Integrating complex legacy systems under OpenFlow control: The DOCSIS use case

Victor Fuentes, Jon Matias, Alaitz Mendiola, Maider Huarte, Juanjo Unzilla and Eduardo Jacob
University of the Basque Country (UPV/EHU)
ETSI de Bilbao, Alda. Urquijo S/N, 48013 Bilbao, SPAIN
Email: {Victor.Fuentes, Jon.Matias, Alaitz.Mendiola, Maider.Huarte, Juanjo.Unzilla, Eduardo.Jacob}@ehu.es

Abstract—**The possibility to deploy telecommunication services based on the availability of a fully flow-aware network is an appealing possibility. Concepts like Network Service Chaining and Network Function Virtualization expect the information to be manageable at the flow level. But, for this concept to be available for the development of user-centric applications, the access network should also be made flow-aware. In this paper we present the integration of a legacy DOCSIS based access network under an OpenFlow Control Framework by using the Hardware Abstraction Layer designed in the FP7 ALIEN project. The result is a dynamic wide area OpenFlow switch that spawns from the aggregation switch to the home equipment and hides all the complexity (including the provisioning) of the access technology to an unmodified and standard OpenFlow controller. As a result, the access network can react not only to any kind of user traffic but also to the connection of CPE to the network. The approach used is technology independent, and the results have been successfully demonstrated over a Cisco based DOCSIS access network.**

Index Terms—**Access Networks, OpenFlow, DOCSIS, Proxy**

I. Introduction

The Software Defined Networking (SDN) initiative is clearly gaining momentum, and after some time on the top of the peak of the *Hype Cycle* [1], the promise of benefits is becoming more and more real, undoubtedly pushed by the OpenFlow phenomena.

Nevertheless, there are still several aspects that could hinder the wide deployment of SDN based architectures. On the one hand, the main reason is that it involves replacing hardware to get an OpenFlow interface. On the other hand, is the unavailability of an OpenFlow control on some networking sub-systems. In fact, there are many proposals and current experiences using OpenFlow in campus and datacenter networks, and even in the interconnection of datacenters as stated in [2]. Furthermore, Research and Education Networks like GEANT and GENI are already testing solutions for the core. But, no much work has been done in areas such as the access network.

One of the reasons is the complexity of the access network, that includes management interfaces for configuration of physical parameters which are not available in Ethernet based networks and for which OpenFlow does not have the required commands. Another reason is that there is usually a provisioning phase that is needed as a previous stage to any packet or frame exchange. In the case of DOCSIS access networks (which will be further explained in Section II), without the provisioning of the Cable Modems (CM), it is not possible to achieve the reactiveness needed at the OpenFlow controlled networks. It is not possible to forward traffic from the CMs to the Cable Modem Termination System (CMTS), thus, no *PACKET_IN* messages will arrive to the Controller. Finally, the dynamic aspect of an access network in which users switch on and off their equipment out of the network's operator control gives an additional complexity which should be considered.

Trying to integrate any kind of equipment under an OpenFlow Control Framework generally implies having on the bottom a controllable or programmable datapath and on the top an OpenFlow endpoint.

There are many ways to classify the available datapaths. For the purpose of this article, a broad classification can take into account the availability of an Ethernet-aware frame manipulation capability. Programmable switches with an accessible Board Support Package (BSP), NPU families like [3] or [4] with a suitable SDK or optimized x86 based boxes with a SDK like [5] will fit into this first category. Basically, the remaining networking equipment that can move Ethernet frames but is not able to manipulate them (like legacy Ethernet switches) or equipment that just does not use the frame concept (like optical ROADM, DOCSIS or GPON based architectures) constitute the second group.

The classic way to solve this problem is through the use of a Hardware Abstraction Layer (HAL) that offers a northbound interface that hides different implementation details and reuses much code when adapted to different hardware platforms.

ALIEN Project proposes a Hardware Abstraction Layer (HAL) [6] which aims to *"design and implement the required building blocks for a novel Hardware Abstraction Layer that can facilitate the unified integration of alien types of network hardware elements (i.e. network element that do not support natively OpenFlow)"*. This indeed means that both types of datapath can be managed, as ALIEN HAL includes support for other kind of technologies like optical ROADM found at core networks or access network related technologies like GPON or DOCSIS. ALIEN HAL relies on the eXtensible OpenFlow DataPath daemon (xDPd) [7] and the Revised OpenFlow Library (ROFL) [8], which have been considerably extended in the project, to provide respectively a *"multi-platform, multi OF version, open-source datapath"* which supports *"Network processors, FPGAs, ASICs.. as well as non-ethernet devices"* and *"OpenFlow support (...)to build control applications,* *controller frameworks and/or data path elements"*.

This paper is organized as follows. Section II presents the relevant related work and introduces the integration of a DOCSIS access network into an OpenFlow Control Framework. Section III presents the proposed architecture. Then, Section IV describes in detail the design of the architecture whereas Section V introduces the implementation details. Finally, Section VI and VII present the validation and also the conclusions and future work.

II. Integration and related work

This paper presents the experience gained adapting a DOCSIS based access network to an OpenFlow Control framework by using the approach proposed in ALIEN. As a reminder, the simplest DOCSIS architecture includes a CMTS which is linked downward to the CM located at the Customer Premises by a cable based RF distribution system and upward to an Aggregation Switch (AGS) that gets the traffic from several CMTSs and connects the whole setup to the Internet.

Additionally, there is a provisioning system sometimes referred as the DOCSIS Provisioning System (DPS). The DPS relies on NTP, TFTP and DHCP servers to configure the CMs and provide them with the correct firmware and configuration files. Each CM behaviour is determined by the configuration file, which indicates the equipment operation mode (i.e., switch, router or WiFi router), Internet and management related network configuration and also its QoS configuration, to cite a few.

It is specially worthmentioning that QoS configuration is of uttermost importance in access networks due to the bandwidth ressource sharing. In the case of DOCSIS access networks, QoS is guaranteed to end-users by means of a unidirectional transport mechanism called *service flow*. At the configuration files of the CM, besides the default pair of Upstream and Downstream service flows, additional service f lows can be configured to transport specific services, defined by a set of QoS parameters and packet classifiers.

There are several approaches to integrate a DOCSIS access network under an OpenFlow Control Framework. The first two deal with legacy CMTS based architectures whereas the third one does not. The first approach relies on having a Controller which is indeed aware of the particularities of the CMTS and its DPS and offers a northbound interface tailored to control and manage them, while offering a southbound OpenFlow-based interface. This could be in fact, the case *2d* referred in [9]. The second approach makes the CMTS, the CMs, the DPS and the AGS appear as a wide-area OpenFlow switch by means of a Proxy that is linked to the OpenFlow Controller. Again, this can be the case *5b* in [9] if the closest Forwarding Equipment (FE) to the user is indeed the resulting wide-area OpenFlow switch. Finally, the third one involves a full OpenFlow aware CMTS.

On the practical side, there are several initiatives, or at least pointers in the Web. The most visible one [10] was presented in the Open Networking Summit [11] and intends to *"Develop a PacketCable PCMM/COPS southbound plugin and supporting modules to allow the OpenDayLight Controller to provision CMTS as a network element that manages service flows with dynamic QoS"*. This approach will give the possibility to create Service Flows on reactive or proactive mode and provision them using PCMM. CableLabs has also published several OpenFlow and SDN related inventions, particularly, in[12] they propose an hypothetical CMTS that does have a native OpenFlow control interface that implements an extension to support different QoS service flows. There is a reference about a commercial product that could implement something related to this [13] but no other information is available.

III. Integrated DOCSIS OpenFlow Architecture

This section describes the architecture designed to integrate DOCSIS access networks under OpenFlow control. The objective is to integrate the entire access network as a wide-area virtual OpenFlow switch. As the CMTS is considered a closed box without Ethernet manipulation capability which cannot be extended or reprogrammed, the integration requires using some helpers to support the OpenFlow switch model abstraction. The proposal fits into the second category mentioned in Section II.

A. The resulting Architecture

The Integrated DOCSIS OpenFlow architecture with additional helpers added is represented in the next figure.

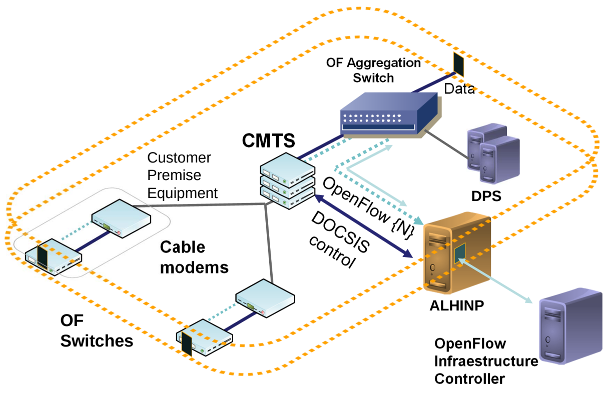

Fig. 1. OpenFlow enabled DOCSIS architecture

In this architecture, the AGS, which is OF enabled, is considered to be part of the access network helper in order to provide functionalities not supported by DOCSIS. The proposal also includes OUIs that will help to extend the native packet classification provided by the DOCSIS traffic classifiers to a more flexible and fine grained one. As it will be later explained in section VII, this could not necessarily imply adding an additional box.

The ALHINP (ALien Hardware INtegration Proxy) is the component which will interface to an outer Controller and manage all the components that constitute the wide-area OpenFlow switch.

The only requirement for the OF AGS regarding OpenFLow is that it must support OF version 1.1+, because multiple tables are required to support incoming vlan-tagged traffic from the clients. Despite the use of multiples tables, the virtual model reports to the Controller a unique one.

B. ALHINP

ALHINP (ALien Hardware INtegration Proxy) [14], which is based on ALIEN HAL, is the main component of the solution. It exposes two northbound interfaces (OF 1.0 and OF 1.2) and, at the same time, orchestrates and manipulates all the underlying network devices to make them perform as an integrated wide-area OpenFlow switch. ALHINP proxy processes OF messages received from the Controller and generates specific actions (at DOCSIS configuration and management level) and, if needed, OF messages to be sent to the required devices.

The architecture of the proxy is detailed in the next figure:

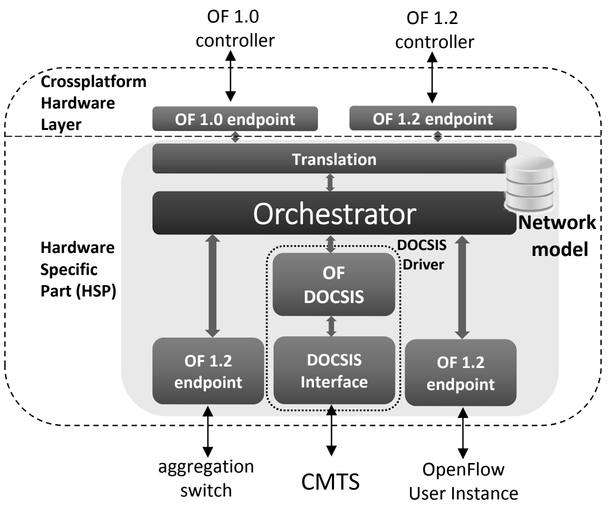

Fig. 2. ALHINP architecture detail

1) Orchestrator: The orchestrator is a part of HSP (Hardware Specific Part) of the ALIEN HAL architecture [6]. By taking information from the network model, it is able to process the incoming OF messages and also generate and orchestrate the delivery of new messages, through the suitable driver, to each device of the network (OF endpoint in case of interaction with an OpenFlow Controller) or Access network driver (DOCSIS driver in this case). In this later case, the mapping between OF flow rule parameters and corresponding DOCSIS classifier ones are defined.

2) Translator: This module defines the algorithms that enable the translation between the real topology and the virtual topology exposed to the Controller. It also controls the VLAN assignment depending on the information provided by the network model.

3) Network model: The network model defines the topology and how the connection between devices is performed. It also provides information related to the management of the network, like the amount of clients supported, CM MAC listings, etc.

4) DOCSIS Driver: Here, the specific code where the interaction with the specific network access technology (DOCSIS in this case) is defined. As it is expected, a generic, in relation to the technology, interface is provided. Later, a translation mechanism converts this generic interface into specific commands and protocols (specific for CMTS CLI or SNMP commands).

5) OpenFlow endpoints: These modules implement the OpenFlow endpoints, which will be in charge of the connection of the northbound interface with the corresponding Controller or with the OpenFlow devices of the platform. It is important to note that they maintain the connection status.

IV. DESIGN DETAILS

The present Section describes the design in detail, focusing on the network boot up, the QoS configuration and the implementation of the OpenFlow messages.

A. Booting the network up

In DOCSIS access networks, CMs can join or leave the network asynchronously, resulting in the dynamic appearance of virtual ports. In these situations, the network reports its new condition to the ALHINP. This subsection describes how the internal communication flows involved (CM to provisioning servers, AGS to ALHINP, OUIs to ALHINP) are detected and enabled by ALHINP.

When the CMTS is booted up, the RF resources are configured over the RF interface for upstream and downstream channels. The CMTS is configured to have the Layer 2 Virtual Private Network (L2VPN) mode enabled. As a result, the CMs are assigned a unique VLAN VID that CMTS uses to tag the traffic incoming from a certain CM.

Meanwhile, the AGS connects to the ALHINP (Fig. 3 A2) as soon as it is available. On the one hand, the ALHINP listens for connections coming from the AGS and the OpenFLow User Instances (OUIs). On the other hand, it periodically tries to connect to the Controller. Depending on the first connection being established, the process varies.

- If a FEATURES_REQUEST message is received from the Controller during the initial handshake process, but AGS is not already connected, no (available) ports are reported on the corresponding reply.
- As soon as the AGS is connected to the ALHINP, its ports, which are now available, will be reported to the Controller via PORT_STATUS messages.
- If the connection with the AGS was already set up, the ports will be announced by means of the FEATURES_REQUEST message, along with their virtual identifier, provided by the AGS.
- It is important to note that due to the fact that virtual ports can appear at any time, if some setting is provided via a SET_CONFIG message to every port in the virtual wide-area switch, this command should be stored to be subsequently sent to every new CPE joining the system.

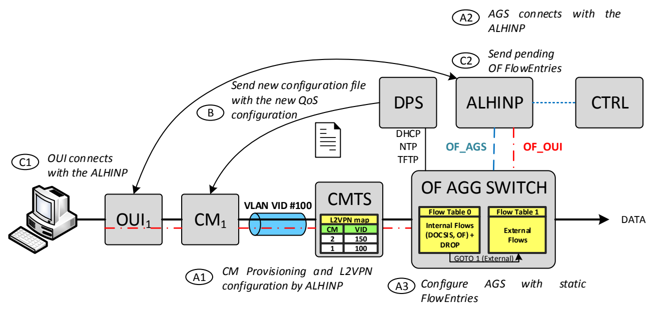
A2 AGS connects with the ALHINP
C2 Send pending OF FlowEntries
Send new configuration file with the new QoS configuration
B
DPS
ALHINP
CTRL
DHCP
NTP
TFTP
OF_AGS
OF_OUI
C1 OUI connects with the ALHINP
OUI_1
CM_1
VLAN VID #100
CMTS
L2VPN map
CM
VID
2
150
1
100
OF AGG SWITCH
Flow Table 0
Internal Flows (DOCSIS, OF) + DROP
Flow Table 1
External Flows
GOTO 1 (External)
DATA
A1 CM Provisioning and L2VPN configuration by ALHINP
A3 Configure AGS with static FlowEntries

Fig. 3. Internal flow required for normal operation of the platform

Once the initial handshake finishes, ALHINP will deploy some static rules over the AGS (Fig. 3 A3). Those rules are related to:

1) Detection of DOCSIS provisioning traffic (Fig. 3 A1): When an unknown CM tries to get the suitable configuration to join the DOCSIS network, traffic must be detected. In order to apply the L2VPN configuration in the CMTS referred to that CM -according to the algorithm set- DHCP is captured and processed by the ALHINP to obtain the CM MAC address.
2) Detection of OF control-traffic incoming from OUI (Fig. 3 C1): Once the CM setup is finished, its OUI will try to connect the ALHINP. Detection of this connection is required to establish an association between its MAC (and DPID, as OpenFlow standard states that a device's MAC should be mapped into the DPID lower bits) and VLAN_VID used for that CM.
3) Default drop rule for all traffic at internal ports without specific rule.
4) Deploy a GOTO_TABLE 1 rule for every packet incoming from any external port of the Aggregation Switch, as all flow entries deployed in flow-table 0 are related to internal flows, or they are rules for untagging incoming traffic from CMs. Flow-table identifier is also abstracted to OF controller.

B. Configuring QoS over DOCSIS Layer

When a CM joins the DOCSIS access network it must overcome a provisioning process; a configuration file is downloaded by the CM (Fig. 3 B) where the *service flows* and the associated *traffic classifiers* are defined. The evolution of this system quasi-static QoS configuration mechanism to a dynamic one based on PCMM is being studied.

C. Connecting OUI to ALHINP

Once the AGS and the client CM are successfully configured and the control traffic is detected and enabled by the ALHINP, the OUI will be able to connect with the ALHINP. In turn, the ALHINP will provide to the OUI the OF configuration previously sent and stored directed to it.

After loading rules, newly available ports may be reported as *available* to controller via PORT_STATUS message with their virtual identifier.

D. PACKET_IN management

A PACKET_IN message is generated in the switches when there is no matching rule for a flow or when it is specifically set via an OUTPORT controller action.

When a PACKET_IN is generated due to a flow table mismatch , if the *inport* is available to the controller, the *inport* field is rewritten. Otherwise it is discarded. If the PACKET_IN has its origin in a rule, specific actions can be taken (e.g. set a new VLAN over the CMTS).

E. FLOW_MOD management

This is one of the most challenging messages to map into this DPID distributed architecture. The message is analysed to get the *inport* from the matching fields and any *outport* action present. For each pair of *inport* and *outport*, an internal path is created (when several *outport* actions are present).

Processing differs depending on *in_port* is wildcarded or not. If *inport* is not wildcarded, orchestration can be simplified. When *inport* is present (not wildcarded) and *outport* is OFP_CONTROLLER, IN_PORT, LOCAL or NONE (not present), processing of the path can be done in the same way, that is, *inport* is translated and a rule is installed at the DPID associated with the *inport*. However, if the *inport* is another real port of the infrastructures, a path over several DPIDs is required, as shown in figure 4.

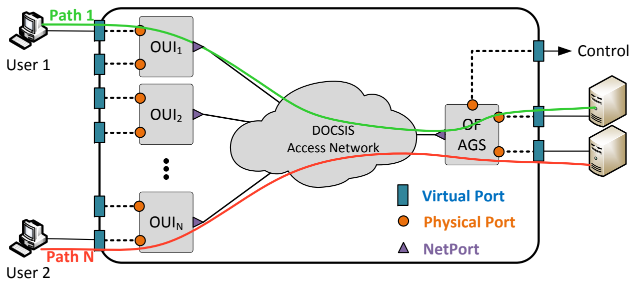

Fig. 4. Path required to support a flow over the network

1) Client to aggregation flow (upstream): First, a flow entry is installed in the second table of the AGS. The match is formed by the original match provided by the Controller but, *inport* is changed to netport (port connected to the CMTS). It is also added metadata match with VLAN VID associated with the corresponding CM as field value. Installed actions are taken from the original action set, where the outport (virtual) is modified with equivalent real port id. If some action related to the QoS must be taken by the DOCSIS network, it will be configured as a second step. Finally, the flow entry over the OUI is installed. The new match will be formed by the original one, modifying the virtual *inport* with the real port id. Only an action is performed, and it is just to send the packet to the network. The vlan tag will be automatically added by the CMTS.
2) Agregation to Client flow (downstream): The first entry is inserted into the OUI, with the original match,

substituting the original *inport* with the netport (the one connected to the CM). Actions are taken from the original action set, but changing the *outport* with its real equivalent. The rule installed in the AGS is installed into the second table. Match is taken from the original message, but translating the *inport*. Action set is formed by push VLAN tag, *SET_FIELD* (VLAN VID) corresponding to the CM and *outport* is set to *netport*.

3) Client to client flow: this case is a mix of upstream and downstream cases. Flow entry corresponding to the exit DPID is formed by the original match modifying the *inport* by the netport value and applying the original action set with the outport translated into the real port identified. The flow entry installed in the AGS only rewrites the incoming VLAN tag with VLAN associated with the egress OUI and it sends the packet through the same interface (*outport* = *inport*). Finally, a flow entry is inserted into the source DPID, with the original match modified (*inport* translated) and the action for sending the packet to the netport.

FLOW	DPID_SRC	DPID_middle	DPID_DST
Client To aggregation	**match** Original match ! Inport modified	------	**Match (table 1)** original match + Metadata = VLAN_VID + Inport = netport
	Actions: +outport: netport		**Actions:** original actions + outport modified
Aggregation to Client	**Match (table 1)** original match ! Inport modified	------	**match** Original match ! Inport = netport
	Actions: + Push VLAN + Set VLAN (cm_dst) + Outport netport		**Actions:** original actions ! outport modified
Client to Client	**match** original match + Inport modified	**Match (table1)** original match + Metadata = VLAN_VID (cm_src) + Inport = netport	**match** original match ! inport = netport
	Actions: + Outport: netport	**Actions:** Push VLAN Set VLAN (cm_dst) + outport IN_PORT	**Actions:** original actions ! outport modified

TABLE I
PARTIAL FLOW ENTRIES INSTALLATION SUMMARY

Partial flows are sent to DPIDs in the opposite order to the flow direction, that is, when a flow from a OUI to the AGS must be installed, the first flow entries to be set belong to the AGS DPID. Secondly, actions over the DOCSIS network are applied (if it proceeds) and finally, the flow in the OUI is installed. This way the packet loss is avoided due to the lack of rules in any of the devices involved in the path for that packet.

If *inport* is not set in the match of a FLOW_MOD, messages must be stored while hard timeout is still valid. In order to maintain the coherence with the behaviour of an Openflow switch model. When a new DPID joins the network, the cached rules are verified to install that rules concerning to recently connected DPID. The summary in table I describes all mentioned above.

FLOW_MOD with outport = OFP_ALL is not implemented due to its complexity. Implementation would require sending the same message tagged with each VLAN used in the network in aggregation, which is an excessive processing.

F. PACKET_OUT

PACKET_OUT performing is processed directly from the control plane, by setting the bytes sent to the controller the maximum possible into the OpenFlow switch PACKET_IN configuration and sending the entire packet to controller. This way, every packet can be directly sent attached to the PACKET_OUT message .

G. STATS management

Flow entry cache, used for FLOW_MOD processing, is also used when it is required to get data for a certain statistic request. The simplest way to ask for statistics is by using the *cookie* parameter, as a unique flow identifier. But this identifier is not always correctly by applications running on top of the controller. Statistics are collected from aggregation, where the most of the flows are processed

V. IMPLEMENTATION

In order to test the suitability of the presented architecture, an experimental setup has been deployed as part of the ALIEN project, consisting on a set of OUIs, a DOCSIS access network and an OpenFlow AGS.

A. OpenFlow User Instance

As depicted in Figure 1, at this moment several OUIs are running over CPEs, which are Linux embedded machines that implement an OpenFlow 1.2 switch based on xDPD [7] and ROFL [8]. Each CPE is equipped with two Gigabit Ethernet interfaces to communicate with the CMs (as currently OpenFlow in-band support is not implemented) and additional Ethernet interfaces used as a user port. These Ethernet interfaces are connected to the end-user hosts.

B. DOCSIS network

The proposed architecture requires an entirely bridged access network because it needs to support L2 circuits established between the end-user clients and the servers. As a consequence, both CMs and CMTS have been configured to work in bridge mode.

Firstly, the Cisco EPC3825 CMs used at the deployment have been configured to work in bridge mode by overriding their default configuration and disabling the WiFi interface that runs by default on them. Each CM is equipped with four Ethernet interfaces, where two of them are used to connect the CM with the OUI (data aggregation + OpenFlow control) running on the CPE.

Secondly, the CMTS used in the deployment is a Cisco uBR7246-VXR with a MC16U DOCSIS card (supporting up to 5 physical upstream links and 1 for downstream). The operating system that runs in the CMTS is the Cisco iOS release 12.2 with support for 802.11q and DOCSIS 2.0. In order to work in bridge mode, the CMTS, which usually works as a router, encapsulates the traffic of each VM into different VLANs. With this mechanism, it is possible to identify the CM that forwarded certain user data.

C. OpenFlow Aggregation Switch

The OpenFlow AGS has been also implemented using xDPd [7] and ROFL [8]. This software switch runs on a SR1690WB server with 4 Gigabit Ethernet ports plus a virtual one, where the provisioning servers are located. The first Gigabit Ethernet interface is used for both the data exchanged with the end-user hosts and for the control traffic originated in the OUIs. The second Gigabit Ethernet interface is used for the data exchanged with the core network. The third interface is used for the control-traffic directed to and coming from the ALHINP exclusively from clients OUIs, and the fourth interface is for OpenFlow control of the AGS.

D. ALHINP

ALHINP is designed and implemented around the ROFL 0.4 library, which provides basic OpenFLow 1.0 and 1.2 endpoints to interface both switches or controllers. The current implementation of the ALHINP uses the OpenFlow 1.0/1.2 protocol to establish connection with the controller. However, the connection to the OpenFlow AGS and the OUIs is done entirely using OpenFlow 1.2 version. Even ALHINP internally uses several flow-tables, although only one is announced to the controller.

It also contains a Command Line Interface (CLI) to configure the equipment of the DOCSIS network as needed. Though it is not mandatory, the ALHINP runs on on a Linux virtual machine located at the same server that the OpenFlow AGS.

E. OpenFlow controller

Finally, the OpenFlow controller in charge of the control of each OUI and the OpenFlow AGSs is Ryu [15]. As in the case of the ALHINP, the OpenFlow controller also runs on the same Linux virtual machine, located at the same server of the OpenFlow AGS. The OpenFlow AGS, ALHINP and the OpenFlow controller communicate with each other by means of an internal virtual network.

VI. Validation

In order to test the validity of the proposed architecture, the experimental setup described in section V has been tested with the OFTEST conformance test-suite [16] successfully passing the tests for OF1.0 and basic tests for OF1.2 (groups or any advanced features are not currently implemented).

Furthermore, the aforementioned experimental setup was presented at the Future Internet Assembly (FIA2014) that took place in March of 2014 at Athens.

VII. Conclusions and further work

This paper shows that the last mile is no longer a stopper for deploying flow-aware applications that reach the user premises. The architecture proposed is able to provide an unprecedented flexibility in this regard and is able to cope with the dynamicity of an access network.

The proxy base scheme used in this approach can be easily deployed over other access networks based on different technologies, such as GPON. Under this scheme, only a OF-GPON driver is needed, that will take into account OLT related configuration issues.

The integration of the OUI in the CM is already being studied, as CISCO Cable Modems run a version of eCoS [17], which is an open-source project. This will boost the deployment of the solution, because no in-home additional equipment will be needed for an ISP to deploy flow-aware services.

VIII. Acknowledgments

This research was partly funded by the Seventh Framework Programme of the European Commission, through the ALIEN (317880) project and by the Spanish Ministry of Science under the A3RAM-NG (AAA in Next Generation Access Networks) TIN2010-21719-C02-01. The authors want to thank Norberto Ojinaga and Octavio Alfageme from Euskaltel and Pedro Ortega from CISCO for their support.

References

[1] Gartner Inc., “Hype cycle for networking and communications, 2013,” Tech. Rep., 2013.

[2] U. Hoelzle, “Open Network Summit - The Google OpenFlow network is in Production,” April 2012.

[3] Cavium, “OCTEON III CN7XXX Multi-Core MIPS64 Processors,” 2014. [Online]. Available: http://www.cavium.com/OCTEON-III_CN7XXX.html

[4] EZChip, “NP Family of Network Processors,” Jun. 2014. [Online]. Available: http://www.ezchip.com/p_np_family.htm

[5] INTEL Corp., “Intel®DPDK: Data Plane Development Kit,” Jun. 2014. [Online]. Available: http://dpdk.org/

[6] ALIEN Project, “ALIEN Hardware Abstraction Layer White Paper,” http://www.fp7-alien.eu/files/deliverables/ALIEN-HAL-whitepaper.pdf, 2013.

[7] BISDN, “Extensible DataPath daemon (xDPd),” Jun. 2014. [Online]. Available: https://www.codebasin.net/redmine/projects/xdpd/wiki

[8] ——, “Revised OpenFlow Library (ROFL),” Jun. 2014. [Online]. Available: https://www.codebasin.net/redmine/projects/rofl-core/wiki

[9] G. Hampel, M. Steiner, and T. Bu, “Applying software-defined networking to the telecom domain,” in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, April 2013, pp. 133–138.

[10] Project OpenDayLight, “Packet Cable PCMM,” https://wiki.opendaylight.org/view/Project_Proposals:PacketCablePCMM, 2014.

[11] T. Kee, “Open Network Summit - OpenDaylight In Action at Cable-Labs,” April 2014.

[12] CableLabs, “DOCSIS Service Flow provisioning via OpenFlow,” 2012. [Online]. Available: {http://www.cablelabs.com/wp-content/uploads/2014/04/60421-publish.pdf}

[13] Oliver Solutions, “accessFlowNE^{TM},” Jun. 2014. [Online]. Available: {http://oliver-solutions.com/accessflowne/}

[14] University of the Basque Country (UPV/EHU), “ALHINP Develoment Repository,” Jun. 2014. [Online]. Available: https://github.com/i2t/ALHINP

[15] NTT Corporation, “RYU SDN framework,” Jun. 2014. [Online]. Available: http://osrg.github.io/ryu/

[16] P. FloodLight, “Oftest,” Jun. 2014. [Online]. Available: https://github.com/floodlight/oftest/blob/master/Detailed_Testing_Methodology.txt

[17] eCos, “embedded Configurable operating system,” http://ecos.sourceware.org/, 2014.