ABSTRACTION LAYER
FOR IMPLEMENTATION OF EXTENSIONS
IN PROGRAMMABLE NETWORKS

Hardware Abstraction Layer (HAL)

Whitepaper

1. Abstract

This Whitepaper describes the concept of Hardware Adaptation Layer (HAL) for applying the OpenFlow protocol to the non-OpenFlow hardware. In this document, first, the motivation of designing the HAL is explained as well as the high level goals and supported network platforms. Then, the logical architecture of HAL is presented including the architectural and functional requirements. As next chapter the network hardware integration models are provided and the supported hardware platforms are classified. Finally, the proposed implementation of HAL is presented including the software interfaces for platform integrators such as the Abstract Forwarding API and the Pipeline Interface.

2. Motivation for the HAL

Software Defined Networking concept is getting more and more popularity as a solution for more efficient network management. The OpenFlow protocol is the most mature proof of concept of the SDN. The idea of separating the data and control elements in network devices is being supplied in new network equipment. Although, the OpenFlow is usually not supported in many specific network equipments e.g. the optical devices, hardware without possibility to be upgraded or without packet processing API. Enabling OpenFlow on all those non-OpenFlow devices with possible simplicity for the integrators are the main motivation assumptions for disseminating the HAL concept.

2.1. Purpose of proposed solution and high level goals

Although OpenFlow has clear specifications [ONF], when it comes to implementing those specifications on devices or platforms, each implementation is different because of heterogeneity of platforms and their architectures. Moreover, the official OpenFlow specification is designed for wired Ethernet platforms and it does not support other platforms such as circuit switched and wireless platforms. Designing a mechanism, targeting hardware-level forwarding platforms to provide a clean, simple, extensible and flexible interface between software and hardware will facilitate providing abstracted information of underlying hardware or platform for creating an abstracted elements for controlling software. This mechanism will expose all of functionality and performance of modern networking hardware, while maintaining the useful properties of embedded operating systems on the hardware platform but at the same time it will be platform agnostic.

As part of SDN activities the ALIEN project aims to undertake the challenge of provide simple concept for non-OpenFlow hardware platforms integrators by designing and defining functions of a Hardware Abstraction Layer (HAL). This abstraction mechanism aims to hide hardware complexity as well as technology and vendor specific features from OpenFlow control framework.

2.2. Supported network platforms

The targeted platform for HAL are:

- Packet Switch: All devices that perform any form of L2 switching are fell into this group.
- Optical: Basically all circuit switched devices (WDM/TDM) are categorized in optical group.
- Network Processor (NP) / NetFPGA: Any device that its data plane can be programmed to perform any (L2-L4) packet processing.
- Access Networks technologies (DOCSIS, GPON,GEPON...): Since these technologies' data plane architecture is so different to the rest of platforms, it was decided to put it in separate group so that its abstraction could be done without compromising its functionality.

In the ALIEN project the implementation of the HAL architecture will be validated on various network equipment such as: EZappliance with EZchip NP3 network processor, GEPON, NetFPGA, DOCSIS, ATCA with Cavium Octeon network processor, ADVA Layer 0 Switch, Dell/Force10 7024 switch with ASIC and Cavium Octeon.

3. Logical architecture of HAL

Following the SDN concept and according to OpenFlow architecture, the data path or forwarding engine inside of an OpenFlow device must be controlled and managed by a controller who resides outside of the device and communicates to the device via a secure channel. The data path is represented to the controller in an abstracted fashion as a table or tables with flows holding packets information and actions associated to them which could be manipulated (programmed) by the controller software.

3.1. Requirements for HAL

Optimum HAL for network devices to support OpenFlow protocol needs to have certain features so that it can support the current and future architecture of networking devices. The requirements for HAL was categorised in functional and architectural.

The desired HAL should follow some architectural guideline in order to support the intended platforms.

- HAL has to reside between controller and the device i.e. controller can get control access on data path only through HAL.
- HAL components should be reusable as much as possible. This will lead to smaller code, better troubleshooting and management.
- It should be portable, platform-independent and flexible to give ability to be implemented on various platforms.

HAL has to have certain functionalities in order to provide a platform for controllers to control the data path:

- It has to provide an interface to upper layers (applications) for handling hardware dependent issues
- It has to provide rich information enough to support full featured network operating systems (controllers)
- It has to expose interfaces for hardware reprogrammability. For example, an ASIC packet-processing fast path could support programmability for deep-packet inspection operations. Another example could be NetFPGAs which users can define their own function on the hardware.
- It has to provide interface for hardware resource management.
- It has to provide mechanism for controlling shared abstracted resources such as forwarding-table in virtualized environment

3.2. Proposed HAL architecture

Following the functional and architectural requirements for desired HAL a unified architecture is proposed. Hardware platform categorization helps to build organized and modular components which then yields to create a flexible and extensible HAL. In the following, HAL components and their functionalities are described.

The proposed HAL consists of two separate layers: upper Hardware Interface Layer (HIL) and lower Hardware Presentation Layer (HPL).

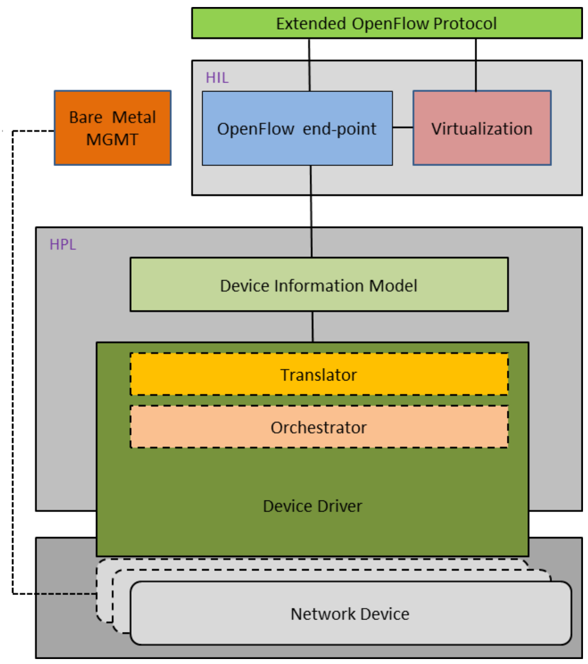

Figure 1 - HAL architecture

HIL: Components in the HIL altogether provide an implementation of common device control and device management protocols i.e. OpenFlow, etc. The HIL components are independent of underlying hardware platform.

OpenFlow endpoint: is a protocol endpoint responsible for maintaining connection with a controller.

Virtualization: This component is proposed as a method to provide virtualization capabilities to HAL-compatible devices. It has three different functionalities: (i) communicating with the Network Management System (NMS) of the Infrastructure Provider (by considering the NMS the responsible for properly configuring the virtual network instances), (ii) communicating with multiple OF controllers through the OpenFlow protocol and (iii) slicing the flowspace. The Virtual Agent (VA) slices the overall flowspace among many OF Controllers based on the configuration received from the NMS. On the other hand VA ensures setting-up control channels with each connected OF controller by creating one virtual switch per controller. Virtual switches of the same VA may use different versions of the OpenFlow protocol, depending on the capabilities of the controller assigned to their slice.

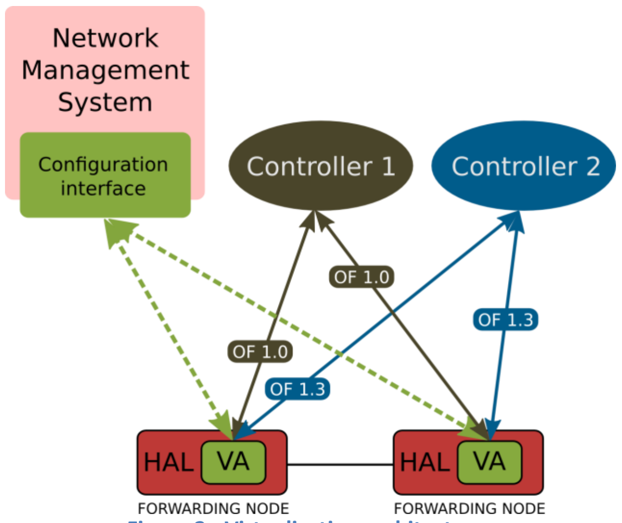

Figure 2 - Virtualization architecture

Bare Metal Management: Although the ultimate goal of creating HAL is to provide an interface for controlling hardware, each hardware platform has different protocol for device management and configuration. The bare metal management component is responsible for hardware initialization and management. In case of configuration changes in hardware platform or introducing new user-defined functionality into the platform, this component will provide tools and interfaces for such events.

HPL: The Hardware Presentation Layer (HPL) includes few components which have distinct functionality. The northbound of HPL provides an API for upper layer to create extended OpenFlow table. In the following, HPL components and their role and functionality are explained.

Device Information Model: is a model of configured OF switch that represents switch's state which probably is platform independent. It is not a representation of a specific hardware. Pure data model without processing.

Translator: is part of device driver. it is responsible for translating all entries and actions from OpenFlow switch model (DIM) into platform specific commands and configurations. This module is not obligatory, e.g. for devices that supports OpenFlow data model.

Orchestrator: is part of driver components and responsible for configuring the entire subnetwork, i.e. multiple components, to simulate the behavior of OF switch modeled by DIM. As an example, GEPON needs configuration for both optical and electrical parts for appropriate functioning.

Device driver: is a piece of software that performs data processing using hardware device/accelerators. In case of closed devices, it only configures a device to obtain the same behavior like modeled by DMI OF switch (a kind of translation). It is a platform dependent component of the HAL.

4. Network hardware integration models

Various types of network hardware give different possibilities for HAL integration with devices. ALIEN HAL supports three main hardware integration models for devices with different requirements, constraints and hardware capabilities.

4.1. Built-in Model

In the Built-in model, the overall HAL implementation runs inside the network device. The HAL acts as the only exposed device API. This integration model is suitable for fully programmable devices with available SDK and for device vendors. The only limitation is that a network device must have enough general purpose processing power to run HAL. Built-in model offers the best level of integration thus should be used for each network device which offers enough capabilities.

4.2. Proxy Model

In Proxy model the HAL implementation runs (at least partially) on a separate machine which is an integral part of a node. The network device itself is configured by HAL implementation using CLI, SNMP or other ways provided by a device vendor. For better performance entire data processing should be done using data plane parts of a network device, not in a proxy machine . This integration model is suitable for closed platforms or platforms with not enough resources for Built-in HAL.

4.3. Orchestrator Model

Orchestrator model is an extension of the Proxy. HAL implementation runs on separate machine and exposes a part of a network as a single device. In this case the HAL implementation must orchestrate a group of devices to behave as a single network element. This integration model is suitable for platforms that are constituted of tightly coupled elements like DOCSIS (CMTS and Cable Modems) and GEPON (OLT and ONU).

5. HAL implementation

The modular architecture of xDPD [XDP] that could use ROFL [ROF] pipeline library for its datapath implementation and freedom that ROFL library provides for implementation OpenFlow protocol on various platform regardless of their underlying hardware and software, leads to conclusion that the combination of xDPD and ROFL could build ALIENs HAL architecture. In the following, it will be explained that how the building blocks of xDPD and ROFL could be used in the ALIEN project.

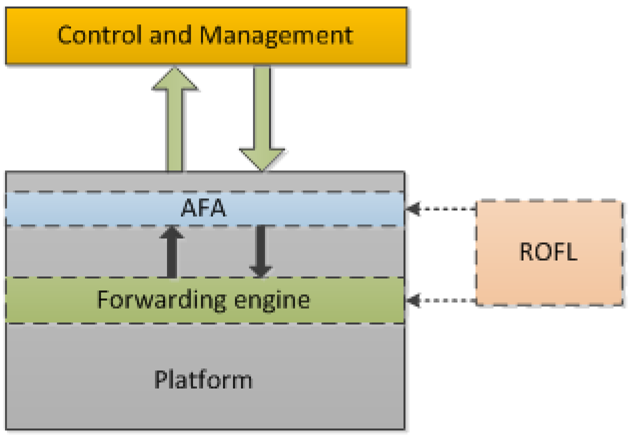

Figure 3 - Sample HAL implementation

5.1. Software interfaces available for platform integrators

Considering the architecture of ROFL , since the library is platform agnostic, it could be used on any platform for different purposes. In a basic conceptual view, a developer could implement an OpenFlow forwarding module by calling ROFL library. Also, ROFL provides an Abstracted Forwarding API (AFA) for controlling and management of the logical datapath. Part of AFA covers OpenFlow API but it also supports management tasks API. The library provides:

- basic support for OpenFlow protocol and maps protocol wire representation into a set of C++ classes. The library tries to hide the details of protocol and provide more understandable API for developing. The current version supports OpenFlow 1.2 but it aims to be a multi-version library.
- an abstraction to build an OpenFlow endpoint for controllers.
- capabilities to create data path elements which could be OpenFlow. The pipeline could be implemented on various platforms e.g., ASIC, FPGA, GNU/Linux, etc.

5.2. CMM module

Control and Management Module is in charge of managing and controlling of physical and logical devices as well as keeping abstracted version of logical switch and its associated OpenFlow end-point. The CMM is hardware agnostic and it is common for all Forwarding Modules (FM).
The CMM communicates with FM using Abstracted Forwarding API (AFA). CMM will be platform independent but also it gives the ability for FM to do tasks internally without communicating with CMM.

5.3. AFA interface

Abstract Forwarding API is binding, the platform specific, forwarding module with control and management module, which is keeping the abstracted switching information. AFA interface with device specific implementation provide below features:

- Management actions: that is creating and destructing logical switches. It binds the logical switch instance and ports or network interfaces.
- I/O subsystem: this component is in charge of port (network interface) discovery, inventory, configuration and management. It also monitors the traffic of the interfaces to capture the packets for sending them for processing, that means OpenFlow processing, and afterward execute the action based on the policy.
- Packet processing: this component is in charge of processing a packet as defined in the OpenFlow specification.
- Background tasks: this component is in charge of managing OpenFlow pipeline, buffer or flow entry expiration.
- Event generation: this component generates events for management and control entities such as port status changes, OpenFlow packet-in and other OpenFlow messages and errors.

5.4. Pipeline interface

As it was mentioned FM employs ROFL pipeline library for implementing the OpenFlow pipeline. The ROFL pipeline library supports logical switch concept. It also supports for multi version of OpenFlow which is convenient for developers to implement different version of OpenFlow protocol based on their device capabilities. It is worth to mention that ROFL pipeline library is not a fully ready to use OpenFlow data path with support of platforms subsystems such as I/O. Although the subsystems are part of FM, they are platform specific components which are different on each platform. The I/O subsystems could be seen as device or interface drivers which have different architecture and API on every platform and systems.

6. Conclusions

Hardware abstraction concept is well known from operating systems where HAL hide hardware complexity from the rest of the system. In the SDN networks with split data and control plane, problem of abstracting different network hardware platforms is growing again. ALIEN project takes the problem of enabling OpenFlow on “alien” hardware without native support of this protocol. Providing an abstraction layer for alien hardware platforms is the key towards network programmability.

Presented in chapter 3 general HAL architecture provides modular and flexible representation of different networking equipment. HAL architecture allows the data path of the network devices to be presented in an abstracted fashion as pipeline (flow tables with OF actions) to the OF controllers. Based on dual-layer HAL concept initial HAL implementation proceeds. xDPD and ROFL presented in chapter 5 of this Whitepaper are key software libraries that could be reused for any hardware integrators.

The building blocks of xDPD framework could be mapped to the HAL architecture. xDPD and ROFL support standard OpenFlow specification (currently v1.2) for packet switching devices. On the contrary, in ALIEN project, circuit switching devices is one of the main targeted platforms that HAL needs to support. With that regard, xDPD and ROFL will be extended to support circuit switches according to OpenFlow circuit addendum v0.3.
AFA and ROFL pipeline could represent a hardware abstraction layer for creating a platform agnostic OpenFlow datapath. While ROFL pipeline provides necessary ingredients for implementing an OpenFlow pipeline inside FM, AFA is in charge of acting as a platform agnostic proxy between the data path and the controller.

At the moment, xDPD forwarding module has been implemented on few platforms such as OCTEON, Broadcom and GNU/Linux which its implementation is available for public.

7. References

[ROF]	Rofl repository and wiki, https://www.codebasin.net/redmine/projects/rofl-core/wiki
[XDP]	xDPD repository and wiki, https://www.codebasin.net/redmine/projects/xdpd/wiki
[**ONF]**	OpenFlow Documentation, https://www.opennetworking.org/sdn-resources/onf-specifications

SEVENTH FRAMEWORK PROGRAMME

Grant Agr. No. 317880

www.fp7-alien.eu