entry/src/rofl/datapath/pipeline/platform/packet.h