UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

OPENFLOW 1.3 WITH RYU CONTROLLER

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

CARLOS FERNÁNDEZ SÁNCHEZ

In partial fulfilment

of the requirements for the degree in

CIÈNCIES I TECNOLOGIES DE LES TELECOMUNICACIONS

Advisor: José Luis Muñoz

Barcelona, July 2015

Abstract

This thesis explores the possibilities of OpenFlow 1.3, Ryu and Open vSwitch. It mainly focuses on Multi-Protocol Label Switching (MPLS) Software-Defined Network (SDN) implementations.

OpenFlow 1.3, Open vSwtich and Ryu have been documented in detail, providing examples of how they work.

During this project three applications have been analysed, and two have been developed. This implementations prove how powerful these tools could be, and provide a good example of the versatility and granular control provided by OpenFlow-driven Software Defined Networks.

As a result of this thesis a detailed document has been created, collecting all information and results obtained during the project.

Resum

Aquesta tesi explora les possibilitats d' OpenFlow 1.3, Ryu i Open vSwitch. Es centra principalment en les implementacions de xarxes MPLS amb tecnologies SDN.

OpenFlow 1.3, Open vSwitch i Ryu han estat documentats en detall, i es proveeix d'exemples del seu funcionament.

Durant aquest projecte, tres aplicacions han estat analitzades i altres dues han sigut desenvolupades. Aquestes implementacions proven com de potents son aquestes eines i constitueixen un bon exemple de la versatilitat i granularitat que proveeixen les SDN basades en OpenFlow.

Com a resultat d'aquesta tesi s'ha creat un document detallat, que recull tota la informació i resultats obtinguts al llarg d'aquest projecte.

Resumen

Esta tesis explora les posibilidades de OpenFlow 1.3, Ryu y Open vSwitch. Se centra principalmente en la implementación de redes MPLS usando tecnologías SDN.

OpenFlow 1.3, Open vSwitch y Ryu han sido documentados detalladamente, y se provee de varios ejemplos sobre su funcionamiento.

A lo largo de este proyecto, se han analizado tres aplicaciones y se han desarrollado otras dos. Estas implementaciones prueban cuan potentes son estas tecnologías y constituyen un buen ejemplo de la versatilidad y la granularidad que proveen las SDN basadas en OpenFlow.

Como resultado de esta tesis se ha creado un documento detallado que recoge toda la información y resultados obtenidos a lo largo de este proyecto.

Acknowledgements

I want to thank my advisor, José Luis Muñoz, for introducing me to Software-Defined Networking and OpenFlow, and guiding me along this project.

Revision history and approval record

Revision	Date	Purpose
0	09/07/2015	Document creation
1	10/07/2015	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Carlos Fernández Sánchez	cfernandez@openmailbox.org
José Luis Muñoz	jose.munoz@entel.upc.edu

Written by:		Reviewed and approved by:	
Date	09/07/2015	Date	10/07/2015
Name	Carlos Fernández	Name	José Luis Muñoz
Position	Project Author	Position	Project Supervisor

Table of contents

1. Introduction

The goal of this document is to provide an **executive summary of the project**. An extensive and detailed documentation of the technologies used in this project and the applications analysed and developed is **annexed to this memory**.

1.1. Objectives of the thesis

The purpose of this project is to analyze and develop a set of applications with the Ryu Controller in order to explore and document de different possibilities of OpenFlow 1.3. Open vSwitch is used to implement the OpenFlow Switches used in all the scenarios analyzed.

Along the course of this project, we decided to focus on MPLS implementations with the set of tools proposed, as the field of Software-Defined Networking is so wide its possibilities are almost endless.

OpenFlow 1.3, Open vSwitch and Ryu have been properly documented in the annex, in order to provide the necessary information to understand the applications discussed

The following scenarios have been discussed, analyzed and documented:

- A simple L2 switch implemented with Ryu and OpenFlow 1.3.
- A MPLS network approach implemented with Open vSwitch tools and OpenFlow 1.3.
- A MPLS network approach implemented with Ryu and OpenFlow 1.3
- An IP router Ryu application, executed within a complex topology using OpenFlow 1.3 and configured through a RESTful API.
- A MPLS network implemented with Ryu executed within a complex topology using OpenFlow 1.3 and configured through a RESTful API.

1.2. Requirements and specifications

The requirements of this project can be organized in two different categories: technical requirements and conceptual requirements. Technical requirements are focused on programming skills and implementations meanwhile conceptual requirements are focused on the basics of Computer Networking.

Conceptual requirements:

- Switching and Bridging
- Routing
- Multiprotocol Label Switching
- REST
- OpenFlow

Technical requirements:

- Use of Mininet
- Use of Ryu Framework
- Use of Open vSwitch and its tools
- Software development with Python Object-Oriented programming
- Use of the software version control system SVN.

- Use of the LaTeX editor to document.
- Use of Linux kernel-based OS.

1.3. Third party resources

Some of the applications analyzed have been developed by the Nippon Telegraph and Telephone Corporation and are licensed under the Apache License. This is the case of the following applications:

- The simple switch application
- The IP Router application

The MPLS applications proposed in chapter 4 of the annex have been developed using the two applications above as a base.

The scenario where MPLS protocol implementation with OpenFlow and Open vSwitch is analyzed is a modification of an experiment proposed by *Università degli Studi Roma Tre*, which can be found as an open resource in its Wiki page, found in [12]

The methods and procedures used to analyse and develop such applications are detailed in section 3 of this document.

1.4. Work Plan

Work Packages:

Project: Planning and environment setup	WP ref: WP1	
Major constituent: Software	Sheet 1 of 3	
Short description: Setup of the different tools that will be used during the project	Planned start date: 2/03/2015 Planned end date: 15/03/2015	
	Start event: Beggining of the project End event: Setup is done	
Internal Task T1: Environment setup: Mininet, SVN, LaTeX... Internal Task T2: Planning the project	Deliverables: Project Plan	Dates:

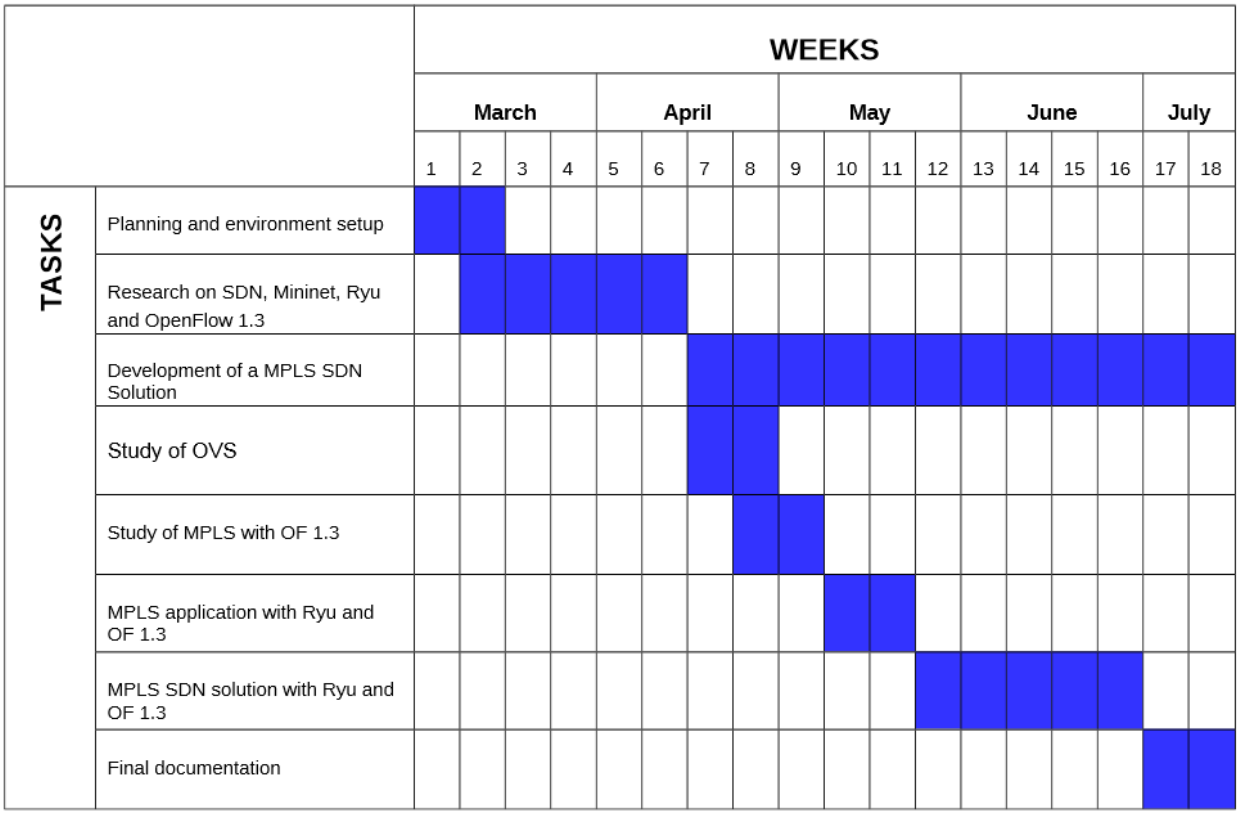
Project: Research on the tools	WP ref: WP2	
Major constituent: Research and testing	Sheet 2 of 3	
Short description: Testing and learning about the different tools: Mininet, Ryu, Python and OpenFlow 1.3	Planned start date: 9/03/2015 Planned end date: 12/04/2015	
	Start event: Project Plan delivered End event: Classic Switch implemented with Ryu and OF	
Internal Task T1: Practice Python Internal Task T2: Studying Mininet Possibilities Internal Task T3: Deep research on OpenFlow 1.3 Internal Task T4: First app with Ryu	Deliverables: Critical Review	Dates:

Project: Development of a MPLS SDN Solution	WP ref: WP3	
Major constituent: Development and simulation	Sheet 3 of 3	
Short description:	Planned start date: 13/04/2015 Planned end date: 10/07/2015	
	Start event: First app End event: Memory delivery	
Internal Task T1: Study and documentation of OVS Internal Task T2: Study and documentation of a MPLS network using OVS and OF 1.3 Internal Task T3: Development and documentation of a MPLS Ryu application Internal Task T4: MPLS SDN using Ryu and OF1.3 and OVS. Internal Task T5: Final documentation	Deliverables: Project Memory	Dates:

Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
3	T1	Study of OVS	Meeting with supervisor	8
3	T2	Study of MPLS with OF13	Meeting with supervisor	9
3	T3	MPLS with Ryu & OF13	Meeting with supervisor	11
3	T4	MPLS SDN solution	Meeting with supervisor	16
3	T5	Final Documentation	Memory Delivery	18

Time plan



1.5. Deviations from the original project

The original project did not plan to focus on MPLS networking.

2. State of the art of the technology applied in this thesis:

2.1. OpenFlow

OpenFlow is the first standard communications interface defined between the control and forwarding layers of an SDN architecture (see [4] and [5]). OpenFlow allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual. OpenFlow uses the concept of flows to identify network traffic based on pre-defined match rules that can be statically or dynamically programmed by the SDN control software. It also allows IT to define how traffic should flow through network devices based on parameters such as usage patterns, applications, and cloud resources.

This standard is developed by the Open Networking Foundation (ONF) (See [9]).

The version of OpenFlow used along this project is the 1.3 version. The protocol and switch specification (1.3.0) for this version was released in June 25, 2012. The last update of this specification has been released in April, 2015 and corresponds to the version 1.3.5 (see [6]). The current version of OpenFlow is the 1.5 version, which specification was released on Dec. 19, 2014 and updated on April, 2015.

2.2. Open vSwitch

Open vSwitch (OVS) is a multilayer software switch licensed under the open source Apache 2 license. The goal of its developers is to implement a production quality switch platform that supports standard management interfaces and opens the forwarding functions to programmatic extension and control. Current version is 2.3.2. See [10] for more information.

2.3. Ryu

Ryu is a component-based framework for Software-Defined Networking applications. It provides software components with well defined API that make it easy to create network management and control applications. Ryu supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc.

Ryu currently supports OpenFlow 1.0, 1.2, 1.3, 1.4. It's fully developed in Python and all of the code is freely available under the Apache 2.0 license. It is the tool chosen to develop the Control Plane in the experiments exposed in this document. See [8] and [11] for more information.

3. Methodology / project development:

The analysis and development of the applications discussed in this thesis could not be possible without the following list of tools

- **Mininet:** A tool that allows the creation of **realistic virtual networks** on a single machine.
- **Open vSwitch (OVS):** A multilayer virtual switch. The main feature of this switch is that it's open to programmatic extension using OpenFlow. Several tools are included in the OVS package and some of them have been used for the development of the applications. The most useful has been **ovs-ofctl** which is a tool for managing the OpenFlow flows of an OVS datapath.(See page 41 of the annex)
- **Ryu SDN Framework:** Toolset for Software Defined Network development. Includes an OpenFlow controller, among other tools.
- **Wireshark:** Network protocol analyser

The methodology used for application analysis could be summarized in the following steps:

1. Creation of a topology using Mininet, where the switches used are Open vSwitch datapaths.
2. Capture the packets at the loopback interface using Wireshark, with a filter to display only OpenFlow packets.
3. Run the Ryu Application we want to analyse.
4. Review the messages exchanged between the switch and the Ryu Application. Take notes on the thesis diary.
5. Send an ICMP message from one host to other. Study the packets captured and take notes.
6. Study the application's code. Take notes.
7. Insert log messages in the code in order to relate the events with blocks of code.
8. Run the application again. Sending messages between hosts. Study the log. Take notes.
9. Capture packets on the relevant interfaces of the datapaths. Take notes.
10. Document the application's behaviour, relating it with the code.
11. Test alternative implementations and document the results.

The methodology used for application development could be summarized in the following steps:

1. Conceive a first approach to achieve the desired functionality: Draw the topology, make a flow chart.
2. Design the application: Think of which classes/methods/functions have to be implemented in order to achieve the desired results.
3. Implement the code. Using Ryu's logger, ovs-ofctl, and Wireshark for debugging purposes.
4. Analyse the application carefully. Take notes. Document the results
5. If the desired behaviour is not achieved, conceive a different approach and start again from step 2.

4. Results

As a result of this project, using the tools explained in previous section, three applications have been analysed, and two have been developed. The last implementation developed proves how powerful these tools could be. A third implementation is proposed, but not developed, addressing the issues of the previous implementation.

The most relevant results are summarized in the following lines.

- The analysis of the Simple Switch implemented with **Ryu** (Page 69 of the Annex), lets us conclude that the application is not robust enough for a topology where more than a single MAC address can be found in the same port. A solution for this problem is proposed: Using source MAC addresses instead of ingress ports as a match.
- The analysis of the MPLS network implemented with **ovs-ofctl** (Page 45 of the Annex) lets us conclude that the current version of Open vSwitch is not mature enough for high efficiency MPLS forwarding, as only one label can be popped from packets (see [1]), and kernel mode is not supported. Version 2.4 of Open vSwitch and Linux 3.19 promise to fix these issues. See [3] and [7].
- The analysis of the IP router implemented with **Ryu** (Page 92 of the Annex) exemplifies that REST APIs and OpenFlow controllers are a powerful combination, allowing a network administrator to change the behavior of a network, or to retrieve data from it without changing a line of the code being executed by Ryu. Also shows how to take advantage of ARP handling methods and controller buffers to properly deliver packets to its destination. This application takes advantage of Object-Oriented Programming to implement a rather complex behavior in a long but fairly understandable code, which can be easily extended.
- The development of the first approach to an MPLS network implemented with **Ryu** (Page 87 of the Annex) is useful to understand how to properly write MPLS rules to the switches, but leads to a poor implementation that only works for hosts which IP addresses belong to the same network. As a result of this, a second approach is developed.
- The development of the second approach to an MPLS network implemented with **Ryu** (Page 92 of the Annex)**,** shows how easily the behavior of a network can be changed with few lines of code. The potential of Software Defined Networking is shown when it's proven that with a minimal pre-configuration through an API, the Application Manages to create a real MPLS network.

The annex contains every application analyzed in detail, providing code, packet captures, flow dumps and logs.

5. Budget

All the software used in this project is licensed under a Free Software License (Apache, GPL…). The software used did not add any cost to the thesis.

The operating system used for the development is Ubuntu, which is a free GNU/Linux distribution, so it also does not add any cost to the thesis

The estimated number of hours dedicated to this thesis is 380h. Taking in account the **real cost** of a recently graduated engineer, which is 12.5 €/h. The total cost of the thesis is 4750€.

6. Conclusions and future development:

6.1. Conclusions

It seems clear after seeing the results that SDN adoption can improve network manageability, scalability, and agility. OpenFlow-driven Software Defined Networks can be developed to be easily managed and upgraded, as they can be debugged as a piece of software.

It's too soon to confirm that MPLS could be completely implemented using Open vSwitch, as the current version does not fully support this protocol. However, a centralized control plane as proven to be very useful with this kind of networks, as it can see the network as whole, in comparison to decentralized networks.

Ryu has proven to be a really powerful framework to prototype and debug Software-Defined Networks. Once the behavior desired is achieved, the design could be implemented in a more complex and efficient platform than Python, for production purposes.

6.2. Future Development

The third implementation proposed in the annex could be implemented with the current technologies, and even a fourth implementation that uses kernel datapaths(see [2]) and more than one label per packet could be developed once the new version of Open vSwitch is released.

Another approach to future development using this thesis as a base, is to expolore other fields of OpenFlow that have not been treated in this project, such as implementing all the OpenFlow Actions and Instructions, or using Group Tables.

Also, Open vSwitch could be further analysed and the development of networks created exclusively with Open vSwitch and its tools would be an interesting work to do.

Bibliography:

[1] Does open vswitch support mpls?

https://github.com/openvswitch/ovs/blob/master/FAQ.md#q-does-open-vswitch-support-mpls.

[2] Open vswitch datapath developer documentation.
https://www.kernel.org/doc/Documentation/networking/openvswitch.txt

[3] What linux kernel versions does each open vswitch release work with?

https://github.com/openvswitch/ovs/blob/master/FAQ.md#q-what-linux-kernel-versions-does-each-open-vswitch-release-work-with.

[4] Open Networking Foundation. Software-defined networking: The new norm for networks. ONF White paper, 2012.

[5] Open Networking Foundation. Sdn architecture overview, 2013.

[6] Open Networking Foundation. Openflow switch specification. version 1.3.5 (protocol version 0x04), 2015.

[7] Sean Michael. Linux 3.19 release adds mpls support to openvswitch.

http://www.linuxplanet.com/news/linux-3.19-release-adds-mpls-support-to-openvswitch.html.

[8] Ryu project team. Ryu SDN Framework - English Edition. RYU project team, 2014.

[9] Open Networking Foundation. https://www.opennetworking.org/

[10] Open vSwitch Community. http://openvswitch.org/

[11] Ryu SDN framework web page. http://osrg.github.io/ryu/

[12] MPLS with OpenFlow: howto. *Università degli Studi Roma Tre.*

http://tocai.dia.uniroma3.it/compunet-wiki/index.php/MPLS_with_OpenFlow:_howto

Glossary

- SDN: Software Defined Network / Software Defined Networking
- OVS: Open vSwitch
- MPLS: Multi-Protocol Label Switching
- GPL: GNU Public License
- IP: Internet Protocol
- MAC: Media Access Control