FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

API design and implementation for management and configuration of SDN products

Rubens Jesus Alves Figueiredo

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Ana Cristina Costa Aguiar

December 7, 2017

Resumo

Abstract

...

Contents

List of Figures

List of Tables

Abreviaturas e Símbolos

SDN Software Defined Networks

Introduction

1.1 Context

1.2 Motivation

1.3 Goals

Berlin Institute for Software Defined networks

Software Defined Networking

Computer networking is a vital part of the services that are offered today, and as such, the performance in technology backing these services is central to the quality of these services. As the service providers reorganize their data centers in the cloud computing domain, enabling several improvements in the predictability, quality of service and ease of use of their services. New technologies are then required to make sure that their services are adapted to the fast changing landscape of networking services. One of the most notable innovations in this field is called Software Defined Networking, because its architecture allows for two essential features

- **Separation of network planes** SDN allows for the separation of the network control plane from the data forwarding plane by having network "intelligence" present in the network controllers, and having them control the forwarding elements that live in the Data Plane
- **Centralization of network management functions** By isolating the management on a separate plane, there is possibility of developing a single controller that can regulate the entire network, having unrestricted access to every element present in the network, simplifying management, monitoring, application of QoS policies, flow optimization, ...

In this chapter we explore the essential characteristics of SDN, the technologies that provide the back end for the development of this technology, and current implementations of the most popular SDN controllers, so that we can see the features that should be present while developing a management interface for SDN controllers.

3.1 Switches

3.1.1 OVS

3.2 OpenStack

3.3 OpenFlow

As the growth of the networking infrastructure of the past few decades became evident, the need for an enviroment that allows for experimentation and testing of different protocols and equipment became evident. If networking research would depend on the previously existing methods, then new ways of creating and developing protocols would become increasingly hard to implement and develop. As such, there was need for a framework that could enable testing of new ideas on close to realistic settings. So, on 28 February 2011 the version 1.1 of OpenFlow was released, and this proposal quickly became the standard for networking in a Software Defined Network. Since 2011, this protocol has suffered some revisions, and the latest version supported is version 1.5.1. Since this framework has evolved quite a bit, this section focuses on the versions 1.3, which are the versions that are used in development of this dissertation.

Several reasons led to the quick standardization of this protocol, which are related not only to the initial requirements of the platform, like the capability of supporting high-performance and low-cost implementations, and the capability of ensuring separation between production and testing traffic, but also the extensibility that the open source development model provides, removing the limitations that closed or commercial solutions give the network researchers.

The big advantage of OpenFlow is that it is, from the data forwarding plane point of view, easy to process. Since the control decisions are made by the controller, which lives in a separate plane, all the switch needs to do is correctly match the incoming packets, and forward them according to the rules established by the controller. The components that are part of this system and enable this functionality are:

- **FlowTables** This element describes the main component of the switching capabilities of the OpenFlow switch. Inside the switch there are several flow tables that can be used to match incoming packets, and process them in the rules that are specified by the controller. These rules can contain actions that affect the path of the packets, and these actions usually include forwarding to a port, packet modification, among others. Classification is done via matching one or more field present in the packet, for example the switch input port, the MAC and IP addresses, IP protocol, basically all information required to correctly process the incoming packet. The required actions for an OpenFlow switch are the capability of forwarding to a set of output ports, allowing the packet to move across the network; to send them to the controller, in the case of a miss of match; and finally the ability to drop packets, which is useful for DDoS mitigation, or more security concerns.

- **Secure Channel**
- **OpenFlow Protocol**

3.4 SDN Controllers

3.4.1 Floodlight

3.4.2 OpenDaylight

3.5 SDN Northbound

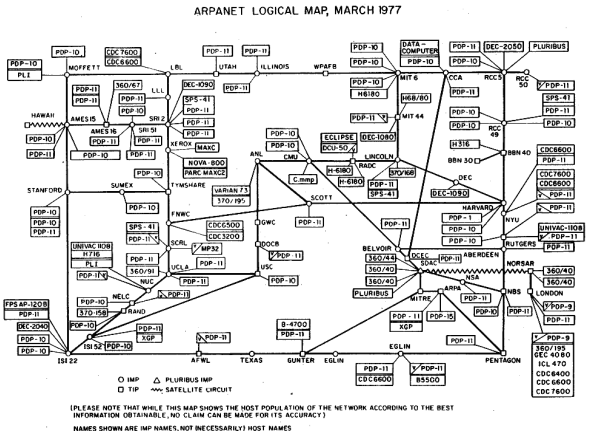
Literature Review

4.1 Computer Networking

4.1.1 Historical context

A computer network is a way to transfer digital information from point A to point B, via an established link between the two. In the early days, the demand to create an interconnected network of data sharing appeared from academic research and military needs, and since the introduction of these innovations, many American universities started to join in the this network, called ARPANET.

(a) Early ARPANET schematics, appr. 1969



(b) More sites connected to the ARPANET, September 1977

Figure 4.1: ARPANET evolution

As the advantages of having an interconnected network of computers became clearer, and with the surge of some others, such as CYCLADES, the french investigation research network, the need to connect the existing networks was rising, and that was one of the first steps of creating a global network, later known as the Internet. Some of the essential mechanisms that can still be found to this day were also developed in the ARPANET, like FTP and e-mail.

One of them was introduced in 1981, RFC 793 [**?**], and with it TCP was "invented". The main motivation for this development was the introduction of an end-to-end, connection oriented, and reliable protocol that allowed for the standardization of several different protocols. Also in this document, the definition of a OSI model, like the one that is prevalent today, or the definitions of reliability, are present, and continue to be relevant until today.

Figure 4.2: The current OSI model

One

4.1.2 Market data

By continuing to evolve and increase in both functionalities and users, the Internet as we know it is a global network, encompassing several protocols, and allows for instant communication of people around the world. A report indicating this evolution allows for some interesting conclusions about the state of the Internet market until 2021. This forecast was developed based on data originating from projections made from some Telecom and Media groups, direct data collection, and some estimates.

Global IP traffic As the report mentions, the monthly traffic, per capita, in 2016 is around 13 GB, and in 2021 is projected to be at 35 GB

Mobile devices traffic While today wired devices still make up for the majority of IP traffic, in 2021, traffic originating from Wi-Fi and mobile devices should account for 63 percent of the total traffic

Smartphone/ PC traffic By comparing the predicted evolution of smartphone/ pc traffic, the trends indicate that smartphone traffic should exceed fixed PC traffic.

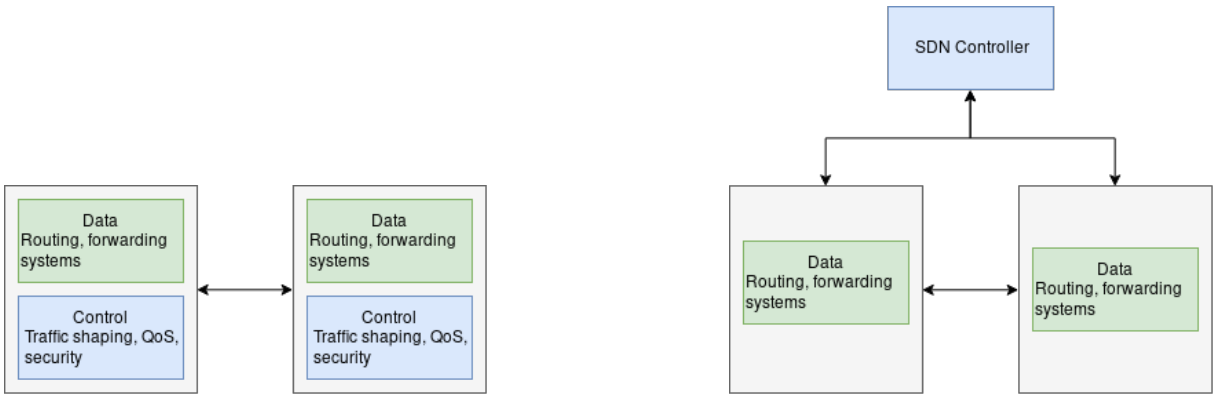
The previous points while obvious estimations, show a clear evolution in the way that Internet is usually accessed, and that is the

4.2 Software Defined Networking

As described in the previous section, there is a clear evolution of requirements, and this evolution was possible due to the adaptation of the exiting technologies to support better, and more efficient protocols that could carry the large amount of data that is transmitted every second. With that in mind, and in order to reduce costs to the service providers, simplify deployment and maintenance operations, developments in Software-Defined Networking (SDN) and Network Function Virtualization have been growing since 2010.

This new paradigm introduces programmability in the configuration and management of networks, by consolidating the control of network devices to a single central controller, achieving separation of the control and the data plane, and supporting a more dynamic and flexible infrastructure. Another important paradigm, that follows the development of SDN, is the concept of Network Function Virtualization. This concept allows to remove the amount of *middleboxes*[1], by replacing these with generic software applications.

The essence of SDN/NFV is described in a short manner if the Open Networking Foundation (ONF) paper: *In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications*. The following picture defines both of the approaches to network, the traditional one, where the data and control plane are just one, and the SDN way, that considers that application and control traffic should be considered in two different ways.



(a) Traditional networking architecture (b) SDN architecture

Figure 4.3: Traditional vs SDN network architecture

[1] Computer networking device that does some operations on traffic, excepting packet forwarding. Examples include caches, IDS's, NAT's, ..

One example of the operation of a switch in the SDN model is the following:

- A switch runs an agent, and this agent is connected to a controller;
- This controller runs software that can operate the network, managing flow control rules, and collecting information, enabling a full view of the network from a central node
- While this controller can be logically centralized, for fault-tolerance and high availability purposes it can be distributed, and by optimizing datamodels and providing caches, one of the earliest ONOS prototypes was able to achieve a distributed Network OS that could be applied to production networks [**?**]

More details on the technology that runs this model can be found in the next chapter.

After analysing some examples of deployment of the SDN model, the analysed literature provides some insights on the requirements that platforms using this paradigm need to obey [**?**].

- **High Performance**
- **High Availability**
- **Fault Tolerance**
- **Monitoring**
- **Programmability**
- **Security**

4.2.1 Why?

4.2.2 Challenges

4.2.3 Observed implementations in the industry

4.3 Cloud Computing

4.4 Databases

Technology Overview

API Design

Existing System