José Pedro Soares Castanheira

Software Defined Networking in access networks

Dissertação de Mestrado em Engenharia Eletrotécnica e de Computadores
07/2018

· U C ·

Universidade de Coimbra

Software Defined Networking in access networks

José Pedro Soares Castanheira

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Orientador:	Doutora Lúcia Maria dos Reis Albuquerque Martins
Co-Orientador:	André Domingos Brízido

Júri

Presidente:	Doutor António Paulo Mendes Breda Dias Coimbra
Orientador:	Doutora Lúcia Maria dos Reis Albuquerque Martins
Vogal:	Doutora Rita Cristina Girão Coelho da Silva

July 2018

Acknowledgments

Primeiramente, gostaria de agradecer à minha família, nomeadamente aos meus pais, irmãos e avós. Estou grato pela incessante presença e aconselhamento em todos os momentos da minha vida.

Aos meus orientadores da Altice Labs Aveiro, André Brízido e Vitor Mirones, pela ajuda, paciência e partilha de conhecimentos e experiências durante todo o tempo do projeto de dissertação.

À professora Lúcia Martins pela sua orientação e todo o tempo e apoio disponibilizado.

Aos meus colegas de curso, mais propriemente à minha SQUAD não só pelos momentos de convívio e diversão, mas também pelo espírito de auxílio mútuo ao longo destes 5 anos.

Aos meus amigos que me acompanharam e apoiaram durante esta etapa.

E a todos os que, de forma direta ou indireta, contribuíram para o meu processo de formação académica e pessoal.

A todos,

Muito Obrigado.

Abstract

Technology evolution over the past decade led to an increased complexity of Telecommunication Networks as well as lower life-cycles for the equipments involved. These problems, together with the high complexity associated with the configuration of equipments led to high capital expenditures and operational expenditures which forces network Service Providers (SP) to look for solutions. Software Defined Networking (SDN) and Network Function Virtualization (NFV) are two current paradigms used by SPs to to address these problems.

The Cloud Central Office (CloudCO) is a SDN/NFV architecture that can operate in a cloud environment. The Central Office Re-architected as a Datacenter (CORD) project is the reference implementation of CloudCO and has the Open Network Operating System (ONOS) as its SDN controller.

This dissertation proposes a SDN/NFV solution for passive optical access networks based on CloudCO, using ONOS for an Internet Protocol Television (IPTV) use case.

The IPTV scenario uses Multicast for IPTV distribution and needs Internet Group Management Protocol (IGMP) snooping to work properly. Because ONOS didn't have an IGMP snooping application, one was developed in the context of this dissertation. Also, the applicability of ONOS and the developed IGMP snooping application is shown through its performance evaluation for a set of Multicast traffic scenarios including the case of a single link failure, several high demanding topologies and heavy load multicast traffic conditions. Conclusions were drawn based on the IGMP packets processing times, the time it took for a host start to receive multicast traffic after sending an IGMP join packet and the compute power used.

Keywords

Software Defined Networking, Network Function Virtualization, Multicast, IGMP snooping, ONOS

Resumo

O progresso tecnológico que ocorreu na ultima década levou a um aumento na complexidade das redes de telecomunicações assim como a um tempo de vida útil baixo dos equipamentos envolvidos. Estes problemas, em conjunto com a alta complexidade existente na configuração de equipamentos levou a altos *CAPEX* e *OPEX* que forçaram as empresas fornecedoras de serviços de telecomunicações a procurar novas soluções. Dois dos paradigmas actualmente usados para fazer face a estes problemas são o *Software Defined Networking* (SDN) e *Network Function Virtualization* (NFV). O *Cloud Central Office (CloudCO)* é uma arquitetura SDN/NFV que opera num ambiente de *cloud*. O *Central Office Re-architected as a Datacenter (CORD)* é o projecto que deu origem à implementação de referência do *CloudCO* e tem o *Open Network Operating System (ONOS)* como controlador *SDN*.

Esta dissertação propõe uma solução *SDN/NFV* para as redes de acesso óticas passivas, baseada no *CloudCO*, usando o *ONOS* num caso de estudo com o *IPTV*. O IPTV usa Multicast como estratégia de difusão de televisão e precisa de IGMP *snooping* para funcionar devidamente. Como o ONOS não tinha nenhuma aplicação de *IGMP snooping*, foi desenvolvida uma no contexto desta dissertação.

A aplicabilidade do ONOS e da aplicação de IGMP *snooping* desenvolvida é mostrada através da avaliação de desempenho do sistema num leque variado de cenários Multicast, incluindo o cenário da falha de um *link*, de cenários com topologias complexas do ponto de vista da aplicação e condições de sobrecarga de tráfego Multicast. As conclusões foram tiradas tendo em conta o tempo de processamento de pacotes IGMP, o tempo que um utilizador demora a começar a receber tráfego Multicast depois de enviar o pacote IGMP para se juntar a um grupo e a capacidade de cálculo envolvida.

Palavras-Chave

Software Defined Networks, Network function virtualization, Multicast, IGMP snooping, ONOS

Table of contents

List of figures

List of tables

List of Acronyms

API Application Programming Interface

BNG Broadband Network Gateway

CLI Command Line Interface

CORD Central Office Re-architected as a Datacenter

CPE Consumer Premises Equipment

CloudCO Cloud Central Office

DDoS Distributed Denial-of-Service

FWD Forwarding

IGMP Internet Group Management Protocol

IP Internet Protocol

MFWD Multicast Forwarding

NB Northbound

NFV Network Function Virtualization

OLT Optical Line Terminal

ONOS Open Network Operating System

ONU Optical Network Unit

PIM Protocol Independent Multicast

PNF Physical Network Function

QoS Quality of Service

SB Southbound

SDN Software Defined networking

VNF Virtual Network Function

1

Introduction

In the past, networks had to be configured by hand in a tedious and expensive process prone to errors and as new technologies appeared the cost to adapt them into the core network was very high. Network service providers have been trying to reduce this kind of expense. To achieve this, various strategies have been attempted and one of the most recent ones is Software Defined Networking (SDN) and Network Function Virtualization (NFV). A network architecture design that uses concepts from both SDN, NFV and also cloud computing is the Cloud Central Office (CloudCO). It is defined by the Broad-Band Forum in the TR-384 [2]. It outlines how they would work together, which network functions should remain as Physical Network Functions (PNF) and which should be virtualized. One possible implementation for the CloudCO is the Central Office Re-architected as a Datacenter (CORD) project. This project is gaining traction between network service providers, and uses Open Network Operating System (ONOS) as the SDN controller.

For a SDN controller to be adopted by a network service provider, it has to support the legacy network functions. One of those functions is the Internet Group Management Protocol (IGMP) snooping. IGMP is a protocol used in networks that allows hosts to advertise which multicast group they want to join or leave. IGMP snooping is the process of listening to IGMP packets from the hosts to learn which host wants to subscribe/leave to a certain multicast group. This way, instead of broadcasting the traffic to every host, the switch is capable of only forwarding the packets to the hosts that want to receive them. In an SDN-NFV environment the IGMP snooping is performed through the SDN controller, which installs and removes the flows in accordance with the IGMP requests. Multicast traffic is used in IPTV, where users receive TV over an IP network. Because all the channels aren't being watched at the same time, IGMP snooping is required to know which channel is on and when zapping is done which multicast group must be left and which multicast group must be joined. With this mechanism, service providers can save bandwidth preventing the network to be overloaded with unwanted traffic. IGMP snooping play a huge role in access networks and is still missing in the ONOS controller application library.

1.1 Objectives and contributions

The objective of this dissertation is to study how ONOS SDN controller works and if it is possible to have a multicast scenario with the standard ONOS applications. As an IGMP snooping application was missing in the ONOS controller application library, one was developed and tested. The tests ranged from simple networks to complex ones, simulating real live conditions.

This master thesis was developed in DSR-Altice Labs Aveiro. From the development

and testing of the IGMP snooping application a demonstration paper was submitted for the "EEE Conference on Network Function Virtualization and Software Defined Networks 2018" which can be seen in appendix A.

1.2 Dissertation outline

This dissertation is composed by the following chapters.

- Chapter 2 introduces the concepts of SDN, NFV, their objectives and advantages and how they should interact. It also presents the CloudCO architecture, principles to choose which functions should be disaggregated and virtualized and some security concerns relative to these concepts and architectures.
- Chapter 3 presents the platforms and software used along this thesis as well as the motivation for choosing them.
- Chapter 4 introduces multicast and which protocols and network functions are necessary to have multicast running on a SDN/NFV environment.
- Chapter 5 describes the tests done using ONOS and the IGMP snooping application developed and presents the results obtained in the different scenarios.
- Chapter 6 presents the conclusions from this thesis and some future work directions are given.

2

Software Defined Networking and Network Function Virtualization

In traditional networks all the network planes are bundled together and are integrated over the same devices. This makes legacy networks highly decentralized. Over the years this kind of design has proven his benefits. Networks grew more resilient and overall network performance also increased [3]. However, this highly decentralized design presents some problems that become more notorious as networks grow and turn into more complex systems. Network configurations have to be done by hand in each device, which is an extremely time consuming task that often leads to configurations errors. Vendors have started to offer network proprietary management tools that change from device to device. Network operators are expected to master each one of the tools in order to manage a large range of devices. The highly decentralized design of the legacy networks further increases the difficulty to design, test and deploy new features as it would mean the replacement of millions of devices. [3, 4]. Capital expenditures (CAPEX) are the costs of in physical goods or services that are used for more than one year. Operational costs (OPEX) represents the costs of operation of devices. As it can be imagined, as networks become bigger and more complex their OPEX increases. Also, the need to buy new equipments leads to an increase in CAPEX. Motivated by all these problems new solutions have emerged. claiming that they can reduce CAPEX and OPEX for network service providers, with faster development and shorter implementation cycles, as well as an easier management of networks. These solutions are called Software Defined Networking (SDN) and Network Function Virtualization (NFV) [1, 3, 4].

This chapter examines the concepts of Software Defined Network and Network Function Virtualization, how the interaction between them is made, how that interaction leads to a new network architecture design, the CloudCO and some security concerns from the access network perspective.

2.1 What is Software Defined Networking?

Software Defined Networking refers to a new way of organizing the network functionalities. Instead of having the control plane and data plane embedded on the same device, they are separated allowing the possibility to have simple forwarding devices corresponding to the data plane and centralized controller corresponding to the control plane. Over time the definition has evolved from only being the network architecture where the control plane and data plane are decoupled from each other to a more complex one depending on who is defining it [3, 5, 6]. Some authors state that SDN is a network architecture where the forwarding decisions are not destination-based but flow-based, the control and data planes are separated, the control plane is on an external entity called controller and software applications run on top of the controller and manage the network [3]. For the Open

Network Foundation the definition is a network architecture where the control plane and the forwarding plane are physically separated and where a control plane serves several devices [7].

To achieve the separation of the control plane and data plane it is necessary to have a well-defined programming interface between the SDN controller and the forwarding devices. This programming interface is called southbound (SB) Application Programming Interface (API and for SDN the most important southbound API is, for now, Openflow. Note that a controller can have more than one SB API [3, 8].

A northbound (NB) API is what the SDN controller uses to communicate with higher-level applications. It offers an abstraction of the network details. Usually the NB API is a Representational State Transfer (REST) API that is used to communicate with higher-level applications and simplifies the underlying topology and protocols to enable a faster and more straightforward development of network applications. A REST API is an API that uses HTTP to get, edit and delete data. As it uses HTTP it can be used by almost every programing language. A SDN controller can have more than one defined NB API.

In figure 2.1 an example of how a SDN architecture would work is displayed. The centralized controller uses the SB API to communicate with the network devices and uses a NB API to communicate with the network applications [6].

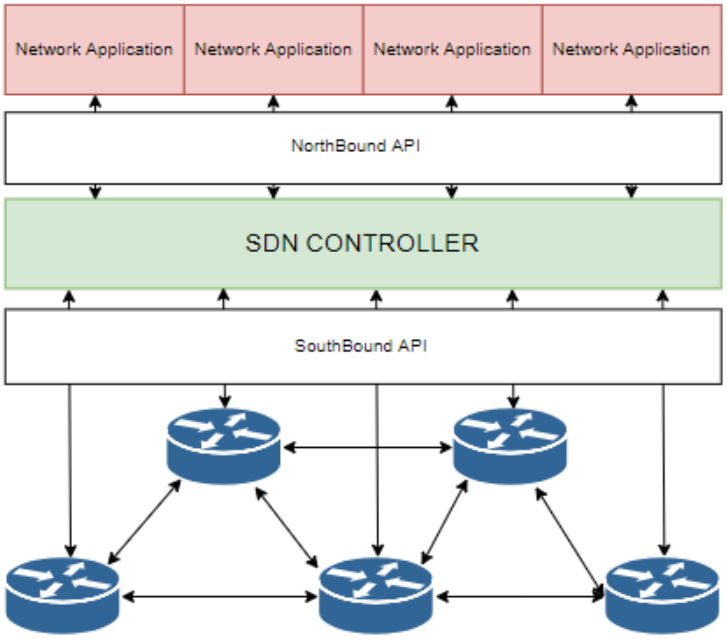

Figure 2.1: Example of a SDN architecture

SDN allows a faster development of new work methodologies as it becomes easier and quicker to implement new network norms and protocols. It also allows for a global

network view that all applications can use leading to more consistent and efficient policy decisions that could lead to an improvement of the network performance. With SDN the integration of different applications becomes easier as the order in which the applications run can be previously defined by the controller. For example, the income traffic can go through a firewall application, then an anti-virus application and in the end a parental control application. Having all these benefits into account, networks in the future should embrace innovation instead of trying to predict future requirements [3, 8].

2.2 What is Network Function Virtualization?

Traditional networks rely on specialized physical network devices to perform a given service. These network devices have a proprietary nature and most of the times have a specific chaining that must reflect the network topology. In addition, a chase for long product cycles and heavily dependent on specialized hardware reflected the lack of capacity to adapt and innovate in a fast paced and always changing world.

NFV (Network Function Virtualization) is one of the proposed solutions to these problems. It proposes to transform network architectures by running networks functions in common off-the-shelf hardware benefiting from the evolution in virtual infrastructure management. This allows for the concentration of many network equipments in server-like equipments that can be located along the network.

The shift from physical network functions to software functions hosted by data centers will enable service providers to reduce the time to market of new functions as it is easier to develop and deploy software components than to develop and deploy hardware components. It will also allow a full automated management via API's for services like dynamic QoS (Quality of Service), service load dynamic adaptation and service healing. New business will appear to service providers as networks can be shared as well as network functions [1, 9, 10].

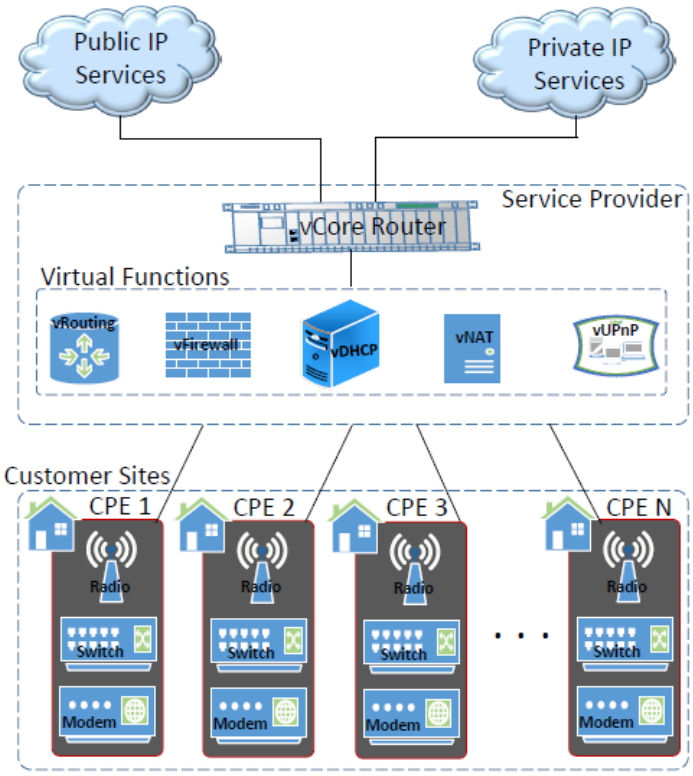

Figure 2.2: Example of a NFV solution architecture [1]

Figure 2.2 shows how an NFV architecture would work to simplify the Consumer Premises Equipment(CPE). Various CPEs would be able to use the same functions or network functions and update one of them would be fast and easy. In this case functions like Dynamic Host Configuration Protocol (DHCP), firewall and routing moved way from the CPE and were implemented in an aggregation point for various networks. Figure 2.2 also shows how this would be a cheaper solution by having simpler equipments on the CPE side, and more efficient virtualized equipments on the service provider side.

2.3 How do they interact

SDN and NFV are two different concepts designed to address different problems in the shift for a new network paradigm. While NFV concept is implementing network functions on common hardware instead of specialized hardware, SDN priority is to concentrate the processing capacities in a central equipment. Even though SDN and NFV can exist separately, the evolution of these concepts is highly complementary. NFV serves SDN by providing the capability to virtualize the SDN controller together with some network elements functions to run on a cloud-like environment allowing the cloud management techniques to be used, e.g. scalability techniques, migration techniques and configuration

techniques. SDN serves NFV by providing an API between the network and the VNFs (Virtual Network Function) to reach an optimized traffic routing and network resources usage.

Access networks are one of the many beneficiaries from this new paradigm shift, as now we are able to offer new business models where operators can offer Anything-as-a-Service to third parties as well as reduce CAPEX and OPEX and the time-to-market. [1, 9, 10].

2.4 Cloud Central Office

A CloudCO (Cloud Central Office) is an adaptation of the traditional Central Office infrastructures that use SDN, NFV and Cloud technologies and is defined by the Broad Band Forum. The CloudCO functionalities can be used by others, like third parties and operators, through a NB API. A NB API is a way to communicate with higher level components or interfaces with a set of well defined functions. Using a NB API allows the CloudCO functionalities to be used by other systems without revealing to them how they were implemented. To do this, SDN and NFV techniques are used on typical data center equipments. The CloudCO architecture has a series of advantages like being able to have a higher level of control on system-level network and service design, in part because of the disaggregation of the legacy network nodes into separated network functions, being more flexible and easy to scale due to the use of virtual network functions, supporting automated and fast deployment and allowing an easy addiction and maintenance of services. It does not take much hardware to implement a CloudCO. It is only necessary some general purpose network switches that interconnect over a switch fabric, some computer hosts and access I/O hardware connected to the switches and some connection to the service provider backbone. Most of this hardware may be installed inside a central office but the CloudCO domain can reach over multiple physical locations. In theory we could have some necessary hardware installed across multiple locations resulting in a single instance of a CloudCO domain or there can even be network functions that normally are at the customer premise disaggregated across multiple locations [2].

In order to move to this new architecture we need to choose the functionalities that should be disaggregated, virtualized or stay as they are now. Disaggregation is the division into constituent parts. Applying that concept to networks, the network disaggregation can be seen as the division of nodal functions into network functions that are more modular and granular. A simple and obvious example is the network control plane on access nodes that can be centralized into an SDN application. In this case virtualization of network functions is the passage of network functions to software that may be hosted in generic

off-the-shelf hardware rather than having them running on dedicated hardware. To choose which functionalities may be disaggregated it is needed to pay attention to the following criteria:

1. Ability to virtualize- Disaggregation is one of the most important means to split functions that can be virtualized from others that cannot.

2. Ability to efficiently split the user plane and the control plane- As control plane functions are compute-intensive and not forward-intensive, they benefit from virtualization and should be separated from the user plane functions that require a fast processing of packets and benefit from specialized hardware.

3. Reusability- It may be advantageous to disaggregate components that can be reused at various physical or logical points in the network.

4. Upgrade cycle- It is easier to make changes to a disaggregated and virtualized function than to one built in hardware.

5. Interface simplifications and standardization- Disaggregation should lower the overall interface and API complexity.

6. Performance- In order to better optimize some components, it may be more suitable to disaggregate them.

7. Operational consideration - A few operational considerations need to be taken into account:

   (a) Orchestration- in general the complexity of orchestration is likely to grow if the orchestrated service requires a larger number of network functions.

   (b) Availability – A higher level of availability can be expected as strategies like reboot of a function, hot swap and redundancy are more efficient with smaller functions.

   (c) Diagnostics – Having more components and an increased number of network functions lead to a rise in the difficulty to diagnose a problem.

In order to choose which network functionalities should be virtualized, we should take the following characteristics into account:

1. High rate of change- It is easier to alter a function when it is virtualized and not on specific hardware.

2. Varying scale- When a function does not have a constant resource usage over time, it can benefit from cloud scale out techniques

3. Differentiation- Functions that can have different implementations between different vendors.

4. Specialization- Functions that require high level of domain information.

5. Interoperability- Functions that can alleviate interoperability by having a centralized common implementation to reduce variability across multiple vendors.

6. Component reuse- Functions that once virtualized can be used in different configurations to serve different needs.

It should be also taken into account the complexity, risk and performance impact of the transition. Functions that are performance intensive, real time sensitive, or already deployed in the equipment should remain as physical network function. The tradeoffs between virtual network functions and physical network function may change over time as new technologies appear and their performance improves. The CloudCO should be able to adapt to these evolutions [2].

2.4.1 CloudCO Domain Architecture

The CloudCO reference architecture is composed of a NFV and SDN architecture applied over NFV infrastructure and a physical one. The NFV component is responsible for the virtual functions and their supporting infrastructure. The SDN component is responsible not only by the control and management of most of the user plane interactions between Physical Network Functions (PNFs), the switch fabric and VNFs but also by the redirection of certain control packets to the right SDN applications. The CloudCO domain orchestrator is the central function in the architecture and also contains the CloudCO NB API. It is responsible for providing the necessary service abstraction layer, not showing the internal procedures of the CloudCO. Figure 2.3 shows the CloudCO reference architecture.

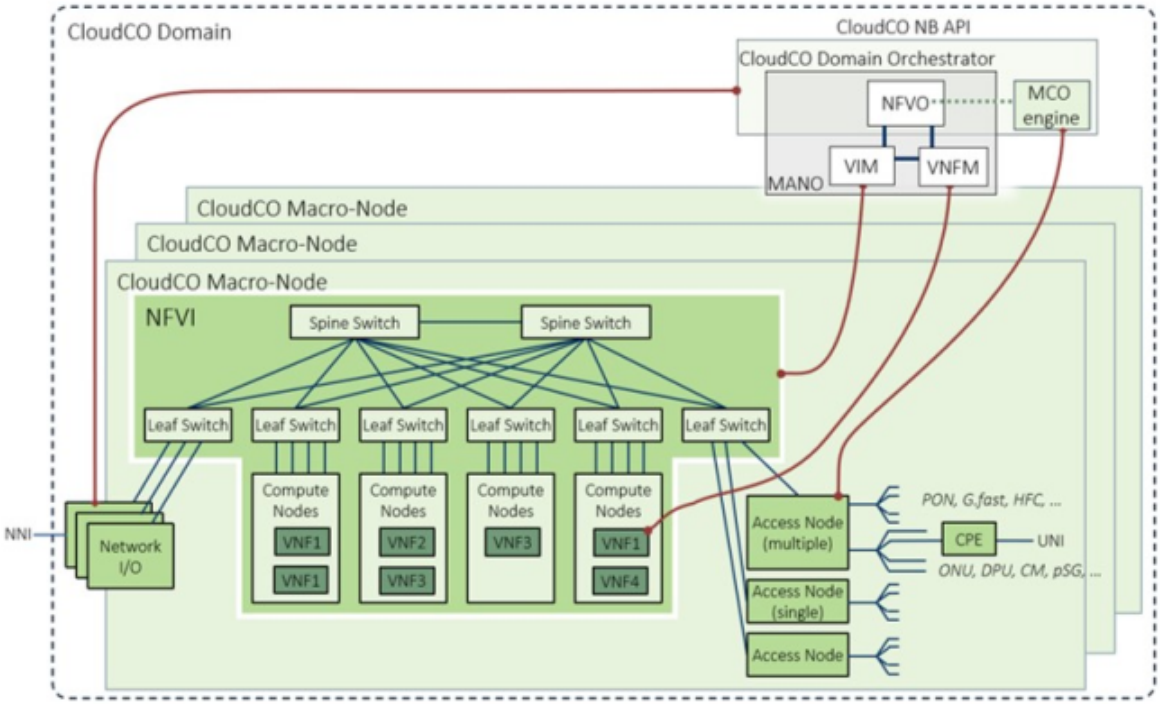

Figure 2.3: CloudCO reference architecture [2]

2.5 Security concerns

Network security is a component that takes a massive part of the cyber security and becomes more important each day that passes. Traditional networks security uses firewalls and proxy servers to protect the physical network. The appearance of new concepts like SDN and NFV brought new technologies that can solve many security and privacy problems but their use can also cause new problems that need to be addressed. CloudCO not only provides a separation between the assets and the threat making the network more secure, but also gives the operator a dynamic control over their network service behavior with the use of API's. Some new security problems can appear from the communications of the different planes and some can appear from a multi-tenant CloudCO. Because the underlying behavior of a virtualized system, there may be security exploits that can be abused if they are poorly designed. It shouldn't be forgotten how some security problems may occur if someone can access the physical network [2, 11].

The existence of a central controller that uses the knowledge of the complete network to analyze traffic patterns for potential security threats is a reality. Common attacks like low-rate burst attacks and DDoS (Distributed Denial-of-Service) attacks can be recognized just by analyzing traffic patterns. Because of the increased performance and programmability of SDN and how the network view works, it is predicted that the discovery,

control and containment of security threats is going to be faster and more efficient. The SDN interface can also facilitate a higheo number of potential attacks like DDoS attacks as a result of having a centralized controller. Trust issues between network elements due to the open programmability of the network and the absence of good proceedings regarding SDN functions and components may also occur [8, 12].

3

Choosing the platforms to work with

In this chapter the platforms and programs that were used in this dissertation are introduced, as well as the justifications for using them. Some of the main features of these platforms are also described.

3.1 CORD

3.1.1 What is CORD?

CORD means Central Office Re-architected as a Datacenter and tries to unify the SDN, NFV and cloud technologies. It can be seen as an implementation of the CloudCO, so in terms of architecture they are similar [13]. The software used on CORD reference implementation takes advantage of four open source projects. ONOS is a SDN controller built thinking on service providers and allowing high availability and scale-out capacity to the network. It provides the control plane to a network, a platform to host control programs that give CORD services and is capable to manage switches and other network components [13, 14]. Docker is a tool that allows applications to share the same Linux kernel, making a container for an application. On CORD it provides a way to deploy and interconnect services on a container and also plays a role on the deployment of the other management elements of CORD instantiated on Docker containers [13, 15]. XOS is a framework for assembling and composing services. It consolidates infrastructure services, control plane services and cloud or data plane services [13, 16]. OpenStack is a cloud OS that is able to control a large number of compute, storage and networking resources all over a datacenter. On CORD it is responsible for provisioning and creating virtual networks and machines [13, 17].

For service providers CORD is seen as a more mature SDN solution than the others available in the market and has huge support from other service providers like AT&T, China Unicom, Comcast, Google, Deutsche Telekom, Telefonica, NTT Group, and Turk Telekom. Recently CORD launched blue-prints for service providers start to deploy field trials and presented use cases from AT&T, Telefonica and NTT. CORD and ONOS are still under development and new features are added in each release depending on market trends and the results from the field trials.

In this dissertation, only the ONOS SDN controller from CORD project was used. The remaining projects were neither included nor discussed for the rest of this document.

3.1.2 Disaggregation and Virtualizing of network components

Reducing some of the principal components of the network to a basic status is possible if the functions that need to remain on hardware and the ones that can be virtualized are

previously identified. The virtualization of the OLT (Optical Line Terminator) results on a control program called vOLT that runs on top of ONOS and implements all the functionalities of the legacy OLT. The same thing can be done to the CPE resulting in a virtual Subscriber Gateway that runs a bundle of functions selected by the subscriber on the central office white-boxes. A CPE is still present in the house of the user but it can be a strip down version of the legacy one. Virtualizing the BNG (Broadband Network Gateway) is also possible and on CORD it is called vRouter and is implemented as a control program hosted by ONOS that manages flows through the switch fabric on behalf of subscribers.

3.2 ONOS

ONOS is the OpenNetwork Operating System and was created by the Open Networking foundation. It is a SDN controller designed for service providers and is known for its high availability and resiliency, a requirement from service providers so that clients don't experience poor quality of service. It was also designed to be able to grant the highest performance possible for scaled network operations. The controller can be deployed on a cluster of servers that run the same ONOS software enabling a fast recovery in case of server failure on one instance of ONOS [18, 19].

In this dissertation it was decided to use ONOS as a SDN controller because of the tight relation it has with CORD. Also, ONOS presents good architectural documentation and has an environment to start to work and develop network applications.

3.2.1 ONOS features

ONOS relies on applications to perform network functions. To communicate with them it uses NB APIs. ONOS has a REST API and a Java API. To the network manager it has available a command line interface as well as a graphical interface. SB APIs are used for communication with a variety of net devices. ONOS allows the installation of more SB APIs in order to adapt to new requisites from service providers. Right now it supports Netconf, Openflow, SNMP and TL1.

One of the main features of ONOS is the use of intents. The use of intents allows applications to state their network control desires in form of a policy rather than a mechanism. ONOS core accepts the intent specifications and compiles it into installable intents that then result in tunnel links being provisioned, flows rules installed or wavelengths being reserved.

New features can be added as applications. Right now ONOS has more than 100 applications like reactive forwarding, DHCP, alarm applications and switch drivers [18].

3.3 VLC

VLC is a free, open source and light-weight media player that allows for a large set of codecs and streaming protocols to be used as well as to display media content [20].

VLC was chosen for its good documentation, for providing a command line interface to start streaming and to receive a stream, and for its capability to send and receive multicast traffic.

3.4 Mininet

Mininet is a network emulator that allows to instantiate virtualized network elements like switches, routers, links and even hosts in a single computer. It allows a fast prototyping of large networks and is developed to connect these network devices to a SDN controller [21].

Mininet was chosen for its flexibility and scalability because it allows different kinds of small or large networks behaving like real networks.

3.5 Open vSwitch

Open vSwitch (OvS) is the standard switch instantiated by the Mininet. It supports OpenFlow [22]. It was chosen because it runs with little resources, is very reliable and its the most common switch used in a virtualized environment.

4

Multicast

In a network we can have three types of traffic: Unicast, Broadcast and Multicast. Unicast is used when we need to send traffic from one source to one receiver. Broadcast is used when we need to send traffic from one source to all the receivers. Multicast is used when we need to send traffic from one source to a group of receivers.

In this chapter an easy example is shown to explain the differences between them, and how they affect a network differently. It is also described which protocols does Multicast use and which network functions are necessary to have multicast traffic flowing in a network. For last, a Multicast use case is presented, IPTV.

Throughout this dissertation the term "switch" will be used to talk about ethernet switches and "router" is used to talk about IP routers.

4.1 What is Unicast, Broadcast and Multicast?

Imagine that a server is streaming a movie and 4 hosts want to receive it. If the sender uses Unicast, each host needs to establish a connection with the server and the movie is sent 4 times, one for each host. As we can see this is not a scalable solution: if the numbers of hosts grows, the number of direct connections will also grow saturating the network with copies of the same movie [23].

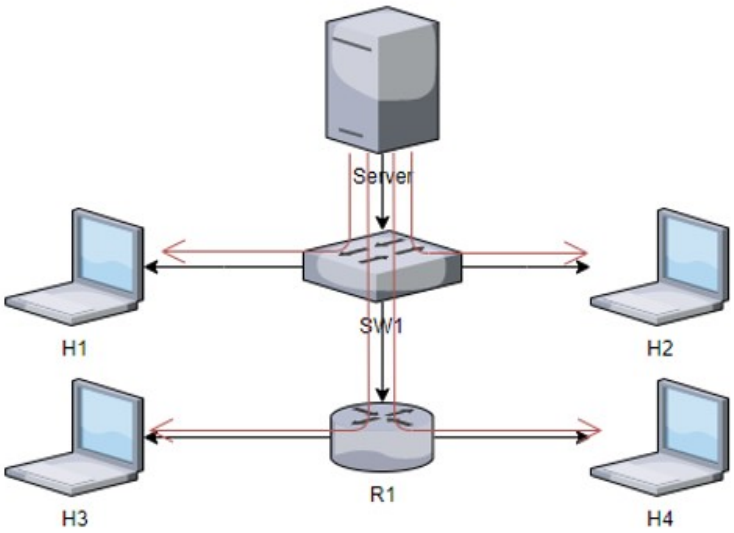

Figure 4.1: Example of a network where a server uses Unicast to stream a movie

In figure 4.1 it can be seen how a Unicast works in a small network with a ethernet switch (rectangular box), a IP router (circular box) and 4 hosts. Each host has a direct connection with a server so four copies of the same traffic are being sent at the same time.

With Broadcast other problems arise. The sender streams to everybody, even hosts

that don't want to receive it or that shouldn't have access to it. So it is not a good option in this case.

Imagine that only two hosts want to receive a movie stream. With Multicast this solution is very efficient as the server will only send the packets once and the switches or routers are responsible for forwarding and copying when needed the packets to the hosts that want to receive them. Figure 4.2 demonstrates how Multicast works and how efficient it is [23].The two hosts that subscribed to the Multicast group receive the traffic without problems and the remaining elements of the network don't receive it. In the Multicast case the router is able to forward the income traffic contrary to the broadcast case.

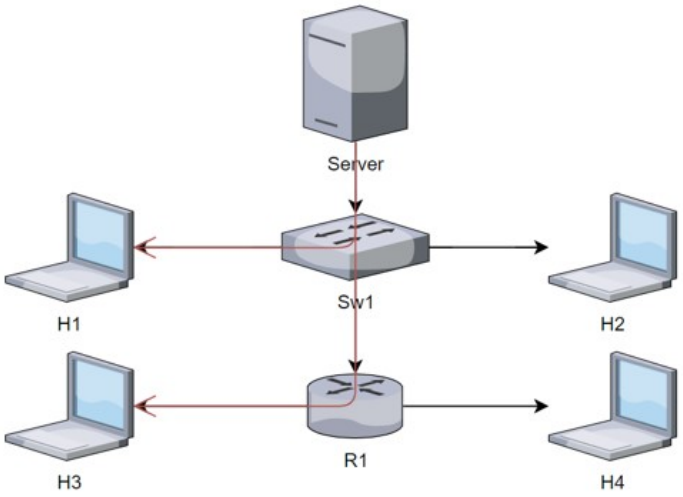

Figure 4.2: Example of a network where a server uses Multicast to stream a movie

Although Multicast is much more efficient, it does not work without some other requirements and some configurations. The hosts need to be able to tell the router when they want to receive Multicast traffic. IGMP (Internet Group Management Protocol) is used for that. Switches also need to know what to do when they receive Multicast traffic because for the incoming traffic the destination is not a host but an IP group. We use IGMP snooping to address this problem, as explained next. The switch will listen to IGMP messages from the hosts and figure out where to forward the packets. It is also necessary a Multicast routing protocol to know where to forward traffic between routers.PIM (Protocol Independent Multicast) is one of the possible and most popular choices to do it [23].

4.2 How does Multicast work

Multicast applications use a specific group of IP addresses that are reserved. These IP addresses range from 224.0.0.0 to 239.255.255.255 and each one correspond to a different multicast group that a host can subscribe. Some of them are reserved addresses with specific tasks, for example 224.0.0.0 to 224.0.1.225 are reserved to permanent Multicast groups, 232.0.0.0 to 232.255.255.255 are reserved to source-specific Multicast groups. To subscribe a Multicast group a host uses the IGMP protocol. In order to forward the Multicast traffic through the right port, the switch resorts to IGMP snooping, i.e., the switch will listen to the communication between the host and the router to know which hosts subscribe to which Multicast group. Routers use diverse protocols to ask for Multicast traffic from other routers. The most widely used is PIM [23].

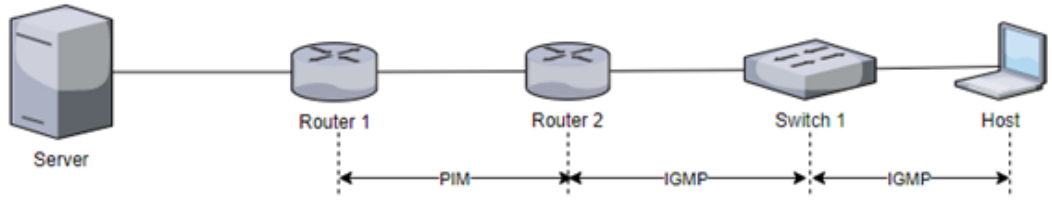

Figure 4.3: Reach of the protocols used for Multicast

In the figure 4.3 it is represented the scope of each protocol. IGMP is used between the the host and the first local router and PIM is used to announce Multicast groups between routers. A deeper definition of the protocols and how they work follows next.

4.3 IGMP

There are different versions of IGMP and different kinds of packets. Each version increases the complexity of IGMP but also increases its functionalities.

IGMPv1 is the first version and only has 2 kinds of messages, membership reports to inform the local router that a host wants to subscribe to a specific Multicast group and membership queries that the router periodically sends to confirm that a host still wants to receive the Multicast traffic from that group. If the router doesn't receive a membership report from a host after sending a membership query it knows that the host doesn't want to continue to receive the Multicast stream. The lack of a mechanism to leave a Multicast group from the host side is one of the main flaws of the IGMPv1 [23].

IGMPv2 is a more sophisticated version of IGMPv1. New kinds of messages appeared in this version like the message that allows to leave a Multicast group from the host side.

This saves huge amounts of bandwidth as a host doesn't need to wait for a timeout to leave a group. There was also the addition of the Maximum Response Time Field; this is the field in the membership query that defines how much time hosts have to respond to the query. This field is extremely important to don't congest the network with an immediate response to the query. Now hosts have a dynamic time frame to respond and choose a random value from 0 to 'Maximum Response Time'. The IGMPv2 also allows the membership query to be group specific and not for all hosts [23, 24].

IGMPv3 introduced the possibility to subscribe to a specific source Multicast group. It can be seen as a more secure and robust protocol and necessary for specific source Multicast implementations. It also allows to subscribe or unsubscribe to various groups in the same packet which reduce the overall number of packets sent from the hosts [23,25].

4.4 IGMP Snooping

IGMP snooping is done by the switches and it is the process of listening to IGMP packets either from routers or from the hosts to learn informations about Multicast groups. It is possible to know which subnet wants to listen to and instead of flooding the network with a Multicast stream, the switch is capable to forward the packets only to the hosts that desire to receive the stream. The switch needs to be able to listen to the different kinds of IGMP messages, distinguish them and do the right processing [23, 26].

IGMP Snooping is one of the main functionalities in the tradictional OLTs (Optical Line Terminator). It allows to optimize the bandwidth in each PON by only sending Multicast traffic when it is necessary and for where it is necessary. In a typical service provider scenario there are hundreds of channels and in a single PON there can be up to 128 ONUs (Optical Network Unit) .

4.5 IGMP Query

In order to better manage a Multicast network, it is necessary a system that prevents the waste of resources, in this case the waste of bandwidth with hosts that do not wish to continue subscribed to a Multicast stream. Usually when the host wants to leave a Multicast group it sends an IGMP packet saying so, but suppose that a device is disconnected from a power source and does not send the leave packet. There should be a timeout mechanism in place. In legacy networks a switch with the role of querier sends a packet periodically, an IGMP membership query, to all the hosts in the network and waits for their response. If there are no responses, i.e. membership reports, from the hosts in a determined amount of time they are removed from the Multicast group. The hosts should

respond with every Multicast group they are subscribed to.

4.6 PIM

PIM is the most used Multicast routing protocol. It was developed to route Multicast traffic without needing to rely on specific Unicast routing protocols. PIM has different modes but the most common ones are Dense mode and Sparse mode. Dense mode assumes that every router wants to get Multicast traffic so it forwards the Multicast packets through all interfaces until a message to stop (prunning message) is received. The prune messages only last 3 minutes. After that the router receives the Multicast traffic again and if it doesn't want the Multicast traffic, a new prune message is sent. Sparse mode assumes that no other router desires to get Multicast traffic, and only forwards Multicast data through interfaces that have received explicit join messages. The number of routers involved in handling the Multicast traffic is minimum [23, 27].

4.7 IPTV

Service providers with IPTV services are among the main beneficiaries of multicast and IGMP snooping. The load on their network is heavily reduced allowing for other network services. In an IPTV network each channel correspond to a Multicast group and changing channels correspond to joining a new group and leaving another. On IPTV the ideal time for a user to start watching a new channel is less than 200 ms and the acceptable time is less than one second [28, 29].

5

Using ONOS

In this chapter some experiments with ONOS are firstly presented. As there is no IGMP snooping application on ONOS, one was developed as described in this chapter. Finally, the performance of the overall solution is evaluated.

5.1 Working with ONOS

5.1.1 Basic setup

During all basic tests the topology is the following:

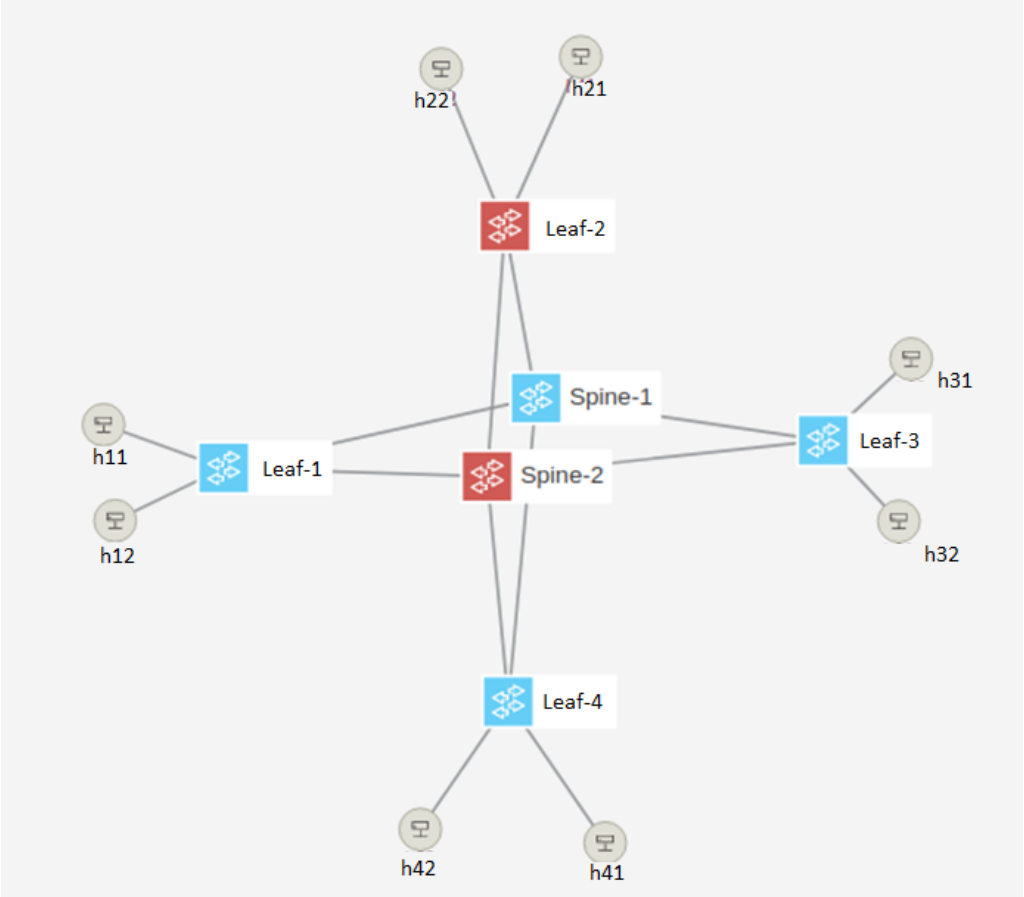

Figure 5.1: Topology used in all basic tests

It contains eight hosts; every two hosts are connected to a single switch, called leaf; that switch connects to two central switches, called spines. The spines connect with all leaf switches. Note that all the switches used are layer two switches.

The ONOS version used was the ONOS 1.13.0- snapshot, the mininet used was 2.3.0 and the VLC version was 2.2.2 .

5.1.2 Sending unicast data

The objective of this experiment is to have a network where an IPTV source is sending a movie to the receivers in Unicast mode. In this case the sender is h11 and the receivers are h21, h22, h31 and h41.The topology used is the one presented in figure 5.1.

The VLC commands used were:

Sender:

```
vlc-wrapper -vvv '<FileName.xyz>' '#std{ access= udp, mux= ts, dst=
<Destination IP>, port= 1234 }'
```

Receiver:

```
vlc-wrapper -vvv udp://
```

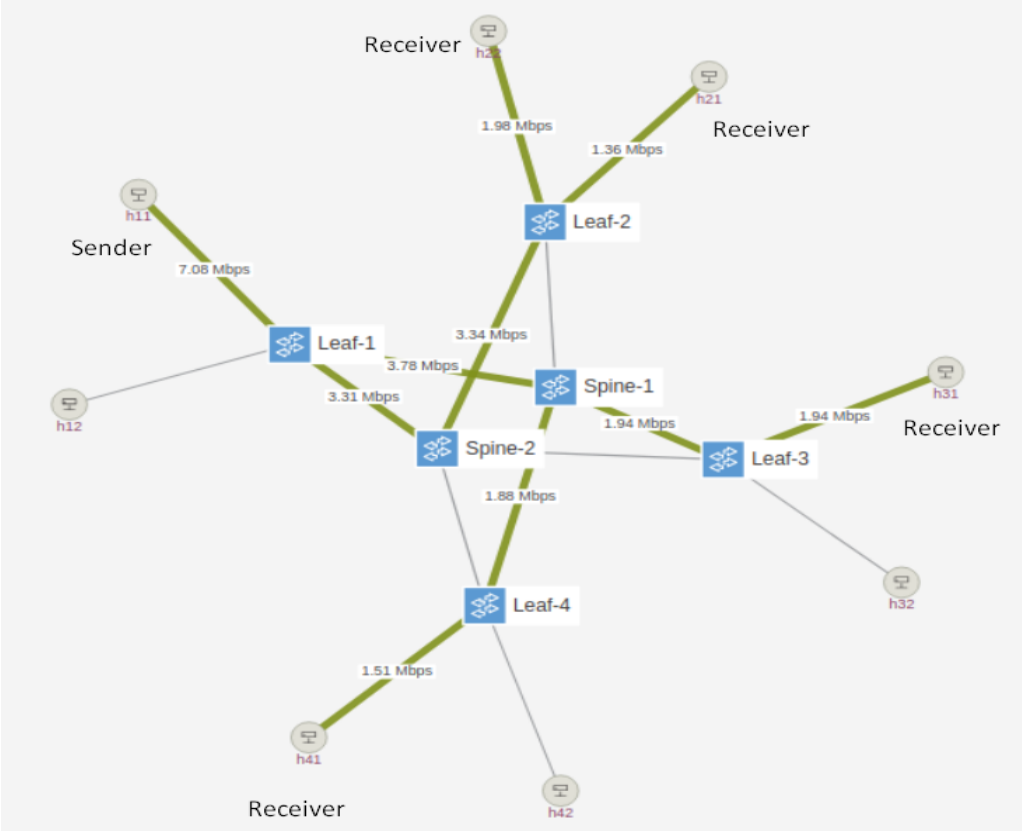

Figure 5.2: Result of sending via unicast

Figure 5.2 displays the network status of four unicast streams, using only the apps that come activated by default with ONOS and activating the forwarding app. The results are clear: when h11 streams to four hosts the throughput scales as the number of hosts increases, so four hosts require four times more bandwidth than one host. In the end h11 had a throughput of 7.08Mbps $\approx$ 3.7808Mbps + 3.3108Mbps. It is also interesting to see how well the reactive forwarding app worked, providing a fast path between the hosts

and installing the necessary flows on the switches. As it was stated before, unicast is not the appropriated method to stream between hosts as it is data intensive. The solution is using multicast but its implementation is not as easy as unicast. The following sections explain the problems, what ONOS already have available to use and how the problems were overcome.

5.1.3 Sending multicast data using ONOS standard applications

5.1.3.A Using the FWD application

Using the same topology and with the same applications installed, a test was made with multicast traffic. In this case h11 streams to the multicast group with IP address 232.0.0.0 and h21, h22, h31 and h41 subscribe this multicast group. We used both specific source multicast and any source multicast for the receivers. After activating the FWD (forwarding) application, it installs two flows in each switch. These low priority flows state that every ARP packet and every IPv4 packet should be sent to the controller if there isn't other higher priority flow to deal with them.

The first Problem encountered was that the hosts didn't send any data. The problem was found to be that mininet hosts didn't have any default multicast gateway connected to their newly created interfaces. To add it to them it was used the command: `<NameOfTheHost> route add -net 224.0.0.0 netmask 224.0.0.0 <HostInterface>`

After adding the right multicast gateways, the test was made as stated before and the result of this test was that instead of forwarding the packets to the subscribed hosts the FWD application broadcasted them, i.e. sent the multicast traffic to all hosts presented in the network. This behavior was the expected one for a network without IGMP snooping [26].

5.1.3.B Using the MFWD application

For the second test the MFWD (multicast forwarding) application was activated and the option ignoreIPv4Multicast was set to true on the FWD application so that it didn't interfere with the multicast packet processing. The setup for this test was the same as the one from the first test. Right after its activation, the MFWD application adds a low priority flow to every switch that states that if a IPv4 packets arrives with a destination 224.0.0.0/4 it should be sent to the controller.

The firsts attempts of this test showed similar results as test one, described in 5.1.3.A, which theoretically didn't make sense as the FWD application should ignore multicast traffic. After trying the first test but with this option (ignoreIPv4Multicast) set to true we were sure that the error was in the FWD application because it did not ignore the multicast

packets. After looking on the application code a bug was found. This bug was that the application didn't update their settings even after changing them over the CLI (Command Line Interface). This bug was corrected and the second test repeated.

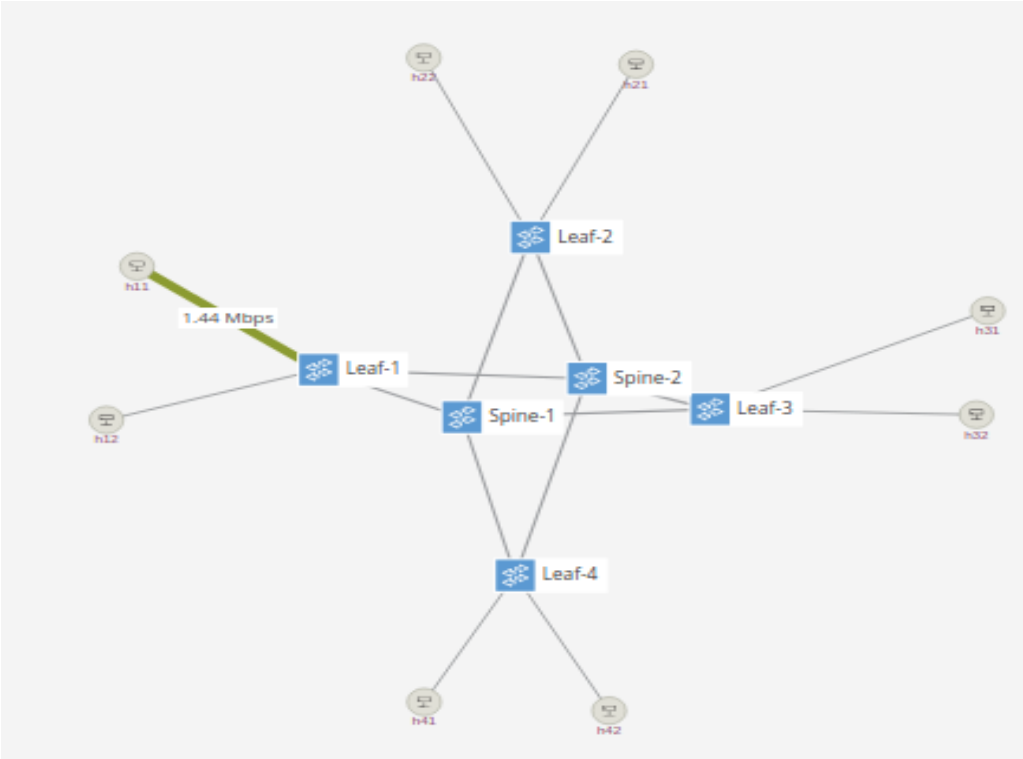

Figure 5.3: Result of the second test

Figure 5.3 shows the results of the second test. It is clear that the hosts are not receiving the packets from the multicast group they are subscribed to. After analyzing the flows installed on the switches we saw that no new flows were installed, there where also no new intents installed but we saw that a multicast route was created with the IP address 232.0.0.0. and with h11 as the sender. ONOS multicast routes are organized on a table that stores the multicast group IP, who is the sender to that group and who are the sinks. It can be said that the MFWD application is not working as expected. Even though it generates the route to the sender it doesn't add the hosts that subscribe to the group as sinks of the route.

There was no documentation on how it was supposed to work, so the next step was to analyze the code of the MFWD and the ONOS core. The analysis of the MFWD application code allowed to conclude that IGMP packets were not processed. In order to be able to do an IPTV use case, it is imperative to have an IGMP packet processor.

5.2 Developing the IGMP application

Because ONOS doesn't have a native IGMP snooping application one was developed following the IGMPv2, IGMPv3, IGMP snooping and IGMP querier specifications. The main objective of the IGMP snooping application is to perform the tasks of both the IGMP snooping and IGMP querier. Without it there was no automatic way to add hosts to multicast groups and to manage them afterwards.

The application starts by installing a flow rule on all network switches and when they receive a packet which matches the flow specifications the controller processes it. In this case the flow selector is a IPv4, IGMP packet with the destinations 224.0.0.0/4. The IGMP snooping app, working within the controller, then receives the IGMP packet and after identifying the IGMP version and type processes it according to the specification.

IGMPv3 membership reports allow for more than one action to be transmitted over a single IGMP packet so the application separates the various groups and then processes each one individually. After gathering the necessary information like the group IP address, if it is source-specific or not and if the host is joining a group or leaving one, the application searches in the multicast route table for routes with the wanted IP group address then it searches for the one with the specific sender, in the case there is one, and at last adds the IGMP sender as a sink for the route.

When an IGMPv2 membership report is received there is only one possible action, subscribing to a multicast group, so the process is simpler. Because IGMPv2 is not source specific the IGMP application only has to find the registered routes with the desired group IP address and add the sender as a sink to all the routes.

The IGMPv2 Leave group packet as the name denotes provides a mean to unsubscribe multicast groups and has a similar behavior as the IGMPv2 membership report processing, but in this case instead of adding a new host to the group the host is removed.

IGMPv1 only has a subscribing mechanism and behaves like the IGMPv2 membership report.

In this case the role of querier is played by the controller itself, more precisely the IGMP snooping application within ONOS. The querier has two tasks, one is to periodically send an IGMP query to every host, the other is to verify the timeout condition of each route for every host. ONOS waits a certain period of time before removing the host from the multicast group. In this case the waiting time is 3 times the querying time, i. e., each time it send an IGMP query it decrements an individual counter for each host and if that counter reaches 0 the host is removed. Every time a host responds to the query that counter is reset. The number of times it waits for the timeout can be adjusted depending on the type of network we are working on.

5.3 Testing the IGMP snooping functionality

5.3.1 One sender and multiple receivers

To test the IGMP snooping functionality an experiment was made that used the FWD application to forward the normal traffic, the MFWD to create the multicast routes and the developed IGMP application to do IGMP packets processing. Using the same topology as before, figure 5.1, and the same scenario, h11 is streaming to the multicast group 225.0.0.0. and hosts h21, h31 and h32 join that multicast group. Both any-source multicast and source-specific multicast were tested but as they presented similar results no distinction between them was made for the rest of this dissertation.

In the beginning h11 starts to stream to the multicast group 225.0.0.0, but because no host is interested in receiving the stream the switch leaf-1 does not forward the traffic. The MFWD application creates a entry for a multicast group with h11 as the sender and 225.0.0.0 as the destination IP address but without any sinks. After hosts h21, h31 and h32 run the VLC command to subscribe to the multicast group with IP address 225.0.0.0, an IGMP membership report is sent and processed by the IGMP snooping application. They are then added to the multicast route as sinks. The MFWD application knowing that a sink was added to a multicast route declares the necessary intents that will install the necessary flows on the switches.

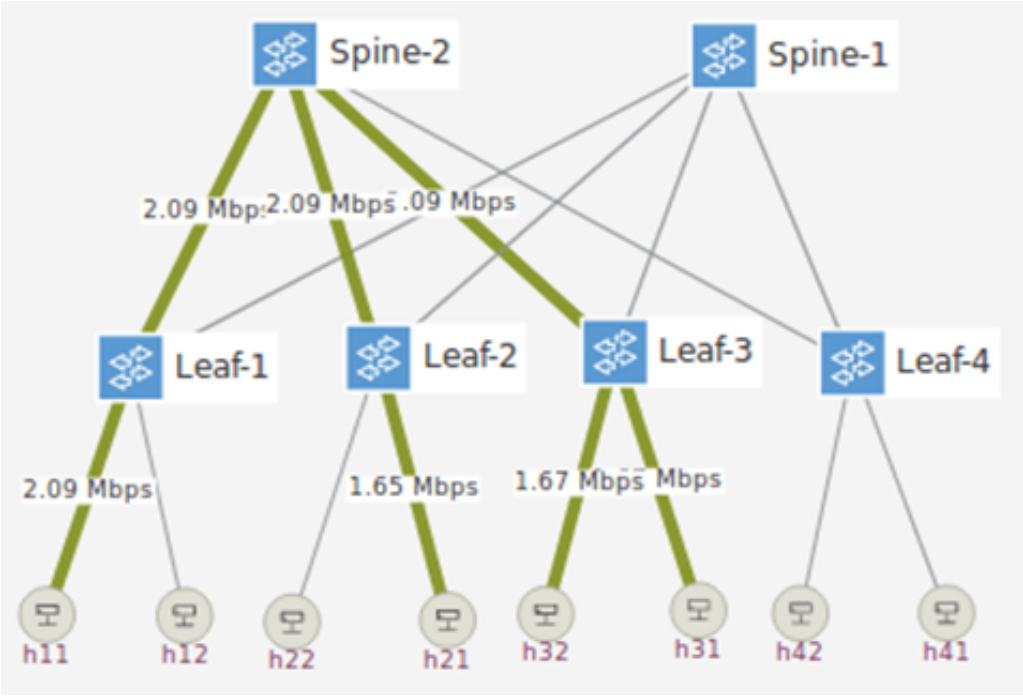

Figure 5.4: SDN network with MFWD, FWD and IGMP snooping apps activated with a multicast stream

Figure 5.4 shows the overall result of this experiment. Every host that wanted to receive the multicast stream is receiving it and the ones that weren't supposed to receive it aren't receiving. To achieve these results the MFWD application and the IMGP snooping application need to work together. The intent functionality is provided by the ONOS core and it is responsible for translating the intents into flow rules, so the path chosen for the multicast stream is from ONOS responsibility. The default algorithm is Dijkstra and it was not changed for these tests. Another important feature that can be observed in figure 5.4 is the bandwidth used which is only 2.09Mbps, a significant reduction from the unicast test.

To complete the test of the IGMP snooping application a receiver closes VLC and sends a IGMP membership report to leave the multicast group. After that the host is removed from the multicast group and the intent is replaced by a new one without the host that left. The new intent removed the unnecessary flows and installed the new ones.

5.3.2 Multiple senders and multiple receivers

The initial setup is similar to the previous one, only this time the network complexity will increase. In this test h11 will stream to the multicast group 226.0.0.0 and h12 will stream to 227.0.0.0. h21 and h41 will connect to the multicast group 226.0.0.0 and h21, h22, and h31 will connect to 227.0.0.0. This way h21 will subscribe two multicast groups at the same time. The purpose is to see how the IGMP snooping application behaves in such scenario.

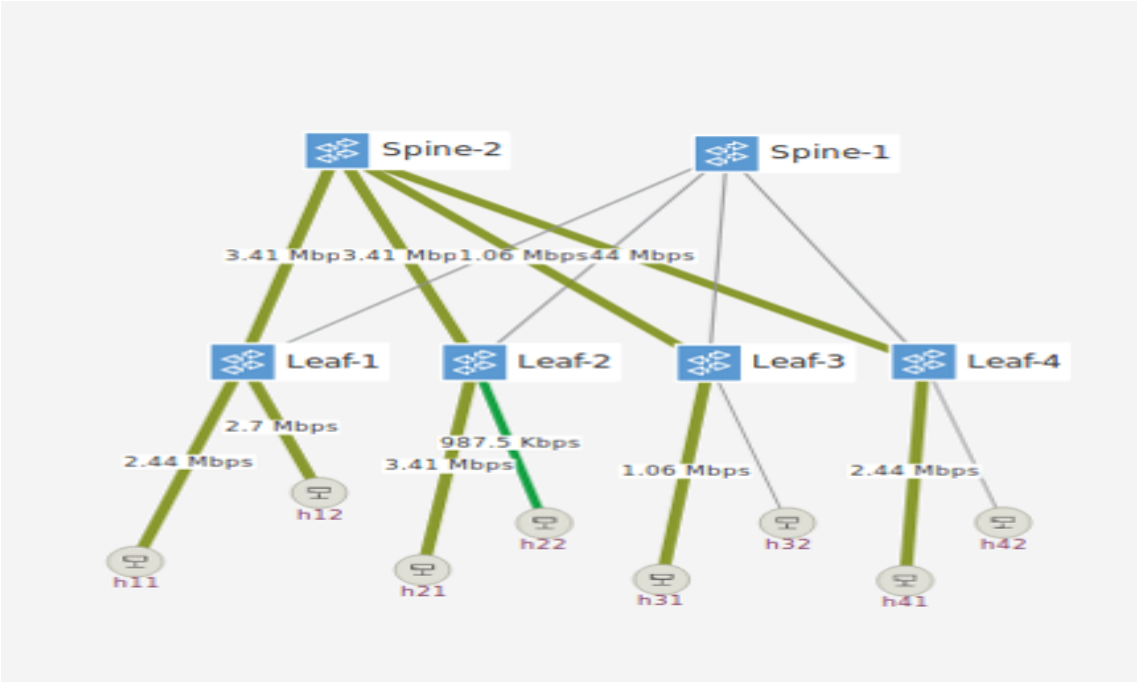

Figure 5.5: Network status of the two streams running simultaneously

Figure 5.5 displays our scenario network with two streams running simultaneously. Only the hosts that subscribed to the multicast groups are receiving the multicast streams. The bandwidth flowing on the link between h21 and Leaf-2 is superior to the bandwidth of the other links because h21 receives both streams. Figure 5.6 presents the new flows installed on Leaf-2 by the intents. One of them sends the incoming traffic to h21 and h22, and the other one only sends it to h21.

IN_PORT:2, ETH_TYPE:ipv4, IPV4_SRC:10.0.0.1/32, IPV4_DST:226.0.0.0/32	imm[OUTPUT:3], cleared:false
IN_PORT:2, ETH_TYPE:ipv4, IPV4_SRC:10.0.0.2/32, IPV4_DST:227.0.0.0/32	imm[OUTPUT:4, OUTPUT:3], cleared:false

Figure 5.6: New flows installed on switch Leaf-2

Some important information is present in figure 5.7. It can be seen how the querier is sending the IGMP Membership Query packets to the hosts every ten seconds and that the host responds with a IGMP Membership Report. In this case it is specific source multicast. It can also be seen how the Membership Report changed from only one group to start to two groups.

```
 91 612.904… 10.0.0.3     224.0.0.22  IGMPv3  Membership Report / Join group 226.0.0.0 for source {10.0.0.1}
 92 620.353… 10.0.0.20    224.0.0.1   IGMPv3  Membership Query, general
 93 620.480… 10.0.0.3     224.0.0.22  IGMPv3  Membership Report / Join group 227.0.0.0 for source {10.0.0.2}
 94 621.128… 10.0.0.3     224.0.0.22  IGMPv3  Membership Report / Join group 227.0.0.0 for source {10.0.0.2}
 95 626.221… 10.0.0.3     224.0.0.22  IGMPv3  Membership Report / Join group 227.0.0.0 for source {10.0.0.2} / Join group 226.0.0.0 for source {10.0.0.1}
 96 630.355… 10.0.0.20    224.0.0.1   IGMPv3  Membership Query, general
 97 633.128… 10.0.0.3     224.0.0.22  IGMPv3  Membership Report / Join group 227.0.0.0 for source {10.0.0.2} / Join group 226.0.0.0 for source {10.0.0.1}
 98 640.358… 10.0.0.20    224.0.0.1   IGMPv3  Membership Query, general
 99 647.976… 10.0.0.3     224.0.0.22  IGMPv3  Membership Report / Join group 227.0.0.0 for source {10.0.0.2} / Join group 226.0.0.0 for source {10.0.0.1}
100 650.360… 10.0.0.20    224.0.0.1   IGMPv3  Membership Query, general
101 659.496… 10.0.0.3     224.0.0.22  IGMPv3  Membership Report / Join group 227.0.0.0 for source {10.0.0.2} / Join group 226.0.0.0 for source {10.0.0.1}
```

Figure 5.7: Wireshark capture from the host h21

5.3.3 Testing the IGMP Timeout functionality

Starting with the same topology as before and having h11 as the sender and h21, h22 and h31 as the receivers, this time the test consists in deactivating one of the hosts and checking that after some time it is removed from the multicast group. This experiment puts to test the timeout capacity of the IGMP application.

Immediately after disconnecting h31, Leaf-3 is still receiving data from the multicast group this can be seen in figure 5.8. This is an expected behavior because not enough time has passed for the host to be timed-out. After some time Leaf-3 stops receiving the multicast traffic. As h31 stopped to send the Membership Reports back to the controller to respond to the IGMP Query packets, it was removed from the multicast group, and new intents were installed which lead to new flow rules installed. This can be seen in figure 5.9. For re-adding h31, it needs to send an IGMP Membership Report to be added again to the multicast group because it was removed from the multicast group by the timeout mechanism. When it sends a new IGMP join packet it is added to the group again, and a new intent is installed, which installs new flows.

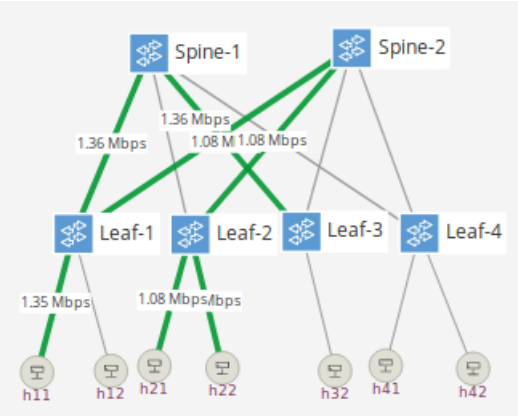

Figure 5.8: Network status right after h31 is disconnected

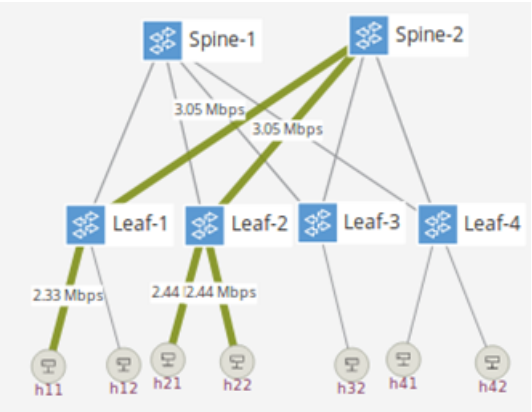

Figure 5.9: Network status after the timeout mechanism is activated

After this test it can be said that the timeout functionality is working, and our application fulfills all the IGMP requirements.

5.4 Stress tests on the IGMP application

After proving that the IGMP snooping application satisfies the IGMP snooping requirements, it is necessary to evaluate how it performs under stress tests representing some "extreme" conditions that can occur in real live scenarios. Only by doing this test can the solution be considered appropriate to be deployed by service providers envisioning the virtualization of some OLT functions in real passive optical access networks.

Three tests were performed. The objective of first one is to see how the ONOS SDN controller and our application behave in a network with a high number of hosts connected to a multicast group. The objective of the second test is to conclude if they can handle sets of multicast groups at the same time. The purpose of the third test is to measure how much time ONOS takes to react to a link failure in a network with redundancy. Each test presents a different topology chosen specifically to each case.

The tests were performed with two computers. One functioning as the SDN controller, running ONOS with the developed IGMP application (VM Xubuntu 32 with 4 Gbyte of RAM, running over virtual machine on a windows 10, 64 bits with 8 Gbyte of RAM and an intel core i5 2.6 GHz) and the other was where the network was instantiated with Mininet (VM Xubuntu 32 with 8Gbyte of RAM, running over virtual machine on a windows 8.1, 64 bits with 16 Gbyte of RAM and an intel core i5 2.5 GHz). Both computers are connected via an Ethernet switch and have the topology displayed in figure 5.10:

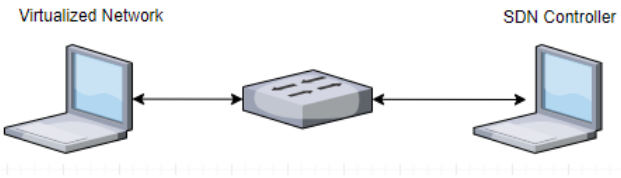

Figure 5.10: Test bed topology

Figure 5.11 presents the system architecture used along these tests. More information about it can be found in the specific test description on the following sub-chapters. All these tests were performed more than 1 time and the values presented are average values.

5.4.1 High number of hosts connected to a single multicast group

The first test consists in a network that simulates an aggregation topology, like the one presented in figure 5.11. A main switch connects to n=16 other switches. Each switch then connects to k hosts. There is a special host, the multicast sender, that connects directly to the main switch which is not represented in the figure. The conclusions taken from this test can be applied to a scenario with OLTs. In the CORD architecture an OLT can be abstracted as a switch OpenFlow where the ONUs are ports of that switch. By increasing

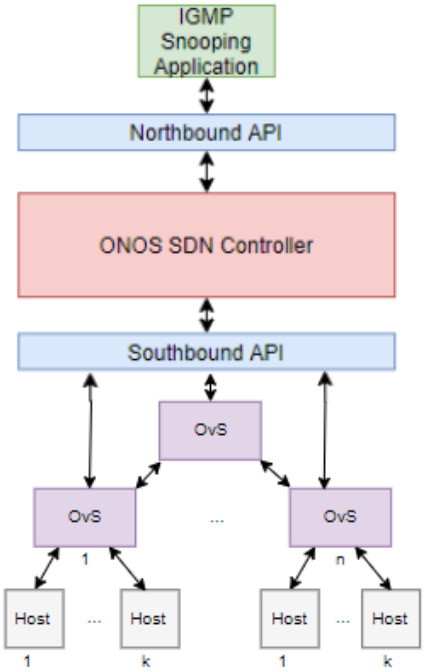

Figure 5.11: Stress tests system architecture

the number of hosts connected to each switch, we are adding load and observing how it affects the SDN controller and the IGMP snooping application. Also, we want to see if it meets the IPTV recommendations regarding the time it takes to connect to a new channel. In this case all the hosts connect to the same multicast group which is the only one available.

The table 5.1 and table 5.2 display the results obtained in these tests for k hosts. The CPU and memory values were calculated in three different instants: initially when the controller was instantiated but the network was not; in the middle, when the network was instantiated and connected to the controller; in the end, when all hosts were receiving the multicast traffic. The available memory is displayed in MB and the CPU usage in percentage. The "Intent Install Time" displayed represents the time it took for an intent to be installed and the "Intent Withdraw Time" the time to be withdrawn after giving the controller the order to do so. The IGMP processing time is the time it took for the IGMP snooping application to process the packet. The Average total time is the time it took for the host to start receiving the multicast traffic after sending a IGMP join packet and was measured via an interface capture analyzed in wireshark. All these times are presented in milliseconds (ms).

When we had a low number of hosts (up to $k = 10$ hosts per switch, 160 total) everything worked as expected. On average, the IGMP snooping application took up to 8 ms to process an IGMP packet and a maximum of 100 ms. The time it took from the moment

	Number of Hosts	CPU Usage			Available Memory (Mb)		
		Init.	Middle	End	Init.	Middle	End
Controller	16	1.2%	50%	3%	2102	2101	2104
	32	1.4%	51%	1.7%	2103	2077	2071
	160	1.5%	45%	8.8%	2080	2077	2053
	320	1.7%	50%	51%	2020	2020	1993
	608	1.6%	63%	92%	1728	1700	1665
Network Virtualizer	16	0%	0.7%	0.5%	7081	7041	6885
	32	0%	0.8%	0.8%	7073	7002	6716
	160	0%	3%	10%	7072	6709	5347
	320	0%	5%	4.3%	6871	6218	3667
	608	0%	11%	63%	7094	5459	42

Table 5.1: Test one computer resources results

Number of Hosts (k)	Intent Install Time			Intent Withdraw Time			IGMP Processing Time			Average total time
	min	max	mean	min	max	mean	min	max	mean	
16	18	45	26	17	45	25	1	25	3	105
32	17	98	36	15	99	31	1	94	6	170
160	16	240	32	11	247	18	1	104	8	195
320	17	182	26	11	171	19	0	1472	16	320
608	-	-	-	-	-	-	1	2636	43	445

Table 5.2: Test one processing times results in ms

an IGMP packet was sent and the multicast traffic started to be received by the host was on average 195 ms (with a maximum of 478 ms). As previously mentioned the results are in accordance with the IPTV recommendations. It was also observed a peak on the CPU usage when the hosts started to send IGMP join packets to the controller but with this number of hosts it was not a problem. In the end, after processing the initial burst of packets, the CPU usage goes back to low starting levels.

When we had a higher number of hosts (total ranging from $k = 20$ up to $k = 50$) some not so good results started to appear. The average time it took to process an IGMP packet was 43 ms and the maximum was 2,6 s. Even though the average time it took to start to receive the multicast traffic, after sending the IGMP packet, was 455 ms, in some cases it reached 2636 ms. This means that the 1 second barrier was exceed. Also, the SDN controller CPU usage never went back to the low starting levels. In a real scenario this can lead to a bad costumer experience and a low quality of service.

For the test with 608 hosts, the intent install time and intent withdraw time was not possible to access. To get them we used the ONOS application events that records every event that happens in the network and for this case the application crashed from the overload of data and to access it was impossible.

Running this test with different processing capacity it was verified that the system response does not vary linearly and further evaluation is in course. The results of these tests can be seen in table 5.3.

Number of Hosts	Intent Install Time (ms)			Average total time (ms)
	min	max	mean	
16	1	32	6	126
32	0	279	16	203
160	0	396	32	736

Table 5.3: Test one processing times results half the processing capacity

With this experiment we can conclude that the memory(RAM) capacity was not so important to the SDN controller, packet processing is a CPU heavy task and only with the right hardware it is possible to achieve optimal times for a good consumer experience. Also, virtualizing the network is a memory intensive task.

5.4.2 High number of multicast groups with one sender and one receiver

The objective of the second test is to show how many different multicast groups can the SDN controller and the IGMP snooping application manage. This way we can know how many different channels can operate in the network and if it handles the average number of channels of IPTV services. The second test uses the same network topology presented in Fig. 5.11 with $n = 2$. The topology is a simple one-to-one connection between a variable number of hosts working as senders and the same number of hosts working as receivers via an Ethernet switch.

On average an IPTV typical scenario has 500 channels available and the peak number of different channels being watched at the same time is 250 channels. With this test we can conclude that ONOS handles that number of multicast groups at the same time without any problem. Every host was added by the IGMP snooping application to the right multicast group. ONOS converted the information about the multicast groups in intents and after compiling them it installed the right flows. However sometimes even though it had the right intents it did not install the necessary flows into the switches, and that happens even with a low number of multicast groups. This lead to hosts not receiving the multicast traffic which can lead to a bad consumer experience, and a low quality of service. This problem results from the ONOS intent mechanism which is still under development

5.4.3 Link failure recovery tests

The third test focuses on the ONOS intent ability where the objective is to see how fast it recompiles the intent and installs the necessary flow rules to accommodate the failure of a link and if the number of hosts in an intent changes the reaction time. The topology used is the classic spine-leaf topology displayed in figure 5.12. The host h11 starts a multicast stream that every other hosts joins. To simulate a link failure the link between Leaf-1 and Spine-2 is put down, and after a few seconds up again. The same thing is done to the link Leaf-1 and Spine-1.

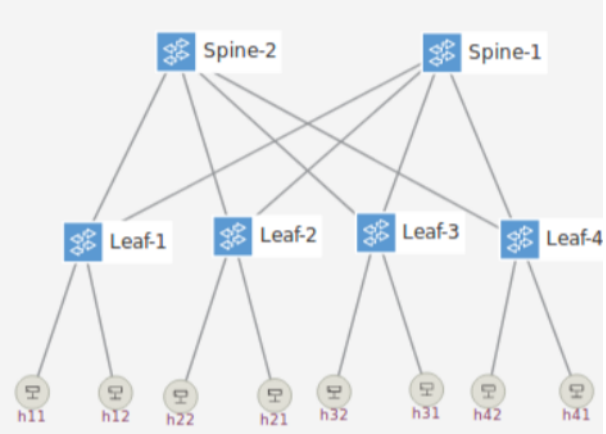

Figure 5.12: Spine-Leaf topology used to test the link failure

In case of a link failure the typical recommended time for the network to recover is less than 50 ms. After this test we concluded that it took, on average, 45 ms to react to a link failure scenario. This value is calculated from the moment the link goes down until new openflow rules are installed. This is considered a good response time and it is almost imperceptible to the final user. If the network was under a heavy load i.e. with more than 160 hosts connected at the same time to a multicast group, and the SDN controller CPU usage was at high levels, it could take up to 300 ms to react which is a poor response time.

6

Conclusion

Throughout this dissertation ONOS was tested as well as the developed IGMP snooping application with access networks in mind. The IPTV use case was the main beneficiary of the executed tests.

After analyzing the results of the tests it can be concluded that ONOS is neither ready to be deployed by network service providers nor ready for the IPTV distribution. First of all, some undocumented problems were found in the ONOS core e.g. configuring the FWD application didn't work and sometimes the intent functionality didn't install the proper OpenFlow rules. Second, ONOS lacks good documentation for the existing applications. Even though ONOS has various network applications already implemented it is difficult to understand how they interact and how they work. Third, the stress tests revealed that ONOS needs to have enough processing capacity that must be properly evaluated in order to achieve good performances in real networks.

The stress tests covered a scenario equivalent to a single OLT but the computation power used allocated to ONOS was more than what a traditional OLT has. The OLT has a CPU specialized in embedded scenarios and less memory available and in spite of that it is still able to perform the IGMP snooping functionalities.

The tests on the IGMP snooping application revealed that it meets the IGMP snooping requirements as well as the ones from an IGMP querier. Namely,it processes all kinds of IGMP packets, is able to add and remove hosts from the multicast groups created by ONOS application MFWD and has a timeout function that removes inactive hosts that didn't send the leave packet.

6.1 Future Work

Even though the IGMP application was widely tested, new scenarios can be made to test it further. In the future, more network applications can also be developed for ONOS in order to increase its viability for network service providers deployment. Also, for the available applications better documentation is necessary and more proof-of-concept scenarios can be used to test them.

Bibliography

[1] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[2] Broadband Forum, "TR-384 Cloud Central Office Reference Architectural Framework," Broadband Forum, Tech. Rep. January, 2018.

[3] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[4] T. D. Nadeau and K. Gray, *SDN - Software Defined Networks*. O'Reilly, 2013.

[5] P. Morreale and J. Anderson, *Software Defined Networking*. CRC Press, 2014. [Online]. Available: http://www.crcnetbase.com/doi/book/10.1201/b17708

[6] P. Göransson, C. Black, and T. Culve, *Software Defined Networks A Comprehensive Approach*. Elsevier, 2017.

[7] ONF, "Software-Defined Networking (SDN) Definition - Open Networking Foundation." [Online]. Available: https://www.opennetworking.org/sdn-definition/

[8] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 27–51, Firstquarter 2015.

[9] M. R. Costa, R. Calé, C. Parada, P. Neves, and J. Bonnet, "NFV & SDN INNOVATION STRATEGY AT ALTICE LABS," *innovaction*, pp. 34–53, 2016.

[10] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[11] Broadband Forum, " TR-370 Fixed Access Network Sharing - Architecture and Nodal Requirements," Broadband Forum, Tech. Rep. November, 2017.

[12] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 623–654, Firstquarter 2016.

[13] L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar, and W. Snow, "Central office re-architected as a data center," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 96–101, oct 2016.

[14] "ONOS - A new carrier-grade SDN network operating system designed for high availability, performance, scale-out." [Online]. Available: https://onosproject.org/

[15] "What is Docker?" [Online]. Available: https://www.docker.com/what-docker{#}/overview

[16] "XOS - Open Networking Foundation." [Online]. Available: https://www.opennetworking.org/projects/xos/

[17] "Software - OpenStack is open source software for creating private and public clouds." [Online]. Available: https://www.openstack.org/software/

[18] "Features - ONOS." [Online]. Available: https://onosproject.org/features/

[19] T. O. N. L. (ON.Lab), "Introducing ONOS - a SDN network operating system for Service Providers," vol. 1, p. 14, 2014.

[20] "VideoLAN." [Online]. Available: https://www.videolan.org/vlc/index.html

[21] R. L. S. De Oliveira, A. A. Shinoda, C. M. Schweitzer, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*. IEEE, 2014, pp. 1–6.

[22] "Features - OvS." [Online]. Available: http://www.openvswitch.org/features/

[23] N. Kocharians and V. Terry, *CCIE Routing and Switching v5.0*, 5th ed. Ciscopress, 2014.

[24] W. Fenner and X. PARC, "Internet Group Management Protocol, Version 2," *RFC*, pp. 1–24, 1997.

[25] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet Group Management Protocol, Version 3," *RFC*, pp. 1–53, 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3376

[26] M. J. Christensen, K. Kimball, and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches," 2006. [Online]. Available: https://tools.ietf.org/html/rfc4541?ref=binfind.com/web

[27] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-g. Liu, and L. Wei, "Protocol independent multicast (pim): Protocol specification," *Internet Draft RFC*, 1995.

[28] S. Shoaf and M. Bernstein, "Introduction to igmp for iptv networks," *white paper juniper networks, part*, no. 200188-001, 2006.

[29] A. Technologies, "How network conditions impact IPTV QoE," vol. 1, p. 10, 2008.

7

Appendix A

IGMP Snooping for Multicast Traffic in Software-Defined Networking

José Castanheira[1] Andre Brizido[2] Lucia Martins[1,3]

***Abstract*— In this demonstration it is presented an IGMP snooping application developed for ONOS SDN controller. The applicability of this SDN/NFV solution in the context of passive optical access networks is showed through its performance evaluation for a set of multicast traffic scenarios including the case of a single link failure.**

I. INTRODUCTION

Over the last few years, a revolution has been occurring over networks. SDN and NFV appeared as new technologies that can change the current network architecture paradigm. SDN refers to a way network functionalities can be organized and can be defined as the separation of the control plane from the data plane to a centralized controller [1]. NFV can be seen as the decoupling of networks functions from the highly specialized physical hardware into software application running in common hardware [2]. From the perceptive of network service providers one of these network functions is IGMP snooping [3]. IGMP is a protocol used in networks that allows hosts to advertise which multicast group they want to join or leave. IGMP snooping is the process of listening to IGMP packets from the hosts to learn which host wants to subscribe/leave to a certain multicast group [4]. This way, instead of broadcasting the traffic to every host, the switch is capable of only forwarding the packets to the hosts that want to receive them. In an SDN-NFV environment the IGMP snooping is performed through the SDN controller, which installs and removes the flows in accordance with the IGMP requests. Multicast traffic is used in IPTV, where users receive TV over an IP platform. Because all the channels aren't being watched at the same time, IGMP snooping is required to know which channel is on and when zapping is done which multicast group must leave and which multicast group must be jointed. With this mechanism, service providers can save bandwidth preventing the network to be overloaded with unwanted traffic. On IPTV the ideal time for a user to start watching a new channel is less than 200 ms and the acceptable time is less than one second [5], [6].

In this work an IGMP snooping application was developed for ONOS SDN controller [7]. In order to show the adequacy of this solution envisioning the virtualization of some OLT (Optical Line Termination) functions in real passive optical access networks a demonstration is presented using a set of multicast traffic scenarios, described next.

[1] Department of Electrical and Computer Engineering, University of Coimbra, 3030-290 Coimbra, Portugal
[2] Altice Labs, 3810-106 Aveiro, Portugal
[3] INESC Coimbra, 3030-290 Coimbra, Portugal

II. DEMO SYSTEM ARCHITECTURE

The system architecture used for this demo can be seen in figure 1. The IGMP snooping application follows the description presented in [3] and was implemented in Java. It communicates through Java API with the SDN controller. OvS were the switches instantiated via Mininet [8] and where the SDN controller installs the flows rules.

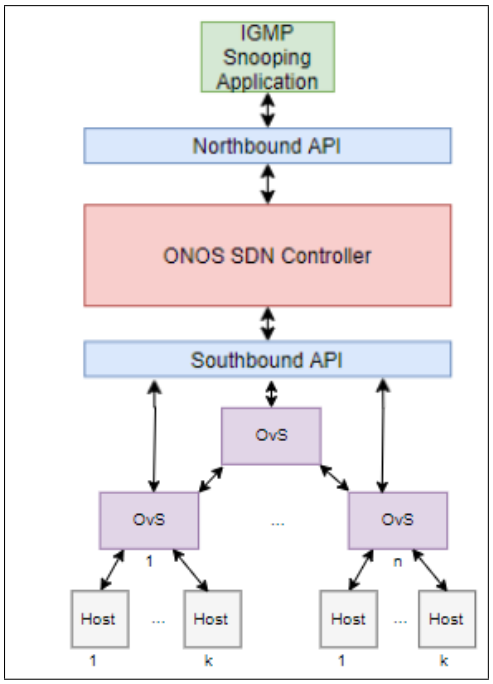

Fig. 1. Demo system architecture

The tests were performed with two computers. One functioning as the SDN controller, running ONOS with the developed IGMP application (VM Xubuntu 32 with 4 Gbyte of RAM, running over virtual machine on a windows 10, 64 bits with 8 Gbyte of RAM and a intel core i5 2.6 GHz) and the other was where the network was instantiated with Mininet (VM Xubuntu 32 with 8Gbyte of RAM, running over virtual machine on a windows 8.1, 64 bits with 16 Gbyte of RAM and a intel core i5 2.5 GHz). Both computers are connected via an Ethernet switch.

A. Test one

The first test uses the network topology presented in Fig. 1. A main switch is connected to $n = 16$ other switches. Each switch then connects to a variable number of k hosts. There is a special host , the multicast sender, not represented in the figure, that connects directly to the main switch. The

objective of this first test is to evaluate the performance of this system with a high number of hosts connected to the same multicast group and check if ONOS and our IGMP snooping application met the IPTV recommendations.

B. Test two

The second test uses the same network topology presented in Fig. 1 with $n = 2$. The topology is a simple one-to-one connection between a variable number of hosts working as senders and the same number of hosts working as receivers via an Ethernet switch. The objective of this test is to check how many different multicast groups can the SDN controller and the IGMP snooping application manage and therefore know how many different IPTV channels can operate in the network.

C. Test three

The third test uses the classic leaf-spine topology [9]. This test is focused on the ONOS intent ability where the objective is to see how fast it recompiles the intent and install the necessary flow rules to accommodate a failure of a link. It also verifies if the number of hosts involved in a intent affects the reaction time.

III. Experimental Results

After running the experiments the following results were obtained:

A. Test one

When we had a low number of hosts (up to $k = 10$ hosts per switch, 160 total) everything worked as expected. On average, the IGMP snooping application took 8 ms to process an IGMP packet and a maximum of 100 ms. The time it took from the moment an IGMP packet was sent and the multicast traffic started to be received by the host was on average 195 ms and on maximum of 478 ms. As previously mention the results are in accordance with the IPTV recommendations. It was also observed a spike on the CPU usage when the hosts started to send IGMP join packets to the controller but with this number of hosts it was not a problem. In the end, after processing the initial burst of packets, the CPU usage goes back to low starting levels.

When we had a higher number of hosts (total ranging from $k = 20$ up to $k = 50$) some not so good results started to appear. The average time it took to process an IGMP packet was 43 ms and the maximum was 2,6 s. Even though the average time it took to start to receive the multicast traffic, after sending the IGMP packet, was 455 ms, in some cases it reached 3 s. This means that the 1 second barrier was exceed. Also, the SDN controller CPU usage never went back to the low starting levels. In a real scenario this can lead to a bad costumer experience and a low quality of service.

Running this test with different processing capacity it was verified that the system response does not vary linearly and further evaluation is in course. It was also observed that the memory(RAM) capacity was not so important to the SDN controller.

B. Test two

On average an IPTV typical scenario has 500 channels available and the peak number of different channels being watched at the same time is 250 channels. With this test we can conclude that ONOS handles that number of multicast groups at the same time without any problem. Every host was added by the IGMP snooping application to the right multicast group. ONOS converted the information about the multicast groups in intents and after compiling them it installed the right flows. However sometimes even though it had the right intents it did not installed the necessary flows into the switches, and that happens even with a low number of multicast groups. This lead to hosts not receiving the multicast traffic which can lead to a bad costumer experience, and a low quality of service. This problem results from the ONOS intent mechanism which is still under development [7].

C. Test three

In case of a link failure the typical recommended time for the network recover is less than 50 ms. After this test we concluded that it took, on average, 45 ms to react to a link failure scenario. This value is calculated from the moment the link goes down until new openflow rules are installed. This is considered a good response time and it is almost imperceptible to the final user. If the network was under a heavy load i.e. many hosts connected at the same time to a multicast group, and the SDN controller CPU usage was at high levels, it took 300 ms seconds to react which is poor response time.

IV. CONCLUSIONS

After these tests we can conclude that ONOS is not yet ready to be used by network service providers and also needs some serious processing power to be able to perform properly in really networks.

The computation power used by the SDN controller in these tests is more than that a traditional OLT pizzabox has. The OLT has a CPU specialized in embedded scenarios and less memory available and in spite of that it is still able to perform the IGMP snooping functionalities.

References

[1] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, Network function virtualization: State-of-the-art and research challenges, IEEE Communications Surveys and Tutorials, vol. 18, no. 1, pp. 236262, 2016.

[2] P. Goransson, C. Black, and T. Culve, Software Defined Networks A Comprehensive Approach. Elsevier, 2017.

[3] M. J. Christensen, K. Kimball, and F. Solensky, Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches, 2006.

[4] N. Kocharians and V. Terry, CCIE Routing and Switching v5.0, 5th ed. Ciscopress, 2014.

[5] S. Shoaf and M. Bernstein, Introduction to igmp for IPTV networks, white paper Juniper Networks, part, no. 200188-001, 2006.

[6] A. Technologies, How network conditions impact IPTV QoE.

[7] ON.Lab, Introducing ONOS - a SDN network operating system for Service Providers, vol. 1, p. 14, 2014.

[8] http://mininet.org.

[9] https://github.com/ciena/mininet-topos