UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

MASTER THESIS

THESIS TITLE: OPEN PON: Seamless integration between 5G core and optical access networks

DEGREE: Master in Science in Telecomunication Engineering & Management

AUTHOR: Nuno Filipe Mateus De Matos Silva

ADVISORS: Sebastià Sallent Ribes
David Rincón Rivera

DATE: September 5, 2016

Title : OPEN PON: Seamless integration between 5G core and optical access networks

Author: Nuno Filipe Mateus De Matos Silva

Advisors: Sebastià Sallent Ribes
David Rincón Rivera

Date: September 5, 2016

Overview

Nowadays business and users traffic demands are getting higher and higher, leaving optical fiber as the only reliable solution to meet its demands, and in access networks the dominating technology is Passive Optical Networks (PONs) where currently exists two main families of standards, the GPON (ITU-T standard) and the EPON (IEEE standard).

In this work we tackle the latest version oof the GPON family, the NG-PON2. We analyse the Physical Media Dependent (PMD) and Transmission Convergence (TC) layer requirements and what is standardized in each one.

We also have a look into the SDN paradigm, which decouples the control and data plane of network equipment concentrating the intellegence and complexity in a centralized controller, the OpenFlow protocol that is one of the main enablers of SDN, since it is the communication protocol betwwen the network devices and the SDN controller, and the SIEPON (IEEE 1904.1) standard which gives interoperability of the transport, service, and control planes in a multi-vendor environment.

The introduction of the SDN paradigm in the xPON world is a hot research topic because it would add the total controllability of the PON infrastructure, and the coordination with the SDN-based control plane of the core networks. It would allow a centralized optimization of the different resources managed by the OLTs and a seamless coordination between OLTs and the core network.

In the TC layer specifications, of NG-PON2, it is not standardized the bandwidth allocation of the system, we set ourselves to create a simple Dynamic Bandwidth Allocation (DBA) algorithm that would optimize the use of channels and wavelengths.

Taking into account the NG-PON2 technology and all its features, the emerging trend to extend SDN to the optical access networks, the multi-vendor adaptation provided by the SIEPON stantdard, and the need to have a platform to develop and test DBA scheduling algorithms made that thecreation and deployment of a fully SIEPON-SDN-NG-PON2 emulation environment became the main goal of this work.

To create this environment we employed the use of MikroTiks, where OpenWRT was installed as well as a set of packages that menabled the behaviour of a CPqD switch (or *ofsoftswitch13*), and the use of Raspberry Pis which maintained its original OS and only required the installation of a sepecific set of tools.

The results obtained were quite positive confirming the possibility of not only creating but also testing different DBA scheduling algorithms in the deployed environment and matching the results expected to our created algorithm.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Optical fiber provides one of the most powerful solutions for existing and future Internet requirements because with optical fiber technologies, bandwidth demands are satisfied and a wide range of services and applications can be supplied.

The incessant rise in data consumption by both business and consumer users drives continual industry innovation to meet this challenge leaving optical fiber and consequently Passive Optical Networks (PONs) as one of the most reliable solutions.

Passive Optical Network (PON) is a telecommunications technology that implements a point-to-multipoint architecture, in which unpowered Fiber Optic Splitters are used to enable a single optical fiber to serve multiple end-points such as customers, without having to provision individual fibers between the hub and customer. Its main components are the optical line terminal (OLT), the optical network units (ONUs) and the optical distribution network (ODN), composed of optical fiber and power splitters which connect the OLT to the ONUs. The main components are further explained in Chapter 1 and they can be seen, as well as the PON architecture, in Fig. 1.

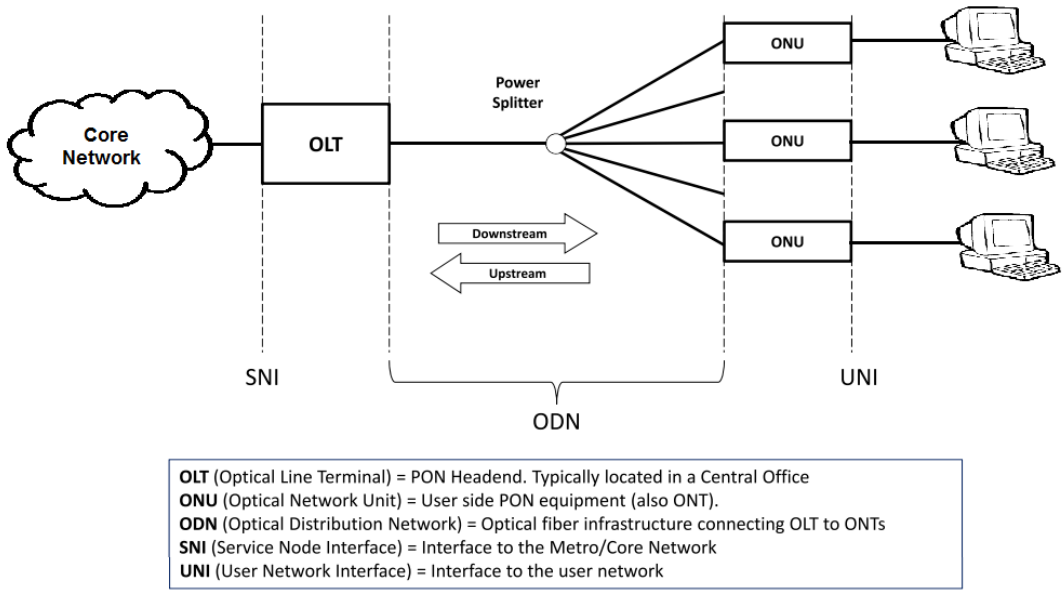


Figure 1: PON architecture and terminology. [2]

In a PON system the downstream channel is broadcasted, thus its sharing between the different users is easily controlled by the OLT. On the other hand, the upstream channel is shared and needs to have some type of Medium Access Controll (MAC) protocol in order to avoid collisions between different users. The main reason for this shared environment is the passive nature of the network.

The PON technology has the ability to relieve bottlenecks in the access network and reducing the amount of fiber and central equipment required compared to other architectures (e.g. point-to-point), these reasons made PONs a success being currently deployed on a mass market scale by numerous network operators worldwide.

When it comes to PON technologies there are two that come to mind, GPON (ITU-T standard) and EPON (IEEE standard). The main difference between the two protocols is the architectural approach. EPON only uses Ethernet frames to carry data, voice, and video. GPON can transport not only Ethernet, but additionally ATM and TDM traffic by using GPON Encapsulation Method (GEM).

When it comes to bandwidth GPON has an advantage allowing higher data rates, both upstream and downstream, and a higher bandwidth efficiency therefore making it more suitable to relieve bottlenecks and for core networks, it also supports triple-play services that are getting more popular. However, EPON equipment is significantly cheaper and simplifies the networks. A simple side-by-side comparison is shown in Table 1. Regarding Table 1 it is important to point out that the power budget can change drastically the range of the network and the split racio.

Table 1: Comparison between GPON and EPON.

Standard	**ITU-T G.984 GPON (2008)**	**IEEE 802.3ah EPON (2004)**	**IEEE 802.3av 10G-EPON (2009)**
Service	Full services	Ethernet data	Ethernet data
Frame	GEM frame	Ethernet frame	Ethernet frame
Distance	60 km	20 km	40km
Split ratio	1:128	1:32	1:64
Upstream	1.244 Gb/s	1.25 Gb/s	1.25 or 10.3125 Gb/s
Downstream	2.488 Gb/s	1.25 Gb/s	10.3125 Gb/s
Wavelength	DS: 1480-1500 nm US: 1290-1330 nm	DS: 1480-1500 nm US: 1260-1360 nm	DS: 1575-1580 nm US: 1260-1280 nm

Although EPON is still the mainstream, especially in Asia, GPON is expanding quite quickly and maturing much faster. This work will be focused on the GPON technologies.

The PON evolution has been defined in two stages: NG-PON1 and NG-PON2. There would be mid-term upgrades in NG-PON1 (that later became XG-PON) and a long-term solution in NG-PON2.

The main introduction in NG-PON2 is the support of 4-8 wavelengths having an aggregated capacity of 40 Gb/s, it also supports 10G symmetrical links which fits the increasing need for higher upstream traffic and allows more services.

Due to what has been described above, there are many operators that have not employed XG-PON expecting to do a direct migration from GPON to NG-PON2. The main characteristis of each GPON technology can be seen in Table 2.

Software-Defined Networking (SDN) is an approach that decouples the control and data planes of network equipment, and concentrates the intelligence and complexity of devices in a centralized controller which has a global view of the state of the network. This seems to be a great trend in telecommunications and quickly being adopted by operators.

The centralized system required for SDN matches the architecture of current optical access networks, consisting of mostly PONs which have a tree topology (centralized). This facilitates the implementation of SDN as well as taking advantage of its full benefits.

Table 2: Comparison between GPON, XG-PON and NG-PON2.

Standard	**GPON (2008)**	**XG-PON (2012)**	**NG-PON2 (2013)**
PON Rate	2.488/1.244 Gb/s	9.953/2.488 Gb/s	9.953/9.953 Gb/s
Downstream λ	1480-1500 nm	1575-1580 nm	1596-1603 nm
Upstream λ	1260-1360 nm	1260-1280 nm	1524-1544 nm (wide) 1524-1540 nm (narrow)
Distance	20/40/60 km	20/40 km	20/40 km
Split ratio	1:128	1:128	1:256

Therefore, the application of the SDN principle to PONs is a very interesting topic. It can open the way not only to a centralized optimization of the different resources (wavelengths, time slots) managed by a single OLT, but also to the seamless coordination between OLTs and the core network, for example by optimizing the buffering delay or the wavelength assignment between both the core and the access domain.

The goal of this work is to contribute to the aforementioned research effort by creating a fully SDN and NG-PON2 emullation environment where different scheduling algorithms can be created and tested and, additionally create and test a scheduling algorithm of our own.

In this work, we have used ONOS, an SDN controller, and use the CPqD Switch to emulate the behaviour of an Openflow enabled OLT, which would work similar to what is described in [1]. One of the advances of the NG-PON2 technology is that uses 4 wavelengths instead of only 1, while this can be extremely beneficial it needs to be well managed in order to be efficient and guarantee a good QoS without disrupting the remaining connections.

To accomplish the desired environment it was simulated an optical access network of one OLT and two ONUs, the devices are simulated using a Mikrotiks as SDN network elements, being controlled and managed by ONOS. To identify and manage the different wavelengths it was used VLANs.

The environment purpose is to run and test different algorithms whose function is to manage the four simultaneous λs used by NG-PON2, since this feature adds extra parameters and more complexity to the previous PON systems a new DBA (Dynamic Bandwidth Allocation) algorithm needs to be found and tested in order to take full advantage of the extra capacity as well as having an efficient and well managed network.

Additionally, as a validation of our testbed, we have proposed a scheduling algorithm that transfers the traffic from a fully occupied λ to another with some capacity still available, the main purpose of the algorithm is to optimize the use of resources, keeping a the QoS and decreasing the energy consumption.

The results show that our environment is working properly, enabling the creation and test of DBA scheduling algorithms, and that the algorithm we have created is also working as expected, optimizing the use of the available channels.

CHAPTER 1. NG-PON2 REQUIREMENTS AND STANDARDS

This chapter contains a description of NG-PON2, a 40 Gbit/s capacity PON system that exploits both time and wavelength (λ) domains. It was designed by ITU-T and it is the follow up to XG-PON1.

XG-PON1 employs a combination of WDM and TDM techniques that ensure the coexistance with legacy PONs, it provides 10 Gb/s downstream and 2.5 Gb/s upstream. The standard presents PMD and TC layer specifications as well as an ONU management and control interface (OMCI), some of this specifications also fit the NG-PON2 standard, thus existing a big similiarity between both standards, especially in the TC layer.

1.1. Network operator requirements

The architecture and terminology relating to PON systems are shown in Fig. 1. The most relevant elements are the optical line terminal (OLT), the optical distribution network (ODN) and the optical network unit (ONU). Since the OLT and ONU are located in the Central Office and at the end of the network, respectively, they are easily accessed and replaced by OLTs and ONUs compatible with NG-PON2. On the other hand the ODN, which is composed of fiber and optical power splitters, would require a great investment and time to replace this being the main reason behind most requirements demanded by operators and service providers, a majority of them according to [4].

We will now describe some of the requirements for the second generation of PON systems.

NG-PON2 could be a disruptive technology exploiting new ODNs with wavelength (λ) splitters instead of power splitters [3]. Nonetheless, considering the investments made to date in power splitters based ODNs it became clear that one of the main requirements of NG-PON2 is the compatibility with such deployed ODNs.

Although λ-splitter based ODNs could be used by this technology (e.g., Greenfield scenario) the optical transmission technology must not require a λ-splitter to work.

Another requirement caused by the re-use of deployed optical fiber infrastructure is the optical path loss. Since it needs to work over the same optical passive infrastructure (ODN) defined for GPON and XG-PON1 it needs to be able to achieve the same optical budget class as defined for XG-PON1, meaning an optical path loss of 29 to 35 dB (maximum) with a 15 dB differential loss. The differential loss as defined in [2], is the absolute difference between the optical losses of any two given paths within the same ODN.

It must also co-exist with previously deployed ITU-T PON systems, i.e., G-PON and XG-PON1 as seen in Fig. 1.1. This smoothens the ongoing migration of subscribers from the old PON systems to the new system as well as allowing the addition of new NG-PON2 subscribers without disturbing or ceasing the services provided to the customers that remain with the old systems.

The co-existence factor needs to be considered while determining the wavelength plan. Since it will work with previously deployed PON systems and we cannot increase the optical budget or disrupt the existing services of the operating PON systems the impact created by linear/non-linear crosstalk should be zero or minimal.

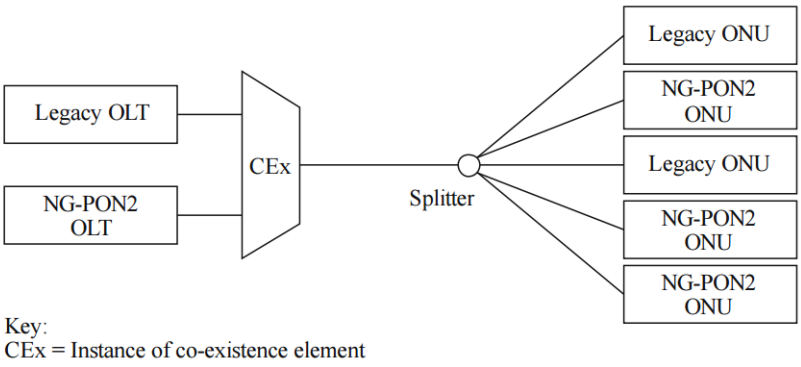


Figure 1.1: ODN coexistance scenario. [4]

Network operators are looking to offer up to 1 Gbit/s of sustainable bandwidth services to any ONU, due to this wish it was set at 40 Gbit/s baseline downstream capacity for the NG-PON2. This is accomplished by using multiple wavelength channels, so a colorless ONU is essential to facilitate the tunability of both receivers and transmitters when they need to change for a different wavelength channel.

All companies try to reduce as most as possible the operational expenditure (OPEX) and network operators are no different, in order to do this they need to keep the inventory to a minimum and have power efficient devices. This means that the number of network equipment part variants should be limited to practical numbers and colorless ONUs facilitate that, especially for the high cost sensitive, mass-market residential services.

1.2. NG-PON2 standards

There was a benchmarking exercise undertaken by FSAN (Full Service Access Network) to compare the available technology options such as Time Division Multiplexing (TDM), Wavelength Division Multiplexing (WDM) and Orthogonal Frequency Division Multiplexing (OFDM) being the chosen approaches Time and Wavelength Division Multiplexing (TWDM) as the primary solution for NG-PON2 and having the option to coexisting with Point-to-Point Wavelength Division Multiplexing (PtP WDM) overlay channels.

The reason for two approaches instead of only one is the need to offer dedicated, high bandwidth and low latency links for some services on the same ODN as residential, as illustrated by Fig. 1.2.

1.2.1. Time and Wavelength Division Multiplexing

TWDM is the main solution for NG-PON2 and is directed to high volume residential application. TWDM-PON uses four, DWDM spaced, bi-directional λ-channels, each of 10 Gbit/s downstream and 2.5 Gbit/s upstream line rate. This results in the aggregated capacity of 40 Gbit/s downstream and 10 Gbit/s upstream. There is made an extensive re-use of XG-PON1 developments plus the newly λ-domain that needs to be controlled and managed in order to enable an incremental increase of capacity and reap its full benefits [6].

For this technology to work properly, it is needed colorless ONUs as well as λ-tunable

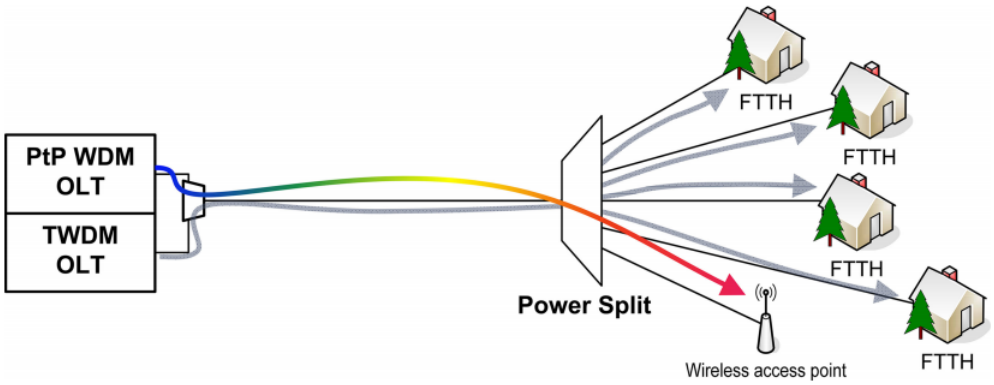


Figure 1.2: TWDM-PON overlaid with PtP WDM PON to meet different service requirements. [2]

transmitters and receivers that need to be developed at lower cost representing one of the key challenges for TWDM-PON [7].

1.2.2. Point-to-Point WDM overlay

PtP WDM enables a dedicated λ-channel to be provided to each ONU allowing NG-PON2 to meet the demanding requirements for business and backhaul services.

Most of the ideas discussed in the Tunable WDM-PON proposal [2] have been adopted as a PtP WDM overlay. In the initial configuration, eight channels of PtP WDM are considered in order to allow full co-existence with the legacy PON systems. However, if there is unused spectrum in a certain deployment that can be used to create additional PtP WDM channels in a flexible way. An image showing TWDM and PtP WDM coexistance can be seen in 1.3.

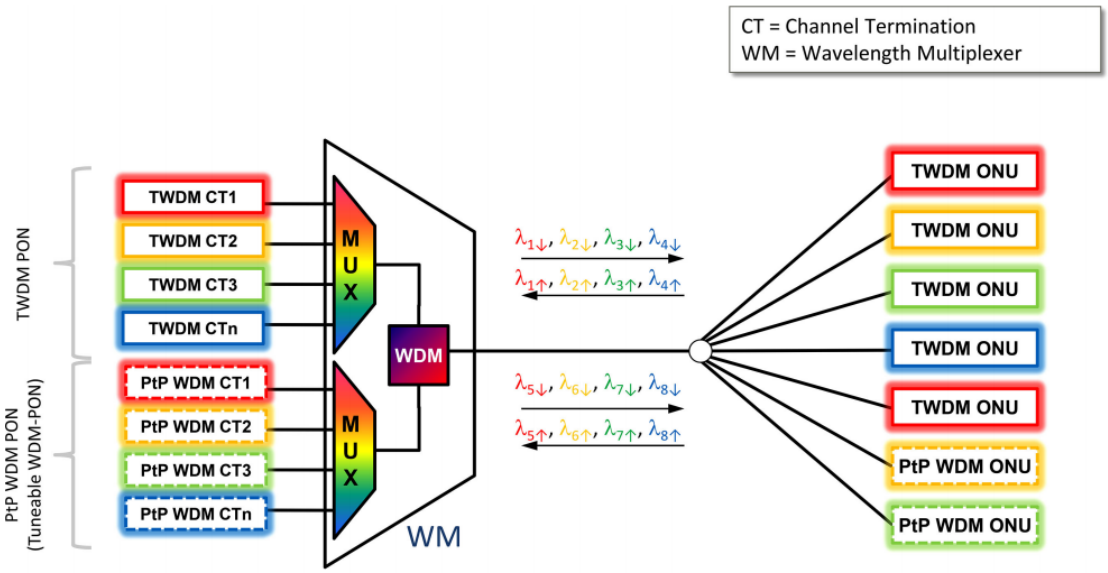


Figure 1.3: NG-PON2 system architecture and TWDM and PtP WDM coexistance. [2]

The ONUs required for PtP WDM use similar low cost tunable receivers and transmitters as the ones from TWDM-PON, the main difference is the operation mode while TWDM-PON uses burst mode the PtP WDM uses a continuous mode. The capacity of the connection as classes that range from 1 Gbit/s to 10 Gbit/s.

1.2.3. Physical Media Dependent Layer (PMD)

The physical layer specification has been made according to [5].

1.2.3.1. Line rates

The baseline configuration is 10 Gbit/s downstream and 2.5 Gbit/s upstream, but there are also options that allow symmetric line rates of both 2.5 and 10 Gbit/s, a table with possible combinations is presented in Table 1.1. This may be very useful for business/backhaul applications which require a higher upstream capacity.

Table 1.1: Nominal line rate combinations in TWDM PON. [23]

Downstream line rate (Gb/s)	Upstream line rate (Gb/s)
2.48832	2.48832
9.95328	2.48832
9.95328	9.95328

The baseline line rate is constantly searching for further improvements and a 25 Gbit/s downstream capacity appears to be available in the near future [13]. Considering the four channels it would increase the overall capacity of NG-PON2 to 100 Gbit/s.

1.2.3.2. Wavelength plan and agility

The wavelength plan for NG-PON2 needed to use the available unused spectrum and was the best comprise for co-existence with G-PON, XG-PON1 and RF Video, ease of filtering, availability and development of C/L-band components and minimizing the Raman fiber impairments.

Wavelength compatible systems	TWDM PON		PtP WDM PON
	Downstream	Upstream	Upstream/downstream
GPON, RF video, XG-PON1	1596-1603 nm	Wideband option 1524-1544 nm Reduced band option 1528-1540 nm Narrow band option 1532-1540 nm	Expanded spectrum 1524-1625 nm (Note 1) Shared spectrum 1603-1625 nm (Note 2)

Figure 1.4: NG-PON2 wavelength plan. [5]

As can be seen in Fig. 1.4 it was specified three upstream wavelength band options for TWDM-PON. These differences come from the different capabilities of the ONUs transmitter to control its wavelength.

As for the PtP WDM PON there is the *Shared spectrum* when there is a scenario of full coexistence with legacy PON systems and the *Expanded spectrum* that takes advantage of the spectral flexibility present in NG-PON2 and enables bands of the spectrum that are not used in a particular deployment to be utilized by PtP WDM.

Spectral flexibility is one of the key features of NG-PON2, it can facilitate the support of different customer types on the same ODN as well as allowing a range of system coexistence scenarios and enables operators to use new wavelength bands when legacy systems are decommissioned.

The tunability of the ONU transceivers allows for new use cases that are opened by the existing wavelength agility. The Internet traffic shows diurnal patterns which can be used by NG-PON2 to realize network power savings. For example, during times of low traffic load all ONUs on an ODN can retune to a common wavelength and allow for unused OLT ports to be powered down [8], also known as OLT port sleep.

Wavelength agility can also be used for network protection, resilience and service restoration. If a wavelength port fails at an OLT, ONUs can retune to a predefined or randomly assigned remaining wavelength port. In case that there is full loss of connectivity to a working OLT, all ONUs can retune to a single wavelength providing a basic service on a protection OLT on a diverse location.

1.2.3.3. ODN compatibility and additional options

According to the network operators' requirements it was necessary to re-use the already existing PON fiber infrastructure so the NG-PON2 PMD has been defined to be compatible with power splitter based ODNs. This plus the need to coexist with legacy PON systems leads the NG-PON2 standard to use the same nomenclature and values to the classes for optical path loss as were used for XG-PON1 as seen in Fig. 1.5 (a) where N1, N2, E1 and E2 refer to different classification of routes, the letters to the type of LSA (Link-State Advertisement) being the E for type 5 that corresponds to autonomous system external LSA and the N for type 7 that refers to not-so-stubby areas (do not have external routes) and the numbers refer to wether internal costs of the route are taking into account or not, in case of 1 they are and in case of 2 they are not. Also the 40 km fiber distance difference [9] from the OLT to the furthest and closest ONUs respectively is also to be supported in NG-PON2

In a Greenfield scenario [1] NG-PON2 could use wavelength splitting which would have the advantage of reducing splitting loss which may translate into extended system reach or permit the use of lower specification transceivers. Furthermore, some services would be attracted to the physical channel isolation offered by wavelength splitting which would offer virtual private line services using wavelength channels.

Even though the same ODN is used the optical path penalty (OPP) is slightly different for NG-PON2 having a new OPP factor that was not relevant in the previous PON generations, the Raman depletion, which can have an impact as large as a 1.5 dB or so power reduction

[1] A Greenfield project is one that lacks constrains imposed by prior work, as defined in [28].

Class	N1	N2	E1	E2
Minimum Loss (dB)	14	16	18	20
Maximum Loss (dB)	29	31	33	35

(a)

Tuning Time Class	Tuning time
Class 1	$< 10\ \mu$s
Class 2	10 μs to 25 ms
Class 3	25 ms to 1 s

(b)

Figure 1.5: (a) Classes for optical path loss in NG-PON2. (b) Wavelength channel tuning time classes in NG-PON2. [2]

at the OLT receiver input.

1.2.3.4. Incremental upgrade

NG-PON2 systems support an incremental upgrade capability, sometimes called ”pay-as-you-grow”, that enables network operators to defer capital costs and meet network capacity growth in a cost effective way. In reality, what this means is that the network operators can add a channel one-by-one to the system. Since the ONUs are colorless and can tune to any NG-PON2 channel, they may be redistributed when new channels are added for load balancing purposes or the newly added channels can simply be used to provide a new service.

1.2.3.5. Transceiver technologies

The most notably new optical components in NG-PON2 are the tunable receivers and tunable transmitters at the ONUs meaning they can selectively transmit (receive) upstream (downstream) channels.

For the tunable transmitter there are many possible components based on already existing and commercially available technologies. The simplest candidate is a thermally tuned DFB laser with either heating only control or both heating and cooling [10]. Whereas a more complex option is the multi-section DBR laser either with or without cooling [2].

As for the tunable receiver the maturity of the available products is not as good as for the tunable transmitter. Nevertheless, there are plenty of technology options, although the thermally tuned Fabry-Perot cavity filter technology seems to have received more attention [11].

According to the different technologies, there are different wavelength tuning times, so it is necessary to establish classes for the wavelength channel tuning time of the ONU transmitter and receiver in NG-PON2 as shown in Fig. 1.5 (b). Class 1 components may consist of switched laser or filter arrays, Class 2 may be devices based on electronically tuned lasers (DBR) and Class 3 could be thermally tuned DFBs. Since technology implementations are not yet defined in the standards these are only examples [2].

According to the class of each device the technological options enabled by the wavelength tunability may vary, e.g., advanced power saving [8] and dynamic wavelength assignment [12] that may require different tuning speeds.

1.2.4. Transmission Convergence (TC) layer

The transmission convergence layer is the protocol layer of the NG-PON2 system that is positioned between the physical media dependent (PMD) layer and service clients. It builds on the XG-PON recommendations [22], with modifications for NG-PON2 specific features. This section was made according to [23].

A NG-PON2 system may contain a set of TWDM channels, a set of PtP WDM channels or both, each technology uses a different TC layer and will be described in the next subsections.

1.2.4.1. TWDM transmission convergence layer

The TWDM TC layer is a part of the TWDM PON protocol stack that specifies the formats and procedures of mapping between the upper layer SDUs and a bitstream suitable for modulating the optical carrier. The TWDM TC layer is composed of three sublayers: the TWDM TC service adaptation sublayer, the TWDM TC framing sublayer and the TWDM TC PHY adaptation sublayer. It is also responsible for functions such as the performance monitoring, security key management, ONU power management, TWDM channel management and protection, as can be seen in Fig. 1.6.

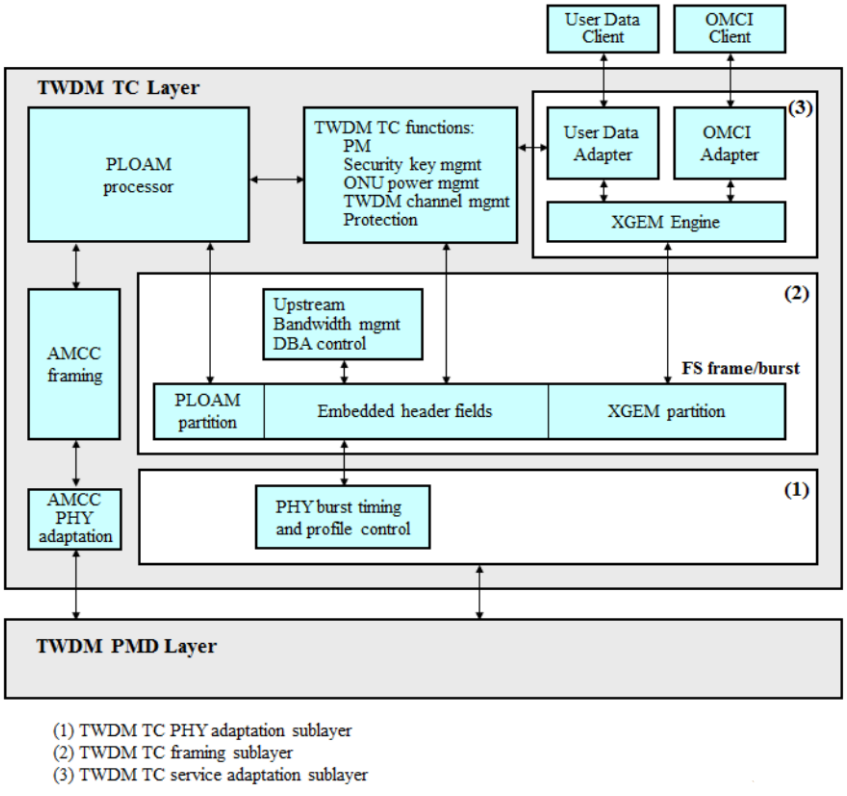


Figure 1.6: Outline of TWDM TC information flow. [23]

The TWDM TC service adaptation sublayer is responsible for the upper layer SDU (Service Data Unit) encapsulation, multiplexing and delineation. It deals with two types of SDUs, user data frames and OMCI (ONU Management and Control Interface) messages, so it can be logically decomposed in a XGEM (XG-PON Encapsulation Method) engine and

two service adapters, one for the user data and the other for the OMCI messages. The user data adapter can also be configured to acomodate a variety of upper layer transport interfaces.

The TWDM TC framing sublayer is responsible for the construction and parsing of the overhead fields that support the necessary PON management functionality. The TWDM TC framing sublayer formats are devised so that the frames, bursts and their elements are aligned to 4-byte word boundaries, whenever possible.

The TWDM TC PHY adaptation sublayer encompasses the functions that modify the bit-stream modulating the optical transmitter with the goal to improve the detection, reception and delineation properties of the signal transmitted over the optical medium.

The ONU control, operation and management information in a TWDM PON system is carried over three channels: embedded OAM (Operation, Administration and Maintenance), PLOAM (Physical Layer OAM) and OMCC (ONT Management and Control Channel). The embedded OAM and PLOAM channels manage the functions of the PMD and TWDM TC layers. The OMCC carries the messages of the OMCI protocol, which provides a uniform system for managing higher (service-defining) layers.

In a NG-PON2 system the OLT channel terminations need to interact with each other, this is possible due to the ICPT (Inter-Channel-Termination Protocol) transportation channel. It allows the support of functionalities such as channel profile configuration and status sharing, ONU activation, ONU wavelength channel handover and rogue ONU mitigation.

1.2.4.2. PtP WDM transmission convergence layer

This clause describes the TC layer for PtP WDM channels. In PtP WDM, the user data path and the OAM data path are separate through the TC layer. The user data is unprocessed by the TC layer, while the OAM data is processed by the PtP WDM AMCC TC layer.

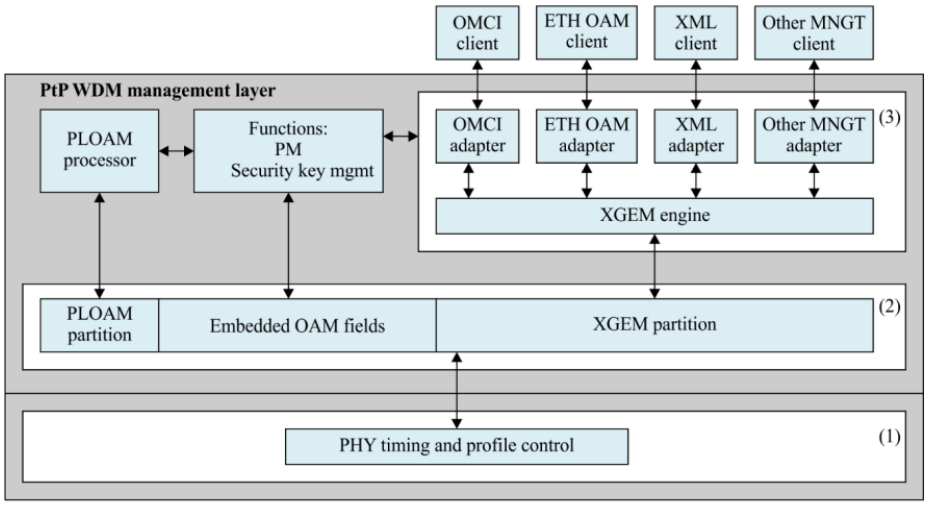


Figure 1.7: Outline of AMCC TC management information flow. [23]

There are two distinct modes of PtP WDM channels: transparent and transcoded. In transparent mode, both the user data and the AMCC TC data are passed directly to the

PMD. In transcoded mode, the user data and framed AMCC data are passed to the two interfaces of the transcoder. The transcoder multiplexes and processes the two streams, and passes the result to the PMD.

The AMCC TC is implemented at both the OLT and ONU sides of a PtP WDM system. In both downstream and upstream directions, it interfaces with the PMD or the transcoder as a continuous bitstream, which is partitioned into cyclic frames. The downstream and upstream frames have the same format.

The AMCC TC layer is composed of three sublayers, just like the TWDM TC layer: the AMCC TC management service adaptation sublayer, the AMCC TC framing sublayer and the AMCC TC PHY adaptation sublayer. The sublayers functions are very similiar to their TWDM TC counterparts and are shown in Fig. 1.7.

1.3. Service Interoperability in EPON (SIEPON)

The Service Interoperability in Ethernet Passive Optical Networks (SIEPON) working group proposed the IEEE 1904.1 standard, which created an open international system-level specifications, targeting ”plug-and-play” interoperability of the transport, service, and control planes in a multi-vendor environment.

SIEPON ensures that data services can be managed, provisioned and monitored across the EPON/GPON. The scope of SIEPON standardization has not been limited to the physical (PHY) and media access control (MAC) layers, it has expanded to the upper layers in order to support all functions of data path such as multicast, tunneling and VLAN modes, and QoS services [36].

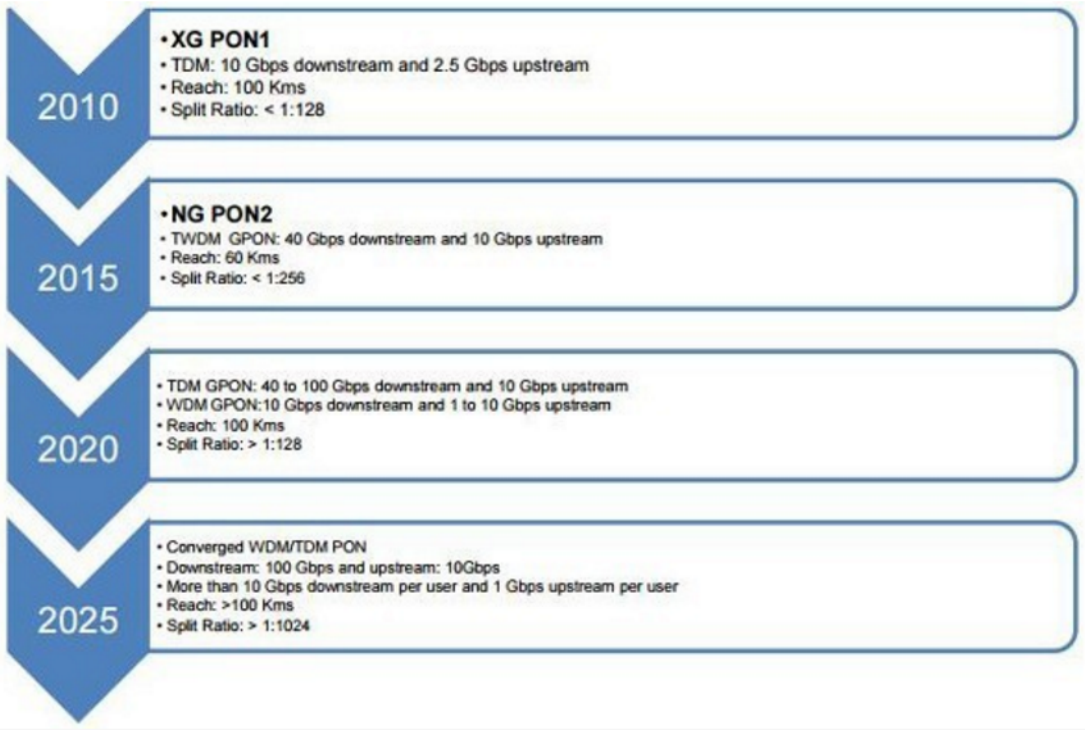


Figure 1.8: Beyond NG-PON2 as envisioned by Analysys Mason, 2009. [27]

1.4. Summary

This chapter has described the main standards that need to be taken into account and followed when developing and working with technology related to NG-PON2.

NG-PON2 uses 4 λ-channels and even though in some cases it was possible to take further advantage of that feature (e.g. λ-splitters). In order to be adopted and economically affordable it needs to re-use most of the already deployed network as well as coexist with the previous PON systems without disturb or disrupt them. Being this the main reason behind most of the standards and its limitations.

The key technology to enable most of the functionality of the NG-PON2 is tunable transceivers and receivers, although there is some availability on the market to enable the success of NG-PON2 it will be mandatory that these devices have further development and become more affordable (lower prices) since they will be deployed massively at the end user premises.

CHAPTER 2. SDN AND OPENFLOW

Software Defined Networking (SDN) is about separating the network's control and data planes so that the control plane becomes open and programmable whereas OpenFlow is the de-facto standard interface between SDN controllers and devices. The necessary fundamentals of both are going to be explained in this chapter.

2.1. Software Defined Networking

SDN is a paradigm that decouples the control and data planes of network equipment, and concentrates the intellegence and complexity of devices in a centralized controller which has a global view of the state of the network [14]. The centralized controller handles the traffic control, switching, routing and Quality of Service (QoS) and a more efficient control of the network is achieved. Software Defined Networking has been establishing itself as a vital part of the Internet environment and is now used in a huge range of different technologies.

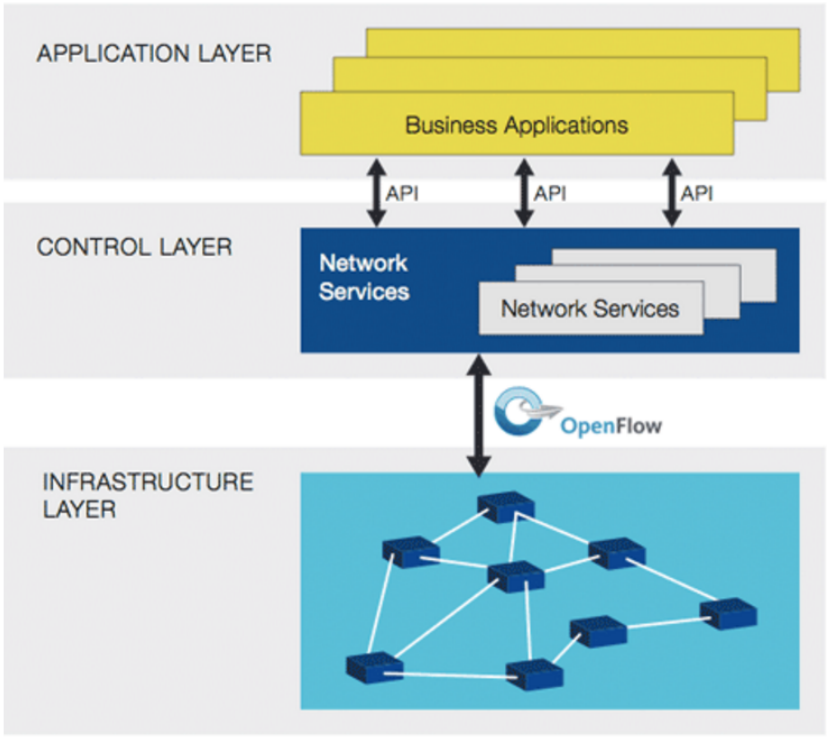


Figure 2.1: SDN archictecture. [15]

2.1.1. Layers

The SDN archicture is separated in three different layers, the Application, Control and Infrastructure layer as can been seen in Fig. 4.1.

Each application communicates with the controller (control plane) through an API (Application Programming Interface), which may vary according to the controller, making it

possible to execute decisions of package forwarding, traffic handling, and QoS policies in real time. The communication between the control and infrastructure layer is made through the OpenFlow standard which will be discussed further in this work.

2.1.1.1. Application Layer

The application layer is composed of network applications that communicate with the control layer, making requests for network resources or some changes in the network itself resulting in a certain behaviour and QoS for a specific traffic or equipment.

As explained before the communication is made through APIs, the most common examples are the NorthBound Interface (NBI) and the Java API, that carry synchronous and asynchronous messages. There can be multiple network applications communicating with the controller at the same instant, handling different types of traffic using the network infrastructure as a shared resource and providing a wide flexibility of functions. It is also possible to establish paths with certain SLAs (Service Level Agreement) according to the user's requests.

2.1.1.2. Control Layer

The control layer is composed by the SDN controller, it is logically centralized and is in charge of translating the requirements from the application layer down to the infrastructure layer. It provides the applications an abstract view of the network topology, an inventory of the features supported by each network device, network statistics, host tracking information and packet information [16]. Applications can then make decisions accordingly and the controller translates those decisions into instructions that will be downloaded and executed by the network elements.

The control layer is also responsible for creating virtual networks or slicing part of the network resources that will be later used by service providers and end users, minimizing its' access to the remaining network resources.

2.1.1.3. Infrastructure Layer

The infrastructure layer comprises the data plane of the SDN architecture, composed by forwarding devices like routers and switches.They receive instructions from the control layer and forward the packets accordingly. The forwarding devices maintain a constant communication with the control layer keeping the system updated on their network status like active ports, adjacent neighbors, supported features, counters information, and event notifications [16].

Logically, the devices are viewed as an SDN Datapath and identified by an Datapath ID (DPID). An SDN Datapath countains a CDPI (SDN Control to Data-Plane Interface) agent, usually refered as SouthBound Interface, and a set of one or more traffic forwarding engines and a zero or more traffic processing functions. These engines and functions may include simple forwarding between the datapath's external interfaces or internal traffic processing or termination functions.

OpenFlow is the de-facto standard for the SouthBound Interface, being accepted by most

vendors and in constant development.

2.1.2. Benefits and Challenges

The SDN architecture has many benefits, thus explaining its fast grown and success in the industry over the years. It began focused on Datacenters and core networks and is now expanding to the optical access networks. Despite its success there are still some limitations and challenges that the SDN networks need to overcome.

The main benefits are a significant reduction of the network configuration time for both deployments and application changes, a improvement in the network efficiency allowing for a greater functionality in the areas of bandwidth management, QoS conditioning and a lowered risk in configuration errors (through automation of certain functions taking out the human error factor), and a multivendor network control since the communication with the control layer is made using an open protocol, OpenFlow.

The limitations and challenges to resolve are: the velocity of the cloud services that generates a bottleneck when provisioning to external network elements, because even though the assignment and deployment of virtualized compute or storage takes a matter of minutes, the network connections between those virtual machines and the end users takes far more time to provision; the lack of a NorthBound standard to facilitate the communication between the control and application layer (in the SouthBound Interface we have OpenFlow); and the automation which can be very helpful, but needs to be well supervised in order to not commit errors.

2.2. Openflow

OpenFlow is the first standard communications interface defined between the control and forwarding layers of an SDN architecture, it allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual.

2.2.1. Openflow Standard and Versions

An OF enabled device communicates with a OpenFLow Controller (OFC) that is capable of, in software, creating rules to modify how the switch data-plane works. OpenFlow is based on a Ethernet switch, with an internal flow-table, and a standardised interface to add and remove flow entries [1].

The OpenFlow protocol is divided in two parts: wire protocol and configuration and management protocol. The wire protocol is responsible for establishing control sessions, defining message structures for exchanging flow modifications, collecting statistics, and defining the fundamental function of a switch (ports and tables). Whereas the configuration and management protocol allocates physical ports to a controller, and define high availability and actions in case of a controller connection failure [16].

It is layered on top of the Transmission Control Protocol (TCP), and prescribes the use of

Transport Layer Security (TLS). Controllers should listen on TCP port 6653 (or 6633) for switches that want to set up a connection.

To summarize the key features supported by current OpenFlow versions Table 2.1 was made.

Table 2.1: OpenFlow versions. [16]

OpenFlow version	**Key Features**	**Release**
OF 1.1.X	MPLS, Vlan, multipath, group tables and multi-flow tables	February 2011
OF 1.2.X	extensible headers, flexible flow match and IPv6	December 2011
OF 1.3.X	tunneling, meter tables, Provider Backbone Bridging (PBB) and additional fields on flow tables	June 2012
OF 1.4.X	optical port management, synchronized tables, flow monitoring and eviction	October 2013
OF 1.5.X	egress tables, extensible flow entries statistics, packet type aware pipeline, TCP flags matching and scheduled bundles	December 2014

2.2.2. Openflow Switch

In a classical router or switch, the fast packet forwarding (data path) and the high level routing decisions (control path) occur on the same device. An OpenFlow Switch separates these two functions. The data path portion still resides on the switch, while high-level routing decisions are moved to a separate controller, typically a standard server [15].

The data path of an OpenFlow Switch presents a clean flow table abstraction; each flow table entry contains a set of packet fields to match, and an action (such as send-out-port, modify-field, or drop). When an OpenFlow Switch receives a packet it has never seen before, for which it has no matching flow entries, it sends this packet to the controller. The controller then makes a decision on how to handle this packet. It can drop the packet, or it can add a flow entry directing the switch on how to forward similar packets in the future [15].

2.2.2.1. CPqD Switch

The switch used in this project was the CPqD switch, also known as *ofsoftswitch13*, it was developed by Ericsson Innovation Center in Braziland formerly maintained by CPqD in technical collaboration with Ericsson Research. Its code is based on the Ericsson TrafficLab *1.1softswitch* implementation and has almost all features from the OpenFlow 1.3, it is simpler than the OVS Switch and destinated to experimentation purposes.

It was chosen due to its support of the standard IEEE 1904.1 SIEPON which allows us to configure flows with a specific QoS, preserving that QoS at low level by managing the

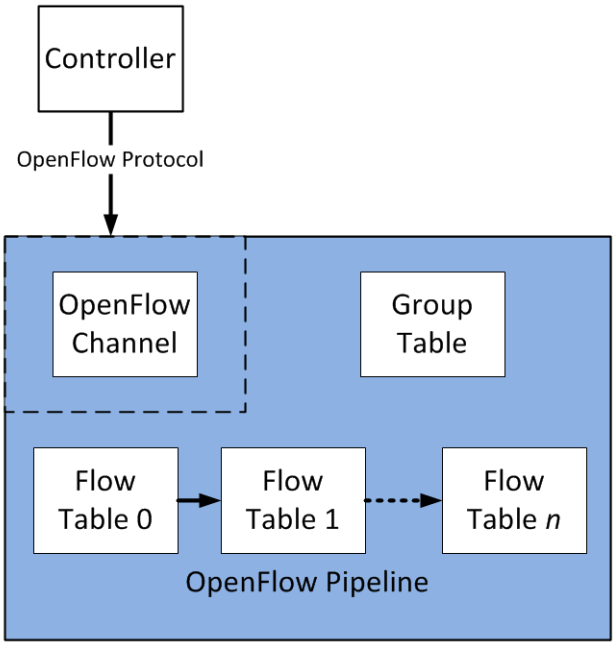


Figure 2.2: OpenFlow Switch.[18]

wavelengths and time-slots properly, that being one of the key features of the project in which this work is integrated.

2.3. ONOS

The Open Network Operating System (ONOS) is a software defined networking (SDN) OS for service providers that has scalability, high availability, high performance and abstractions to make it easy to create apps and services. The platform is based on a solid architecture and has quickly matured to be feature rich and production ready. It was developed by Open Networking Lab (ON.Lab), a nonprofit organization dedicated to create tools and platforms and building open source communities to realize the full potential of SDN [26].

ONOS is a open source system and in the spirit of freedom and openness, its logo is a bird as well as the release names (Avocet, Blackbird and Cardinal, Drake, Emu and Falcon). It is currently on the fifth release, Falcon.

ONOS brings carrier grade features (scale, availability, and performance) to the SDN control plane, enables Web style agility, and helps service providers migrate their existing networks to white boxes. All this actions allow for the service providers to significantly lower their CAPEX and OPEX.

2.3.1. Architectural features of ONOS

There are some architectural features that distinguish ONOS from the other SDN controllers [26], making it one of the best choices and one of the main reasons why it was

chosen for this work, the other reason being its support of optical devices where other SDN controller lack.

2.3.1.1. Distributed core

ONOS is deployed as a service on a cluster of servers, and the same ONOS software runs on each server. Deployment symmety is an important design consideration and enables rapid failover in the event of an ONOS server failure. The network operator can add servers incrementally, without disruption and the instances work together to create what appears to the rest of the network and applications a single platform. Network devices and applications do not know if they are working with a single instance or multiple instances of ONOS. This feature is responsible for the high scalability of ONOS.

2.3.1.2. Northbound abstraction/APIs

In ONOS there are two powerful northbound abstractions: The Intent Framework and the Global Network View.

The Intent Framework allows an application to request a service from the network without having to know details of how the service will be performed. Examples: Set up an optical path from switch A to switch B with X amount of bandwidth; Do not allow host A to talk to host B.

The Global Network View provides the application with a view of the network from the hosts, switches and links to states associated with the network such as utilization. An application can program this network view through APIs. An example can be seen in Fig. 2.3.

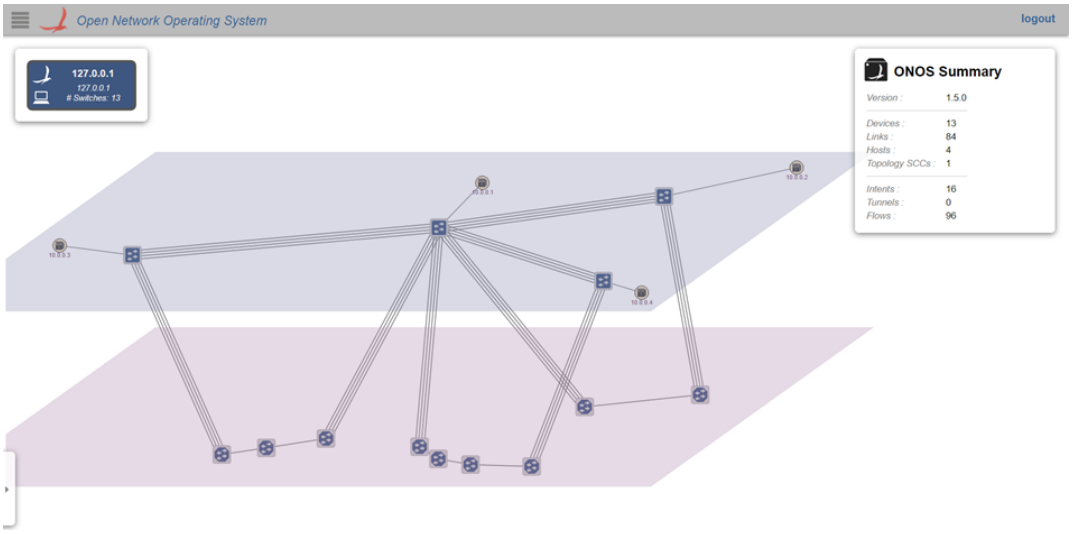


Figure 2.3: Example of an optical and IP network represented in ONOS GUI.

2.3.1.3. Southbound abstraction/APIs

The southbound abstraction is built using network elements, such as switches, hosts, or links. ONOS represents each network element as an object in a generic form. Through

this the distributed core an maintain the state of the network element without knowing the specifics of the element represented by the underlying driver.

The main benefits of this abstraction is the ability to manage different devices using different protocols, without effect on the distributed core of the system, and the ability to add new devices and protocols to the system.

2.3.1.4. Software Modularity

The ONOS team put a great care into modularity to make it easy for developers to work with the software. Modularity is how the software is structured into components, modules, and how those components relate to one another.

Software modularity allows for and architectural integrity and coherence, simplified test structure, easier maintenance with fewer side effects of changes, extensibility and customization of components, and avoidance of cyclic dependencies.

2.4. Summary

As detailed is this chapter SDN and OpenFlow are intrinsically connected. OpenFlow is the standard that helped solidify the grown and success of SDN, being considered a kind of SDN enabler.

SDN is a concept that separates the data and control plane and can be considered a disruptive technology. It is still growing at a fast rate being adopted by pretty much all manufacturers and changing network deployments around the world.

OpenFlow is the standard that allows the communication between the SDN devices and the SDN controller, it facilitates and translates the orders of the controller to the device and it allows the existance of a multivendor infrastructure since they all use the same standard.

The OpenFlow Switch it is the main network device for this work, being the one connected to the controller and exchanging OpenFlow messages. In this work it was chosen the CPqD Switch due to its support of meters.

The chosen controller was ONOS due to its architectural features and its support for optical devices and optical intents which other SDN controllers lack. It was also proven to be able to support a virtualized OLT, which is one of the objectives for this work.

CHAPTER 3. PROBLEM STATEMENT AND PROJECT GOALS

In this chapter it will be discussed the main motivations behind the work done and it will be provided a brief overview of similar projects that interconnect the optical access networks and software defined networking.

3.1. Problem statement

Nowadays need for higher bandwidth, demanding QoS parameters and lower power consumption leaves optical technologies as the main solution for access networks, more precisely passive optical networks (PONs).

As explained previously PONs have been evolving at a fast rate and there are two main technological options, the EPON (IEEE standard) and GPON (ITU-T standard). In this work it was decided to follow the GPON standard, in fact its latest version NG-PON2 described in Chapter 1.

NG-PON2 introduces a new set of variables that needs to be managed, being the most notorious the four wavelengths that can work simultaneously, the possibility of having a point-to-point wavelength connection (PtP WDM) and the tunability of the ONUs. To take full advantage of this new system, it is necessary to correctly manage and control these new parameters as well as optimize the previous ones, like the QoS of each link, fault tolerance, security, etc.

In the NG-PON2 standard the TC module is responsible for the bandwidth management and DBA control, meaning it is in charge of assigning time slots and wavelengths. But the ITU-T G.989.3 (TC layer specification) does not standardize the bandwidth allocation algorithm, so one of the goals of this project is to create a effective algorithm to optimize the allocation of wavelengths and its time slots, therefore, optimizing the bandwidth allocation.

As explained in the last paragraph it is necessary to create a new bandwidth allocation algorithm or adapt an existing one in order to achieve one of the project goals and to take the aforementioned full advantage of the NG-PON2 system. But, to see which one is the best fit according to different sceneries and to test those different scenaries and algorithms it is also required an emulated environment, which became one of the goals of this work.

The advantages and success of SDN lead to a possible expansion from Data Centers and core networks to the access network and therefore the optical access network. The centralization required is easily obtained since optical access networks are centralized by nature and the agility provided to service providers is very appealing.

More specifically, when the access technology is NG-PON2 where we have ONUs that can tune to multiple wavelengths, four λs in each OLT and the possibility of creating specific wavelength channels that cross the main network without being disrupted (PtP WDM) it makes the option of having an SDN controller with a general view of the entire network more and more advantageous.

Taking into account the NG-PON2 technology and all its features plus the emerging trend to extend SDN to the optical access networks the focus of the project became the creation of an environment where the different management algorithms for the NG-PON2 system

could be created and tested, all this with the overview of an SDN controller, which sees the OLT as an OpenFlow Switch belonging to the core network, easily managed and in sync with the rest of the devices as well as a scheduling algorithm to optimize the use of resources, the QoS, the energy consumption and test the created environment.

To conclude, the goal of this work is to contribute to the aforementioned research effort by creating a fully SDN and NG-PON2 emulation environment where a set of different optimization algorithms can be created and tested, we then propose a scheduling algorithm that optimizes the use of channels and wavelengths in terms of QoS and energy as well as testing the environment created.

3.2. Adapting SDN to the access - state of the art

When it comes to adapt SDN to the already deployed access networks, there is a particular problem that still remains. For logistical reasons, the hardware for access networks often remains in place for a long time. Much of the hardware is located in the street-side cabinets and sometimes user premises which means that the replacement would not only be buying new equipment, but also getting access to on-street locations and perhaps customer premises. So it was needed an approach that can get the technology working at line rate using existing hardware with minimal hardware replacement.

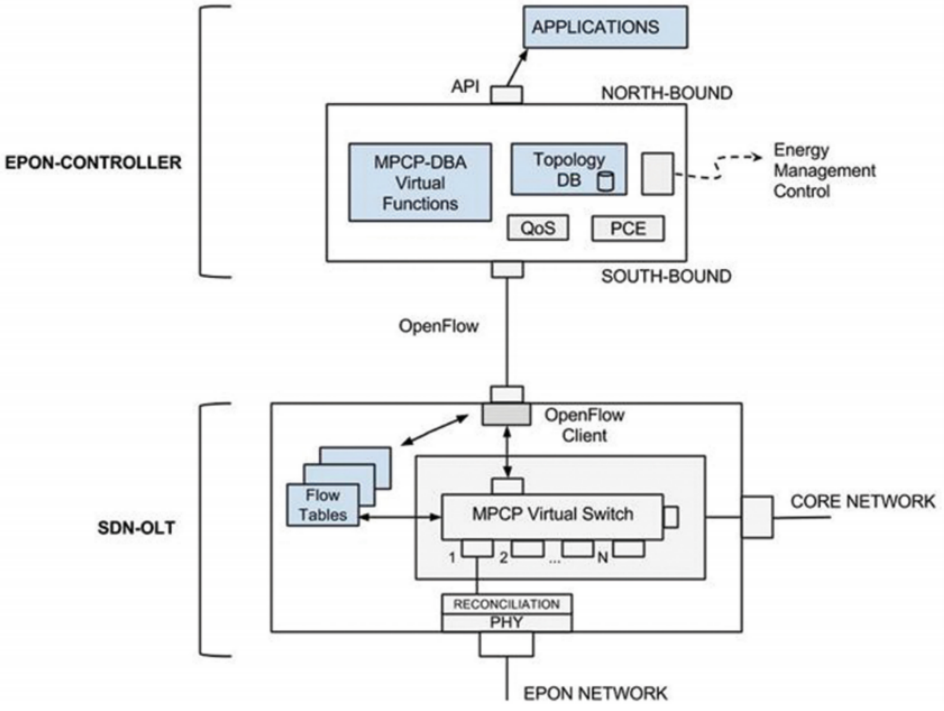


Figure 3.1: SDN-EPON architecture. [17]

Accordingly, in [20], which focuses on GPON and B2B (Business To Business) applications, it proposes an extension to OpenFlow in order to provide traffic mapping and forwarding capabilities, while keeping its original functionality, architecture and capabilities. Each OLT and ONU in the GPON contains flow tables and communicates over a secure

channel with the controller via the OpenFlowPLUS protocol, this extension of the OpenFlow proposes three more actions all of them related to the mapping of Ethernet frames.

Since in access networks there is a tree topology and all the traffic needs to pass through its root (head-end switch), even if the communication is between the leaves of the tree, it facilitates an opportunity to modify the traffic to be OpenFlow enabled at the root, which corresponds to the OLT so an SDN-OLT is one of the focus when trying to adapt PONs to SDN.

An SDN-EPON architecture is proposed in [17], where they take an OpenFlow-enabled switch as a base, then add the EPON functionalities to the OLT. There are three elements the SDN-OLT (data plane), EPON-controller (control plane) and the EPON network as can be seen in Fig. 3.1. The forwarding functions are kept within the SDN-OLT switch and moving the management functions to the EPON-controller. It is also proposed splitting the functionality of the MPCP sub-layer of the OLT in the high level policies, which are migrated to the centralized controller, and the data plane functions that remain in the SDN-OLT. To make this possible it was identified the need to extend the OpenFlow protocol.

An SDN-based architecture for optical access networks by applying a centralized optical wavelength assignment mechanism that transfers to the OLT functionalities that are typically implemented in the ONUs is proposed in [19]. It allows full virtualization of the optical spectrum space, making so that arbitrary wavelengths be available on-demand through software reconfigurable control allowing for a virtualization of physical upstream and downstream connections as logical flows between bidirectional OpenFlow port identifiers.

The authors of [21] propose an integrated management architecture for SDN-based access, metro and core networks. The access segment is enhanced with OpenFlow-enabled OLTs that pass bandwidth assignment information to the Metro segment and the Core segment. Then a Generalized SDN controller is responsible for the control planes of the entire network, thus being able to coordinate and optimize the end-to-end performance.

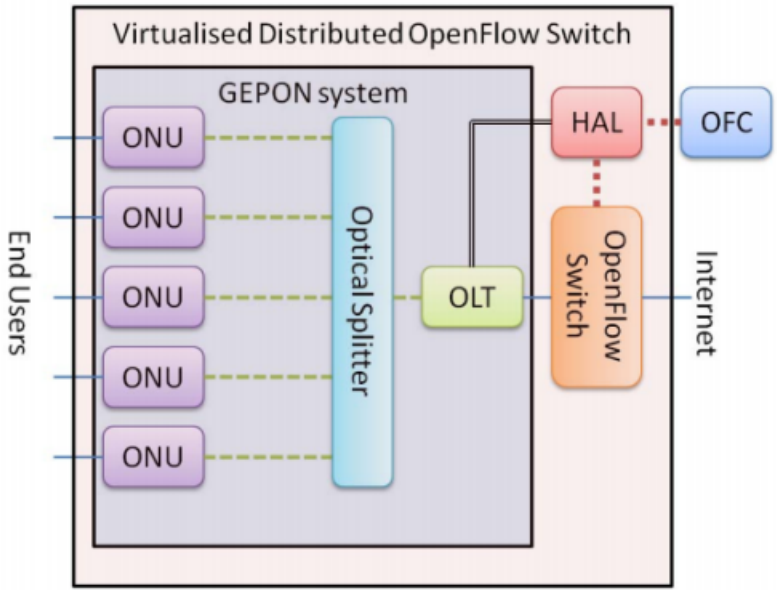


Figure 3.2: Architecture used in [1] to allow GEPON to use OpenFlow.

In [1] it was proposed and architecture to make the GEPON system OpenFlow enabled,

this was achieved by having an intermediate controller which intercepts and modifies the OpenFlow messages and it can be seen in Fig. 3.2. The key features of this proposed architecture is the OpenFlow enabled switch outside the OLT, the extra helper box labelled HAL (Hardware Abstraction Layer) and that the OLT management port is now connected to the HAL. The HAL acts as an intermediate between the GEPON switch and the OFC (OpenFlow Controller), it ”translates” the messages and commands from one system to the other and vice-versa. The OFC will see the OLT and ONUs as a simple distributed OpenFlow Switch with many ports. Although this solution was designed for GEPON system it can easily be adapted to other PON systems.

When it comes to the virtualization of the OLT and its functionalities there is one project that is leading the way, the project is called CORD (Central Office Re-architected as Datacenter) and is a collaborative effort of AT&T and Open Networking Lab. Although it does not target specifically the virtualization of the OLT it is one of the key points of the project, demostrations were already made with GPON and G.fast.

CORD is an architecture for the Telco Central Office that builds from existing work on SDN, NFV and commodity hardware in order to build a cost-effective, agile networks with significantly lower CAPEX/OPEX and to enable rapid service creation and monetization [24].

In order to re-architecture the central office as a datacenter it was necessary to virtualize the existing hardware devices, transforming each device in its software service counterpart and running it in commodity hardware. The devices virtualized were the Optical Line Termination (OLT), the Customer Premises Equipment (CPE) and the Broadband Network Gateways (BNG).

To create a virtual OLT (vOLT) it was necessary to have commodity GPON interface cards as shown in Fig. 3.3, control functions that set up and manage the control plane functions of an OLT, and host software functions extracted out of the existing OLT [25].

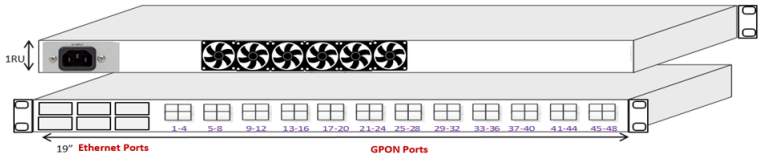


Figure 3.3: GPON OLT IO Blade. [25]

The GPON boards need to have the GPON Media Access Control (MAC) chip under control of a remote control program via OpenFlow, that program is implemented as an application running on top of ONOS and the hosted functions are mainly to set up an SDN agent on the GPON IO blade.

3.3. Summary

In this chapter it was explained the motivations behind the work and the project goals. Those being the deployment of a fully SDN-NG-PON2 emulation environment and a simple scheduling algorithm that optimizes the resources used, the QoS and the energy consumption of the network.

It was seen some works regarding the use of SDN in the optical access networks and how the already deployed networks could adapt to SDN. Most of them agreed that the best device to make this adaption was in the OLT, therefore there is a focus on creating SDN enabled OLT for the different PON technologies.

Finally in the state of the art it was examined one of the leading projects in expanding SDN to the optical access where a virtual OLT was created and already tested in demonstrations for the GPON and G.fast technologies, this means that it might well be used and implemented in the future deployed technologies, such as the NG-PON2.

To the best of our knowledge, no emulation environments like ours have been described in the literature, and apart from the results of the tests, the desing and decription of the environment itself is already an important contribution.

CHAPTER 4. EMULATION ENVIRONMENT

4.1. Protocol layers

As seen previously the NG-PON2 technology has two standardize protocol layers, the PMD layer and the TC layer. In this work we create an environment that emulates the behaviour of the NG-PON2 PMD layer and later we developed an algorithm to manage and optimize wavelengths and time slots at the TC layer level using Raspberry Pis for the DBA emulation.

A scheme is presented in Fig, 4.1 showing the overall architecture of the network, where the devices are located logically in the ONUs and OLT and the control and data plane of the network.

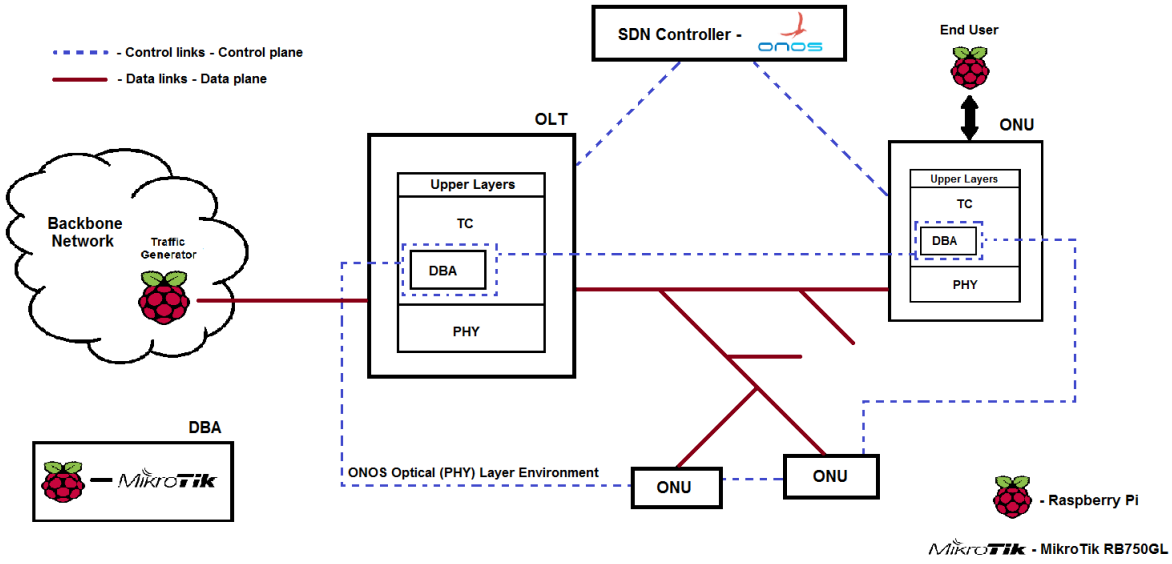


Figure 4.1: Network scheme and protocol layers.

4.2. Topology

The topology chosen for this environment was the same present in passive optical networks represented in Fig. 1, the tree topology. This topology has the characteristic of having multiple users using the same channel, in the downstream all messages are broadcasted and in the upstream there needs to exist a Medium Access Control (MAC) so that there are no collisions between the different users in the shared network. This topology is highly scalable by adding branches to the already existing tree and easily managed and maintained, partly due to its centralized nature.

In our case there are no optical devices due to its elevated cost, instead we use low cost but effective devices such as the MikroTiks and Raspberry Pis that emulate the OLT-ONU bandwidth management modules.

In our network there are three MikroTiks RouterBOARD 750 GL and six Raspberry Pis 1 model B. The MikroTiks represent the control plane where as the Raspberry Pis represent the data plane network. The reason why only two ONUs are simulated is due to the number of available ports on the MikroTik present in the OLT, only 5, to have a proper environment for NG-PON2 there should be at least four ONUs, one for each wavelength, but regardless if the MikroTik present in the OLT is changed by one with more ports it can easily be added more ONUs following the same layout. A scheme of the created topology can be seen in Fig. 4.2.

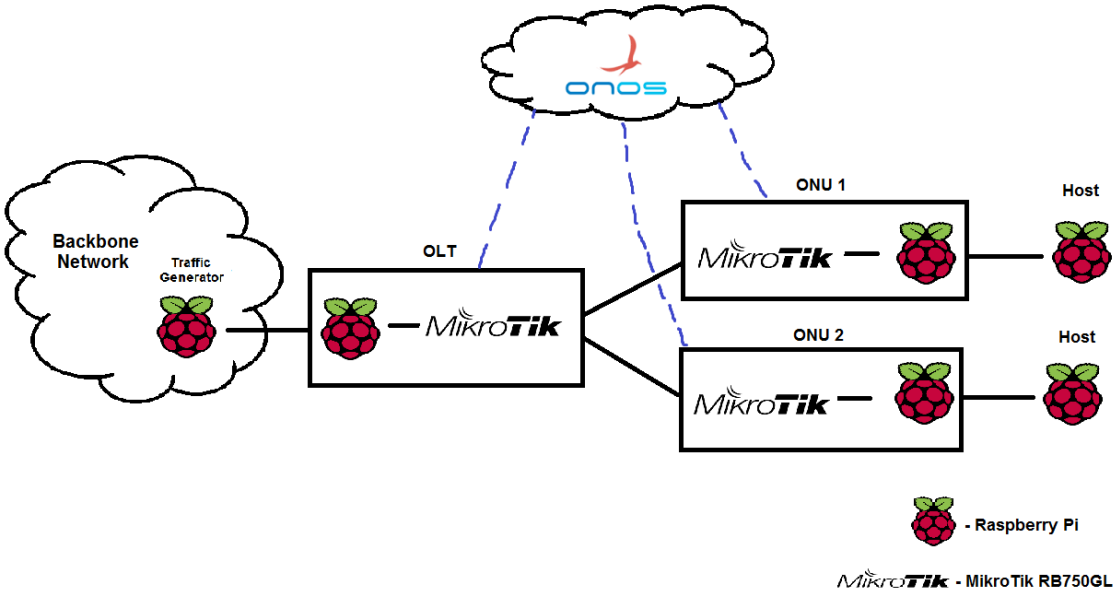


Figure 4.2: Scheme of the architecture created.

To remotely access and configure each device of the network it was necessary to create two different networks, the control plane and data plane, and assign an access point to each of them through a public IP. One of the networks, the 192.168.88.xx network, has MikroTiks and the ONOS controller. It is accessible through the public IP, 147.83.118.81, which is set in the same PC as the one containing the SDN controller. The remaining network, the 192.168.89.xx network, is composed of all the Raspberry Pis of the network. It is accessible through the public IP, 147.83.118.85, which is set in the Raspberry Pi that is shown in Fig. 4.2 labelled as *"Traffic Generator"*. The representation of the networks as well as the access points can be seen in Fig. 4.3.

The reason why the MikroTiks are in a different network of the Raspberry Pi's "inside" the OLT and ONUs is because they have a direct connection to the controller and in order to successfully connect they needed to be in the same network. However the hosts and Traffic Generator should be in different networks to represent the users and core network, respectively.

The Raspberry Pi's, that serve as intermediaries between the Traffic Generator/Hosts and the MikroTiks, are configured as bridges whose and its main function is to reorder the traffic received according to a specific scheduling algorithm, emulating the DBA. Then the MikroTiks can simply work as FIFO (First In First Out) switches.

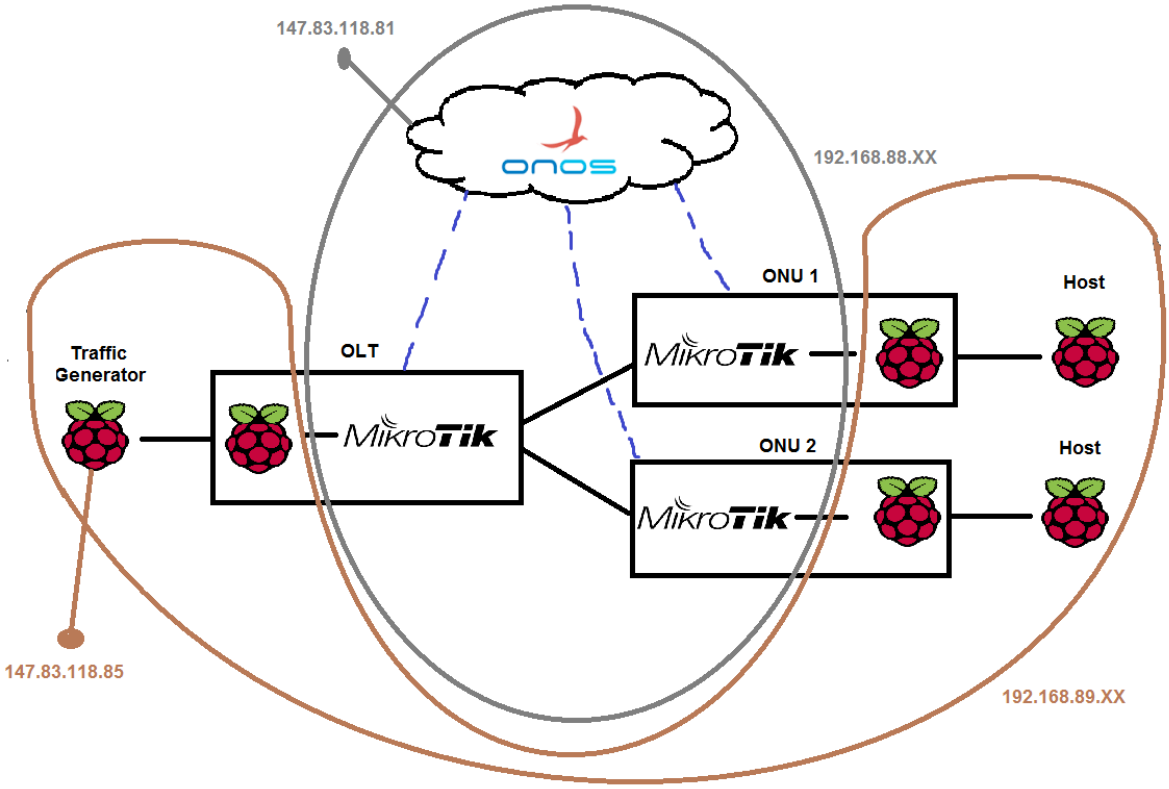


Figure 4.3: Representation of the networks and public access points.

4.3. MikroTik & OpenWrt

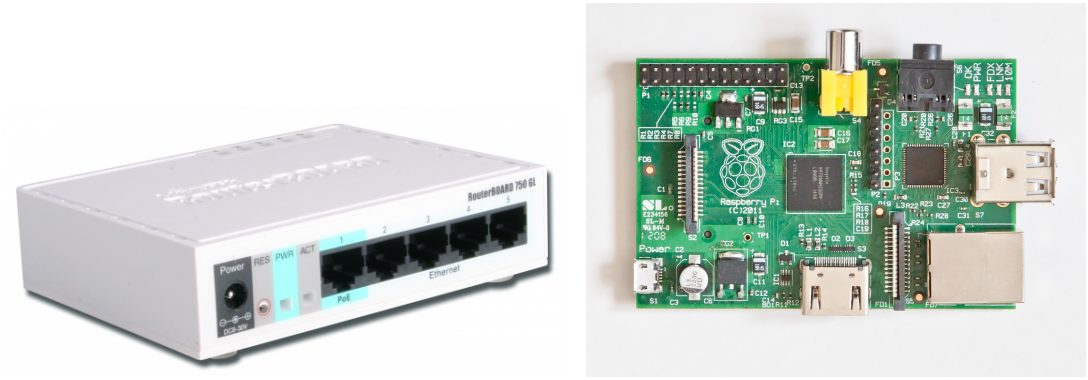
MikroTik is a Latvian manufacturer of computer networking equipment. It sells wireless products and routers.The main product of MikroTik is an operating system based on the Linux kernel, known as the MikroTik RouterOS, but it also manufactures a series of integrated circuit boards, marketed under the name RouterBOARD.

The RouterBOARD line, combined with RouterOS, is targeted at small/medium sized wireless Internet service providers, typically providing broadband wireless access in remote areas. In our case only the RouterBOARD was needed being the RouterOS discarded.

The MikroTik boards used were the model RouterBOARD 750 GL, a picture of which can be seen in Fig. 4.4 (a). In order to work as a CPqD Switch it was needed to remove the MikroTik RouterOS and install the OpenWrt instead.

The OpenWrt, as defined in [29], is an embedded operating system based on the Linux kernel, primarily used on embedded devices to route network traffic. Instead of trying to create a single, static firmware, OpenWrt provides a fully writable filesystem with package management. This allows us to fully customize the device without having to build a complete firmware around it.

We then turned the OpenWrt in an OpenFlow switch, more precisely into the CPqD switch. This was done by another work of the project in which this work is integrated [40], where providing SIEPON was one of the key features therefore requiring CPqD switches. However, there were some configurations that needed to be added, the complete configuration file of the network can be found in the Appendix A.



(a) (b)

Figure 4.4: (a) MikroTik RouterBOARD 750 GL. (b) Raspberry Pi 1 model B.

4.4. Raspberry Pi and Tools

The Raspberry Pi, as defined by [33], is a credit card-sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word processing, browsing the internet, and playing games. It is slower than a modern laptop or desktop, but it still is a complete Linux computer and can provide all expected abilities that implies, at a low-power consumption level and can be bought at a remarkably cheap price. The version used in the project was the Raspberry Pi 1 model B, and can be seen in Fig. 4.4 (b).

In our environment we use the Raspberry Pis for two different functions, 3 of them are hosts/traffic generators while the other 3 work together with the MikroTik in order to simulate an optical device, be it an ONU or an OLT and emulates the DBA algorithms.

All the Raspberry Pis were installed with the Raspbian as its operating system and all needed to disable the DHCP (Dynamic Host Configuration Protocol) and set some static IPs instead in order to be remotely accessed, a necessary feature due to the distance between the laboratory where the network is built (in Barcelona) and the place where the author performed part of the work (in Lisbon).

In the intermediary Raspberry Pis it was also needed to install the package *bridge-utils* in order to bridge the two interfaces (one connected to the MikroTik, the other connected to the other Raspberry Pi) and establish a link between the hosts and MikroTiks, thus having a fully connected network. The configurations needed to make and the files changed are in Appendix B.

There was some extra software needed for the environment work as expected, the main software needed in each type of Raspberry Pi is now going to be listed and described in the next subsections.

4.4.1. MGEN

MGEN is a open source software that provides the ability to perform IP network performance tests and measurements using TCP and UDP/IP traffic. This tool generates real-

time traffic patterns so that the network can be loaded in a variety of ways and with different traffic profiles [34].

It is used in the hosts/traffic generator to generate traffic and emulate the desired behaviour from our environment, creating different types of traffic at different instances with various distributions of speed as well as the protocols used.

4.4.2. Iperf

IPerf is a popular network tool that was developed for measuring TCP and UDP bandwidth performance. It is possible to change various characteristics of the TCP/UDP protocol and perform a number of tests that provide insight on the network's availability, delay, jitter and data loss [30].

It was also installed on the hosts/traffic generators and it serves to give the users an alternative to the MGEN, even though it was not our first choice it can be really useful to make different tests on the network.

4.4.3. Wireshark

Wireshark is a free and open source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues [35]. It is probably the most popular and effective free traffic analyser.

It is installed in all of the Raspberry Pis and it is used to verify if all messages and packets are being sent correctly.

Even though is very effective it is quite heavy for the Raspberry Pis and requires a graphical interface which further delays the remote access. Despite all this it is still the first choice and the main tool to use when something is not working as expected.

4.4.4. Tcpdump

Tcpdump is a common packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached [31].

It was used to make a quick analysis to the network and to confirm the characteristics of some packets, since it is less heavy than the Wireshark and runs on the command line it is more suitable to run remotely. For most instances tcpdump was enough and there was no need to use the Wireshark.

4.4.5. TC filter

The command line tool tc [41] is part of the iproute2 package and it allows us to configure the traffic flowing through our machine. It is one of the few tools available to manipulate low level traffic in Debian machines.

We can apply multiple filters to the incoming packets and then apply different actions to them according to our goals. The sequence of filters followed by actions, can produce many different algorithms and is using this tool that the scheduling algorithms will be made.

4.5. Virtual Local Area Networks (VLANs)

A virtual LAN (VLAN) is any broadcast domain that is partitioned and isolated in a computer network at the data link layer (OSI layer 2). LAN is an abbreviation of local area network. To subdivide a network into virtual LANs, one configures a network switch or router [32].

Since there were no optical devices, it was used VLANs to identify the different wavelengths of the network. The NG-PON2 it supposed to use up to 4 wavelengths at a time, so it was specified 4 different VLANs, one for each λ. The physical port and links between switches (OLT-ONU) are divided logicaly in four λs (VLANs).

The VLANs range from 11 to 14 and it is how the intermediary Raspberry Pi reorders and redirects the traffic to specific wavelengths, tagging the traffic with a VLAN tag representing its λ. Each wavelength (or VLAN) has a traffic limit and the ONUs can only be using one at a time, in each direction. Since they have only one receiver and transmitter, they can only tune to one wavelength for downstream and another for upstream in a certain point in time, the ONUs can later tune to other wavelengths but never more than one simultaneously. This is not the case for the OLT that can use all wavelengths simultaneously.

A simple scheme showing where the VLAN tags are added or removed as well as they existence within the network can be seen in Fig. 4.5.

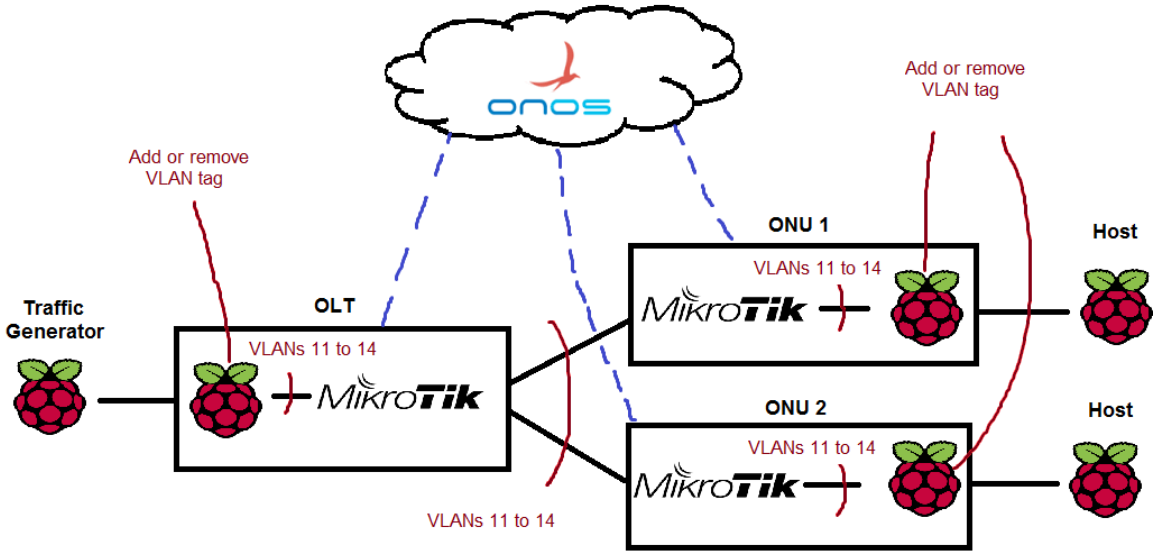


Figure 4.5: Representation of the VLANs within the network.

From an SDN controller's perspective, it will see a simple OpenFlow switch with multiple VLANs, that can easily be managed and configured.

4.6. Traffic Behaviour

To better understand how the traffic flows behave in this environment a brief description of the occurrences happening in downstream and upstream is going to be made.

4.6.1. Downstream

In the downstream direction the traffic is generated in the *traffic generator* then proceeds to enter the OLT and the intermediary Raspberry Pi.

In the Raspberry Pi ”inside” the OLT, it will pass through its tc filter and be assigned a VLAN tag according to its destination and the current occupancy of the VLANs, since they all have limited capacity, and also be reordered according to its priority. If a VLAN has reached its full capacity and there is no more space in the buffer the packet will be dropped or redirected to other VLAN.

Otherwise, it will go to the MikroTik on the OLT already with a specific VLAN tag and in the correct order according to priority. From here is pretty straightforward since the MikroTik is working with a FIFO scheduler and as CPqD switch, it will simply broadcast the traffic within the specified VLAN.

In the MikroTiks of the ONU it will check if the traffic destination is one of the ONU subscribers and if it isn't it will discard the packets. If it is, it will send the traffic to the following Raspberry Pi.

When it reaches the Raspberry Pi it will simply remove the VLAN tag and send the traffic to the destination host, finally reaching its destination.

4.6.2. Upstream

In the upstream direction it works similarly to the downstream except the Raspberry Pis of the ONUs switch roles with the one on the OLT, prioritizing and choosing the VLANs and its respective time slots. They need to know which VLANs are available and who else is transmitting in the same VLAN so it can avoid packet collisions.

4.7. Summary

In this chapter it was presented and analysed the topology deployed, a tree topology, as well as some of its characteristics and the reasons behind this choice.

It was shown which devices were used, which models and the changes or configurations needed to make in all of them in order to provide a good solution to our emulated environment. It was also explained how to reach every device remotely, enabling anyone to configure the network and its functionalities from a remote location.

After having the devices, it was necessary to make some changes to its software and, in the MikroTiks case, completely change the operating system. It was explained the necessary configurations made in the operating systems for the whole network to work as expected

and the tools required to install, giving a full definition of each tool and its purpose on the network.

Finally, it was explained how the wavelength channels were going to be managed and configured in a IP network, by recurring to the use of VLANs to identify the λs and it was given a brief description of how the traffic flows in this network in the downstream and upstream direction.

CHAPTER 5. ALGORITHM & RESULTS

In this chapter we are gonna explain the algorithm proposed as well as the one in which it was based. After that we will show some results and the conclusions that can be drawn from those.

The algorithm and its implementation main purpose is to demonstrate the tesbed capabilities. In future works the focus should be on the development of more complex and capcle algorithms as well as more intense testing.

5.1. Heterogeneous Earliest Finish Time Algorithm

The algorithm proposed is quite simple and based on the *Heterogeneous Earliest Finish Time (HEFT)* scheduling algorithm, a algorithm whose purpose is to schedule a set of dependent tasks onto a network of heterogeneous workers taking communication time into account [37].

It is divided in two phases, the first phase it is to order all the tasks according to their priority and dependent task attached to it and the second phase is to assign and schedule the prioritized tasks to the available workers, starting with the one with highest priority.

When it comes to performance the HEFT is well respected among heuristic algorithms for this problem. However it is a greedy algorithm and is not capable of making short-term sacrifices for long term benefits, so in more complex situations it can easily fail to find the optimal scheduling.

5.1.1. School lectures demonstration

Based on the description of HEFT scheduling algorithm it will be provided a simple example of its application based on the demonstration present in [38], where lectures are scheduled using the minimum number of classrooms possible.

But, in our case the lectures will be traffic packets and the classrooms will be ports of a switch, where each port has a different channel.

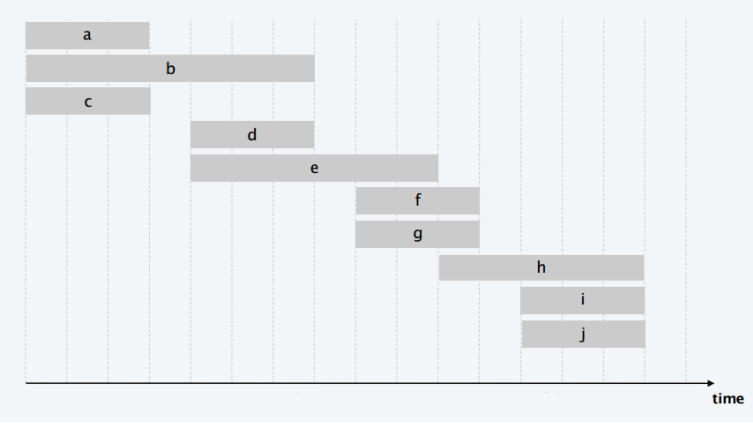


Figure 5.1: Traffic packets to be scheduled. [38]

The incoming traffic packets and their duration as well as the time they reach the switch can be seen in Fig. 5.1

The scheduling off all traffic packets is made following the order presented bellow:

- Packet *a* : there are no available active channels, open up a new channel (1) and assign packet *a* to it;
- Packet *b* : there are no available active channels, open up a new channel (2) and assign packet *b* to it;
- Packet *c* : there are no available active channels, open up a new channel (3) and assign packet *c* to it;
- Packet *d* : is compatible with channel 1 and 3, assign packet *d* to channel 3;
- Packet *e* : is compatible with channel 1, assign packet *e* to it;
- Packet *f* : is compatible with channel 2 and 3, assign packet *f* to channel 3;
- Packet *g* : is compatible with channel 2, assign packet *g* to it;
- Packet *h* : is compatible with channel 1, assign packet *h* to it;
- Packet *j* : is compatible with channel 2 and 3, assign packet *j* to channel 3;
- Packet *i* : is compatible with channel 2, assign packet *i* to it.

The final result is presented in Fig. 5.2.

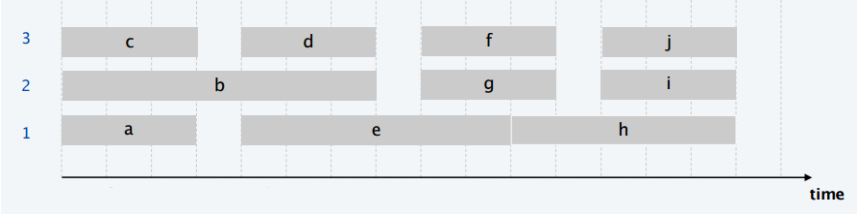


Figure 5.2: Traffic packets fully scheduled. [38]

5.2. Proposed algorithm

The created algorithm was based on the HEFT scheduling algorithm due to its simplicity, both of understanding and implementation, as well as the efficiency regarding the use of available resources.

In our algorithm, there are some differences in the type of resources available as well as the things needing scheduling. Instead of tasks we have traffic packets and emulating wavelengths (through VLANs) instead of workers. Besides the used terms the other adaptations was that there were no dependent tasks, or dependent traffic, and the time each packet took to be delivered was not taken into account.

The algorithm is implemented in the intermediary Raspberry Pi, emulating some parts of OLT/ONU TC layer, and is made using the *tc filter* tool contained in the *iproute2* package. It was made using the CBQ (Class Based Queueing) which is a classful qdisc that implements a rich linksharing hierarchy of classes. It contains shaping elements as well as prioritizing capabilities. Shaping is performed using link idle time calculations based on the timing of dequeue events and underlying link bandwidth [39].

When dequeuing to send traffic to the following network device, CBQ decides which of its classes will be allowed to send. It does so with a Weighted Round Robin (WRR) process in which each class with packets gets a chance to send in turn. The WRR process starts by asking the highest priority classes for packets, and will continue to do so until they have no more data to offer, in which case the process repeats for lower priorities [39].

In our work we did not have a strict hierarchy of classes because it was used multiple interfaces, each one requiring only one queueing disciple, class and filter attached to its interface, but nevertheless, the incoming traffic followed a hierarchical path.

One of the most important features to the success of this algorithm was the policing action in tc, it allows us to police the traffic rate going through an interface and perform an action if a specific rate, defined by us, is passed. This was the main reason why CBQ was chosen since it is the only queuing discipline that allows a policing policy to be attached to a class, all other queuing disciplines treat a filters attempt to police the packet as though the filter failed to classify the packet.

We created one queuing disciplines and one classes in each used interface in order to limit the traffic rate and establish the capacity of our link between the intermediary Raspberry Pi and the MikroTik as well as the capacity of the channels that this link aggregates.

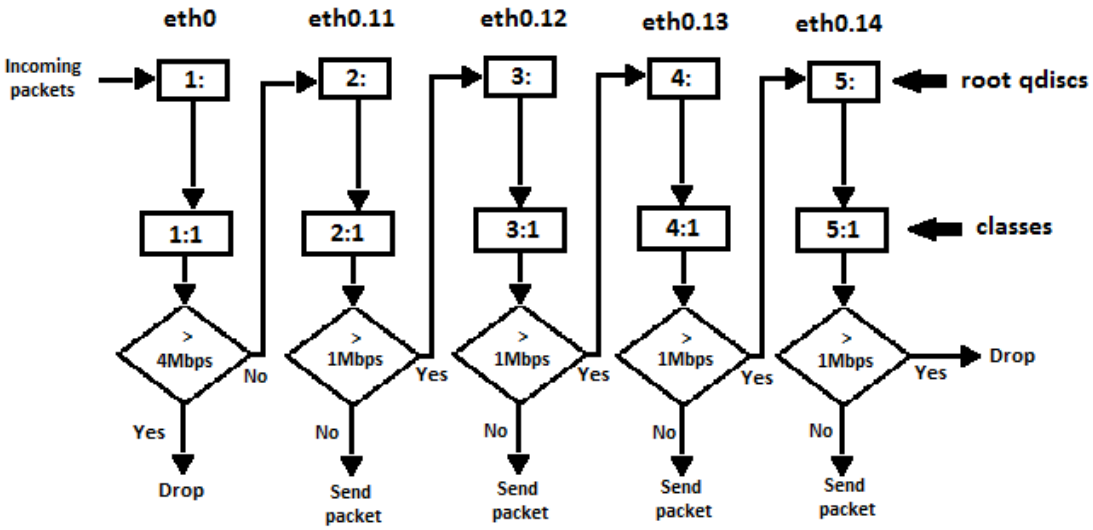


Figure 5.3: Scheme of the algorithm as seen by the CBQ.

Due to the limitations of the hardware and the available traffic speeds allowed by the traffic generator the capacity could not be very high and was set at 4 Mbit/s of aggregated capacity where each channel has 1 Mbit/s, the reason why this values were chosen is because we can easily scale the values by 10 000 and obtain the real aggregated capacity of the NG-PON2 system, of 40 Gb/s, and the single channel capacity of 10 Gb/s.

Then it was added a filter to each class creating the aforementioned traffic hierarchical path. The first filter, belonging to eth0, redirects all incoming packets to eth0.11, then, using the traffic police policy we set a filter in the eth0.11 interface that redirects incoming packets to eth0.12 if a traffic rate of 1 Mbit/s is exceeded. The same is done in eth0.12 and eth0.13 redirecting packets to eth0.13 and eth0.14, respectively. In the last interface, eth0.14, if the 1Mbit/s rate was exceeded instead of redirecting to another interface the packet would be dropped. A brief scheme illustrating how the CBQ sees the algorithm is shown in Fig. 5.3.

For a packet to be dropped in eth0.14 would mean that all the previous channels, including the eth0.14, are completely occupied and the traffic is above 4 Mbit/s which should never happen since the traffic is limited in the first queueing discipline, in eth0, to 4 Mbit/s.

The traffic packets that are redirected to the interfaces eth0.11, eth0.12, eth0.13 and eth0.14 are tagged with the VLAN 11, 12, 13 and 14, respectively. The VLANs in turn serve to identify the wavelength to be used in the NG-PON2 system.

The complete script to create the algorithm is shown in Appendix C. A small illustration which gives us a better idea of where the algorithm is running and its place on the network is present in Fig. 5.4.

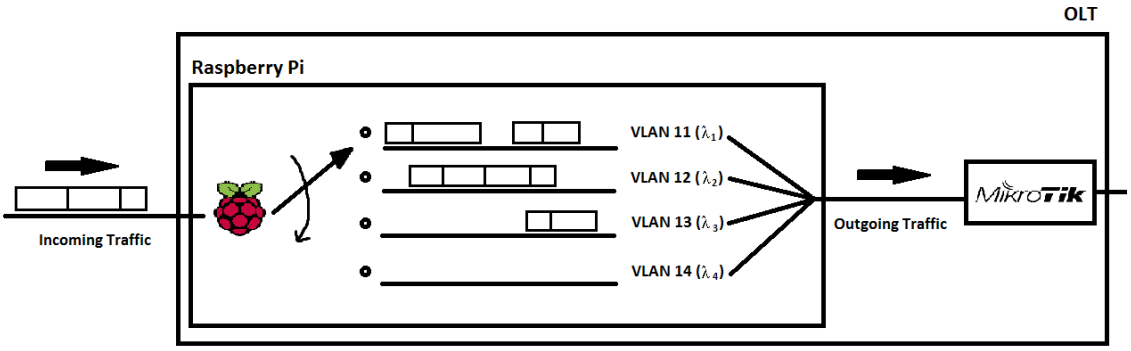


Figure 5.4: Simplified illustration of the algorithm and its place in the network.

5.3. Tests

To test our algorithm, it was needed to create a set of tests that would tryout different traffic speeds and with different traffic distributions.

It was tested four different traffic rates: 0.8 Mbps, 2.4 Mbps, 4 Mbps and 5.6 Mbps. And two different types of traffic distribution for each speed, a periodic distribution which presents constant traffic speed and a Poisson distribution which is more irregular being some times above and others below the assigned traffic rate. Leading to a total of eight different traffic tests.

The 0.8 Mbps rate has the purpose of testing if only the first channel is used (VLAN 11) while the other remain turned off. The second slowest rate, 2.4 Mbps, has the intention of seeing how the traffic load is distributed among the channels, ideally there should be no packets sent by the last channel (VLAN 14). The 4 Mbps rate is to see how the algorithm behaves when in its limit capacity and the final traffic rate, 5.6 Mbps, is to see how the

traffic is handled when the incoming rate is well above the capacity of the system.

All the tests had the duration of one minute, leading to a total traffic of 48, 144, 240 and 336 Mbits for the rates 0.8, 2.4, 4 and 5.6 Mbps, respectively.

Between the tests the Raspberry Pi containing the algorithm was always rebooted so it would clean previous filters and not be influenced by older results and tests. In order to see the statistics of the filters, classes and queueing disciplines it was used the script present in Appendix C.

5.4. Results

The results of the previously mentioned tests will be displayed in a tables and in a bars chart to facilitate analysing and interpret the results. It should be noted that each packet has the size of 500 bytes and maximum number of packets that one of our interfaces can send in one minute is 15 000 packets.

5.4.1. Periodic traffic

Periodic traffic should create a constant flow of traffic always with the same rate, sending a constant number of packets with the length of 500 bytes. The number of packets sent per second vary according to the desired rate, to see specific rates and the script for the generated traffic check Appendix C.

This type of traffic is quite useful to check if the algorithm is working as expected. However, it is not a very realistic traffic distribution for todays traffic patterns.

For the tests with the periodic traffic distribution the results were as shown in Table 5.1.

Table 5.1: Number of packets transmitted by each interface in the periodic distribution.

Rate [Mbps]	eth0.11 [pkts]	eth0.12 [pkts]	eth0.13 [pkts]	eth0.14 [pkts]
0.8	12 000	2	0	0
2.4	13 558	13 233	8 573	141
4	14 872	14 532	14 037	13 577
5.6	14 853	14 792	14 533	13 968

Analysing the results of Table 5.1 we see that for the 0.8 Mbps rate only the first channel is used which is what is expected and a signal that our algorithm is working well with low traffic rates.

For the 2.4 Mbps rate the first and second channels are almost fully occupied, therefore the third channel is also transmitting which is how the algorithm should work, however, there are some packets going to channel four that shouldn't have passed through the third channel. Probably is due to a burst in the traffic generated that lead to it.

As for the two remaining rates all channels are almost completely occupied which is expected since the traffic rate being sent is equal or higher than the link aggregated capacity.

Using the values from Table 5.1 we can calculate the traffic generated, transmitted and dropped, which results in Table 5.2.

Table 5.2: Traffic generated, transmitted and dropped in the periodic distribution.

Rate [Mbps]	Traffic generated [Mbit]	Traffic transmitted [Mbit]	Traffic dropped [Mbit]
0.8	48	48	0
2.4	144	144	0
4	240	228	12
5.6	336	233	103

From Table 5.2 we can clearly see that for the first two rates all the generated traffic was transmitted with success and in the last rate only 69% of traffic was transmitted which is expected having into account that the rate at which the traffic was generated is well above the link capacity.

In the third rate 95% was transmitted, which is good, but could still be higher since the traffic rate is the same as the link capacity, the reason why it is not higher is that the filters that monitor the traffic are configured to start redirecting traffic when the channel capacity reaches 990 kbps, so in reality the channel capacity is slightly lower than 1 Mbps. This little difference in capacity, plus the small traffic rate variations, even in periodic traffic, is enough to justify why all the traffic generated was not transmitted.

From the data presented in the previous tables we then made the bar chart represented in Fig. 5.5, where the average usage of each channel, for the minute in which the test took place, is shown.

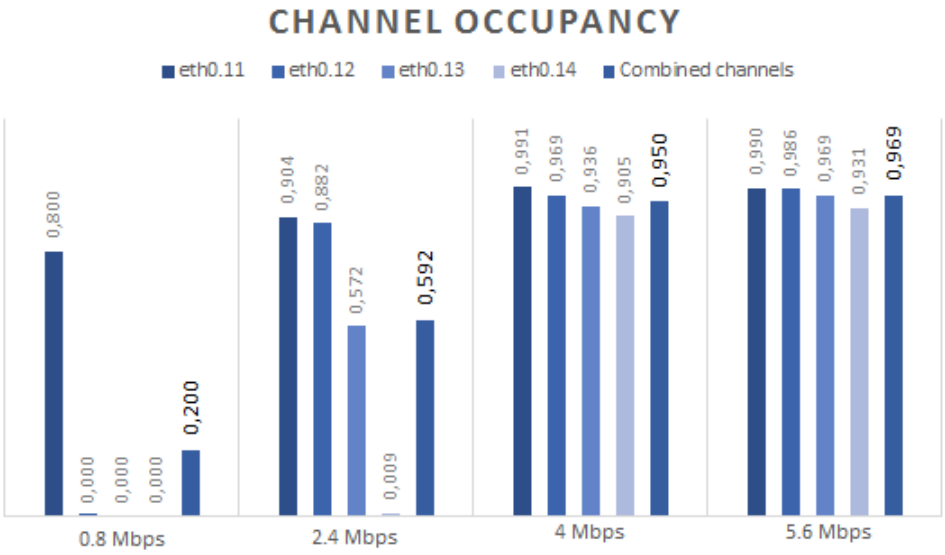


Figure 5.5: Channel occupancy average in the periodic distribution.

In Fig. 5.5 we can see that the algorithm seems to be working as desired when confronted with periodic traffic, it fills the lower channels first before going to the higher ones, e.g. it will only transmit in the interface eth0.13 if eth0.11 and eth0.12 are full (case of rate 2.4 Mbps).

It is also worth noticing that in cases of higher traffic rates the load distribution between the four channels seems to be well balanced.

5.4.2. Poisson traffic

The Poisson traffic distribution has as parameters the average rate, of packets per second, and the size of each packet. In order to compare the results with the ones obtained with the periodic traffic it was used as average rate the same rate as in the periodic traffic and the same size of packets, 500 bytes. To see specific rates for the traffic generated check Appendix C.

The traffic with Poisson distribution even though having the same average rate as the periodic traffic it has more burstiness, having periods of time when transmits at a superior or an inferior rate than its periodic counterpart. It is a better approximation than the periodic distribution af the real life traffic, but still is quite predictable, which is not always the case of real life traffic rates.

For the tests with the Poisson traffic distribution the results were as shown in Table 5.3.

Table 5.3: Number of packets transmitted by each interface in the Poisson distribution.

Rate [Mbps]	eth0.11 [pkts]	eth0.12 [pkts]	eth0.13 [pkts]	eth0.14 [pkts]
0.8	9 626	2 063	106	2
2.4	13 361	11 708	7 163	2 948
4	13 828	13 435	12 988	11 887
5.6	14 054	13 911	13 810	13 753

From the Table 5.3 we can already see some minor differences where the traffic is more spread through the channels, due to the periods of time in which its transmitting above the average rate, but overall it has the same tendency as the periodic traffic.

For the higher rates the channels do not transmit as many packets as in the periodic distribution leading to a lower traffic transmitted even though the average rate of traffic generated is the same, this can be explained by the same reason why the traffic with lower rates is more spread throughout the channels, there are periods of time where the traffic generated is above the system limit of 4 Mbps making so that the exceeding packets are dropped.

Table 5.4: Traffic generated, transmitted and dropped in the Poisson distribution.

Rate [Mbps]	Traffic generated [Mbit]	Traffic transmitted [Mbit]	Traffic dropped [Mbit]
0.8	48	48	0
2.4	144	144	0
4	240	209	31
5.6	336	222	114

Using the values from Table 5.3 we can calculate the traffic generated, transmitted and dropped, which results in Table 5.4. It is important to mention that since this is the product

of a Poisson distribution the generated traffic is an approximation.

From Table 5.4 we can clearly see that for the first two rates all the generated traffic was transmitted with success like in the periodic distribution since the traffic rates are not high enough to make a difference when it comes to loss of packets.

The third rate has 87% of the traffic generated transmitted and the last rate only 66%, these values are lower than the periodic distribution equivalents because for these higher rates the channels transmitted overall less packets. The reason behind this was previously explained and is mainly due to the moments in which the traffic rate is above the link capacity, which with a Poisson distribution happens more often since it has more burstiness.

From the data presented in the previous tables we then made the bar chart represented in Fig. 5.6, where the average usage of each channel, for the minute in which the test took place, is shown.

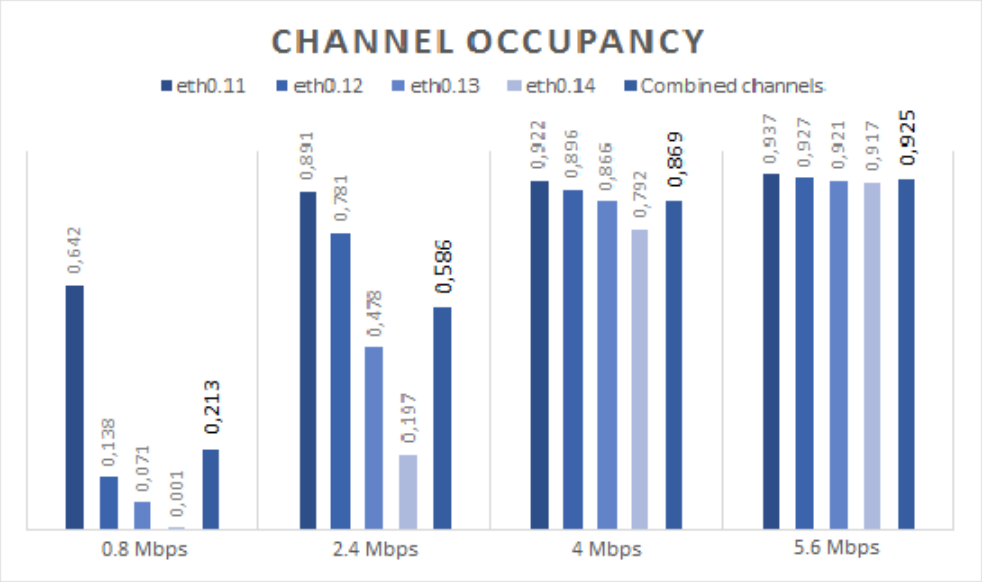


Figure 5.6: Channel occupancy average in the Poisson distribution.

In Fig. 5.6 we can see that the algorithm is still working, but since the traffic rate tends to vary over time there are periods where it ends up transmitting in three channels due to its high rate and periods where not even one channel is being used to its full capacity. This uncertainty leads to a higher spread of traffic throughout the different interfaces, as already pointed out previously.

The total channel occupancy average is different for one distribution to the other because of the uncertainty regarding the actual traffic generated in the Poisson distribution, which can be higher or lower than the average traffic rate defined for the distribution this changing one of the main parameters to calculate the channel occupancy average.

To conclude, the algorithm behaviour to both periodic and Poisson distribution is quite satisfying. Regarding the traffic load distribution through the channels it presents significant differences with lower traffic rates, but, when the rates start getting close to the system capacity the behaviour and results are quite similar. All the results obtained were within the expected values, thus leading us to believe that the algorithm is well implemented and working as desired.

The fact that at times not all channels are needed, when using the proposed DBA scheduling algorithm, means that we can turn those channels off until they are required again leading to energy savings throughout the network.

Also, is worth remembering that the output of the channels mentioned in the proposed algorithm is in fact the Raspberry Pi's port connected to the MikroTik, which in turn belongs to the ONOS Optical (PHY) Layer.

5.5. Further improvements

Even though it worked as desired it is still a quite simple algorithm with a lot of room for improvement, be it in more complex and effective algorithms or simply better ways of implementing it.

Using more filters it is possible to prioritize the traffic according to the protocol used, destination, TOS, etc. Allowing to create quite complex algorithms which can be adaptable to multiple scenarios and optimizing even better the use of resources and the saving of energy in the NG-PON2 networks.

In the test there could have existed even more different types of traffic distribution to emulate the real life traffic as much as possible.

Also to have a better look at the response of the system, it would be a great addition if it was possible to monitor the traffic sent by each of the interfaces belonging to the VLANs in real time.

Overall, the proposed algorithm and tests still have much to improve but, it is the first step into emulating an optimal and effective DBA scheduling algorithm for the NG-PON2 networks.

5.6. Summary

In this chapter it was presented and explained the HEFT scheduling algorithm, the algorithm that we adapted to evaluate the NG-PON2 behaviour and implemented using the tc tool and its features.

It was made clear the reason behind the choice of a CBQ, which is a quite difficult classful queueing disciple to work with, how the algorithm was implemented within the CBQ rules and structure and a throughout description on how the algorithm behaves upon receiving an incoming traffic packet.

Then it was explained the battery of test created to test the algorithm, with different types of traffic distribution and different traffic rates, leading to a total of eight different tests.

It is shown through the use of tables and charts the results to the multiple tests created, followed by an analysis and explanation of said results.

Finally, it was thought of possible ways to improve the created algorithm or develop a new one.

CHAPTER 6. CONCLUSIONS AND FUTURE WORKS

To finalize this work we elaborated a conclusion to the work done in this thesis and what should be done next.

6.1. Conclusions

At the beginning of this work we set ourselves to create an algorithm that would take full advantage of the NG-PON2 system, the added new wavelengths, the tunability of the ONUs and the possibility of having a point-to-point wavelength connection between two points of different optical access networks. The PtP WDM possibility enables a new set of services, since it can guarantee QoS between the two connected points a very appealing feature, especially for services highly sensitive to QoS parameters (e.g. 5G). The SIEPON and SDN features are key to this since they can maintain the QoS parameters in network, regardeless of the different vendors' devices and of other services.

We then realized that it was necessary to create an environment where our algorithm could be created and tested, and thus appeared one of the main goals of this work, the creation of a fully emulated SIEPON-SDN-NG-PON2 network with all the tools required to create and test DBA scheduling algorithms. In order to make this possible, we created a physical topology using Raspberry Pis and MikroTiks and after many configurations and changes at the software level it was created a fully SIEPON-SDN-NG-PON2 emulated network. This network was made with the intention of being open and shared within colleagues on the project, providing a platform to create and test DBA scheduling algorithms for the NG-PON2 technology.

Once the environment was deployed it was needed a test to analyse if it emulated network really worked as expected and, in alignment with the original goal of the project, it was decided to create a simple algorithm to manage and control efficiently the traffic going through the available channels of our environment. The algorithm makes so that a new channel is only used if the remaining higher priority channels are full, in case there is no available channels the traffic packet is dropped. The algorithm was successfully created and applied to the network, therefore fulfilling the first function of the environment, the ability to develop and apply DBA scheduling algorithms.

We then created a battery of tests to be applied to the developed algorithm and see how it would behave under different traffic loads. The algorithm passed all the tests successfully exhibiting the expected behaviour in all of them. With this we fulfilled the second function of the deployed environment, the ability to test created DBA scheduling algorithms, as well as having produced a simple but effective scheduling algorithm.

The tests showed that the traffic was succesfully redirected to the channel with highest priority available and once that channel was fully accupied it went to the next channel of highest priority without dropping packets. The algortihm behaves better when there is regular traffic, in the Poisson traffic distribution tests there occured more traffic spikes leading to a wider use of the available channels. once more is worth mentioning that the tests were just preliminary and to demonstrate the power of the testbed, to be developed in future works.

To conclude, in this thesis we demonstrated the technical feasibility of a fully SIEPON-SDN-NG-PON2 emulated environment, bringing together the SIEPON and SDN concepts plus the NG-PON2 technology, which will undoubtedly impact the future of access networks.

6.2. Future works

The testbed should be improved in order to better emulate the real life traffic patterns and behaviour, leading to more realistic and significant results. The Raspberry Pis and the MikroTiks could be replaced by more costly software-based routers, like the [45], increasing the CPU performance and the traffic limit of the emulated environment. The traffic generator should be improved to generate multiple types of traffic and traffic distributions simultaneously and at much higher rates. Finally, the tools used to gather the results should also be improved or changed in order to provide a better view of what is happening in the emulated environment during the tests and when they finish.

Since the environment was developed with the purpose of allowing the creation and testing of DBA scheduling algorithms, it is expected for this platform to be used to create and test other algorithms in the pursuit of the best DBA scheduling algorithm for NG-PON2. Scheduling algorithms like the GPS [46] or the WFQ [47] should be tried out since it would add proportional sharing features, despite the fact that it would be more difficult to implement due to their complexity and would require a different data model from the one used in our testbed.

Power consumption of xPON networks could be improved by turning OLTs or ONUs into sleep mode as described in [42], switching off some wavelength channels or even devices, according to traffic usage, which would introduce a new feature that would need to be added to the testbed in order to be evaluated.

6.3. Environmental impact

The proposed algorithm optimizes the use of wavelength channels to the minimum required, allowing for the unused wavelength channels to be turned off until they are required. When applied to the entire optical network, it will lower the overall power consumption therefore lowering the environmental impact.

Since the presented work lays the foundation for the design of power-aware scheduling algorithms, similar to those described in [43] or [44], it is possible to have an even further positive impact on the environment.

Additionally, the adaptation of SDN to the optical access networks brings more possibilities for a greater environmental impact due to having a wider view of the network devices, it can better optimize the use of resources like adjusting OLTs usage, switching between active and sleep mode [42], according to the daily traffic patterns.

BIBLIOGRAPHY

[1] R.G. Clegg, J. Spencer, and R. Landa, "Pushing Software Defined Networking to the access", at the Third European Workshop on Software-Defined Networks, 2014. ix, 3, 17, 25

[2] Derek Nesset,"NG-PON2 Technology and Standards", *Journal of Lightwave Technology*, vol. 33, no. 5, March 2015. ix, 1, 5, 7, 10

[3] J.-I. Kani, F. Bourgart, A. Cui, A. Rafel, M. Campbell, R. Davey, and S. Rodrigues, "Next-generation PON part I—technology roadmap and general requirements", *IEEE Commun. Mag.* , vol. 47, no. 11, pp. 43–49, November 2009. 5

[4] "40-Gigabit-capable passive optical networks (NG-PON2): General requirements",ITU-T G.989.1 Recommendation, March 2013. ix, 5, 6

[5] "40-Gigabit-capable passive optical networks 2 (NG-PON2): Physical media dependent (PMD) layer specification",ITU-T G.989.2 Recommendation, December 2014. ix, 8

[6] D. van Veen, W. Poehlmann, B. Farah, T. Pfeiffer, and P. Vetter, "Measurement and mitigation of wavelength drift due to self-heating of tunable burst-mode DML for TWDM-PON", presented at the Opt. Fiber Commun., San Francisco, CA, USA, 2014, Paper W1D.6. 6

[7] R. Murano, "Optical component technology options for NGPON2 systems", presented at the Opt. Fiber Commun. Conf. Expo., San Francisco, CA, USA, 2014, Paper M3I.1. 7

[8] H. Saito, H. Iwamura, N. Minato, S. Kobayashi, M. Sarashina, H. Tamai, K. Sasaki, and M. Kashima, "Field trial and simulation of bandwidth allocation for efficient OLT operation on virtualized PON", presented at the Eur. Conf. Exhib. Opt. Commun., Amsterdam, The Netherlands, 2013, Paper Tu.4.B.5. 9, 10

[9] Gigabit-Capable Passive Optical Networks (GPON): Long Reach (40km Differential Distance), ITU-T G.984.7 Recommendation, July 2010. 9

[10] W. Pohlmann and T. Pfeiffer, "Demonstration of wavelength-set division multiplexing demonstration of wavelength-set division multiplexing for a cost effective PON with up to 80 Gbit/s upstream bandwidth", presented at the Eur. Conf. Exhib. Opt. Commun., Geneva, Switzerland, 2011, Paper We.9.C.1. 10

[11] R. Murano and M. J. L Cahill, "Low cost tunable receivers for wavelength agile PONs", presented at the Eur. Conf. Exhib. Opt. Commun., Amsterdam, The Netherlands, 2012, Paper We.2.B.3. 10

[12] S. Kaneko, T. Yoshida, S. Furusawa, M. Sarashina, H. Tamai, A. Suzuki, T. Mukojima, S. Kimura, and N. Yoshimoto, "First λ-tunable dynamic load-balancing operation enhanced by 3-msec bidirectional hitless tuning on symmetric 40-Gbit/s WDM/TDM-PON", presented at the Opt. Fiber Commun. Conf. Exhib., San Francisco, CA, USA, 2014, Paper Th5A.4. 10

[13] Y. Luo, H. Roberts, K. Grobe, M. Valvo, D. Nesset, K. Asaka, H. Rohde, J. Smith, J.S. Wey, and F. Effenberger, "Physical Layer Aspects of NG-PON2 Standards-Part 2: System Design and Technology Feasibility [Invited]", , *Opt. Commun. Netw.*, vol. 8, no. 1, January 2016. 8

[14] N. Feamster, J. Rexford, E. Zegura, "The road to SDN: an intellectual history of programmable networks", *ACM Queue*, vol. 44, no.2, April 2014. 15

[15] Open Network Foundation (ONF), http://opennetworking.org. ix, 15, 18

[16] David Quiroz, "Master Thesis: Research on path establishment methods performance in SDN-based networks", EETAC, Universitat Politecnica de Catalunya, November 2015, https://upcommons.upc.edu/handle/2117/80972. xi, 16, 17, 18

[17] H. Khalili, D. Rincon, S. Sallent, "Towards an Integrated SDN-NFV Architecture for EPON Networks", Dept. of Telematics Engineering, *LNCS*, vol. 8846, pp. 74–84, 2014. ix, 24, 25

[18] K. Rutka, "The OpenFlow Soft Switch", at Erlang User Conference in Stockholm, June 2012. ix, 19

[19] N. Cvijetic, A. Tanaka, P.N. Ji, K. Sethuraman, S. Murakami, T. Wang, "SDN and Openflow for dynamic flex-grid optical access and aggregation networks", *IEEE J. Lightwave Technol.* , vol. 32, no. 4, 2014. 25

[20] P. Parol, M. Pawlowski, "Towards networks of the future: SDN paradigm introduction to PON networking for business applications", in Proceeding of the Federated Conferennce on Computer Science and Information Systems (FEDCSIS), 2013. 24

[21] A. Sgambelluri, F. Paolucci, F. Cugini, L. Valcarenghi, P. Castoldi, "Generalized SDN control for access/metro/core integration in the framework of the interface to the Routing System (I2RS)", in IEEE Globecom Workshops, 2013. 25

[22] "10-Gigabit-capable passive optical networks (XG-PON): Transmission convergence (TC) layer specification", ITU-T G.987.3 recommendation, January 2014. 11

[23] "40-Gigabit-capable passive optical networks (NG-PON2): Transmission convergence layer specification", ITU-T G.989.3 recommendation, October 2015. ix, xi, 8, 11, 12

[24] ON.Lab, CORD Project, http://opencord.org. 26

[25] Central Office Re-architected as Datacenter (CORD) white paper, AT&T and Open Networking Lab, June 2015. ix, 26

[26] ON.Lab, ONOS Project, http://onosproject.org. 19

[27] J. Salgado, R. Zhao, N. Monteiro, "New FTTH-based Technologies and Applications", Fibre to the Home Council Europe, 2014. ix, 13

[28] Greenfield definition: https://en.wikipedia.org/wiki/Greenfield_project. 9

[29] OpenWRT definition: https://en.wikipedia.org/wiki/OpenWrt. 31

[30] Iperf definition: https://en.wikipedia.org/wiki/Iperf. 33

[31] Tcpdump definition: https://en.wikipedia.org/wiki/Tcpdump. 33

[32] VLAN definition: https://en.wikipedia.org/wiki/Virtual_LAN. 34

[33] Raspberry Pi Foundation, https://www.raspberrypi.org/. 32

[34] U.S. Naval Research Laboratory, Networks and Communication Systems Branch, http://www.nrl.navy.mil/itd/ncs/products/mgen. 33

[35] Wireshark definition: https://en.wikipedia.org/wiki/Wireshark. 33

[36] H. Khalili, D. Rincon, S. Sallent, J.R. Piney, "SDN architecture in a SIEPON", Dept. of Network Engineering, Universitat Politècnica de Catalunya (UPC), 2016. 13

[37] Heterogeneous Earliest Finish Time algorithm definition: https://en.wikipedia.org/wiki/Heterogeneous_Earliest_Finish_Time. 37

[38] J. Kleinberg, E. Tardos, "Algorithm Design", Chapter 4, available in http://www.cs.princeton.edu/∼wayne/kleinberg-tardos/. ix, 37, 38

[39] CBQ manual: http://linux.die.net/man/8/tc-cbq. 39

[40] Aantonio Mesones Ruiz, "Master Thesis: Virtualization of xPON with SDN", Universitat Politecnica de Catalunya, October 2016, publication pending. 31

[41] tc manual: http://man7.org/linux/man-pages/man8/tc.8.html. 33

[42] M. Hajduczenia, M. Takizawa, "Power-Saving Mechanisms in IEEE Std 1904.1 (SIEPON)", available in http://www.ieee802.org/3/bn/public/adhoc_eval/adhoc_eval_hajduczenia_02_0313.pdf 48

[43] M. R. Palattella, N. Accettura, M. Doher, "Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15.4e networks", *Personal Indoor and Mobile Radio Communications (PIMRC)* , 2012 IEEE 23rd International Symposium. 48

[44] N. Quang Hung, N. Thoai, N. Thanh Son, "Performance Constraint and Power-Aware Allocation for User Request in Virtual Computing", Ho Chi Minh City University of Technology, August 2011, available in https://arxiv.org/ftp/arxiv/papers/1210/1210.1026.pdf. 48

[45] Brocade vRouter, data sheet available in: http://www.brocade.com/en/backend-content/pdf-page.html?/content/dam/common/documents/content-types/datasheet/brocade-vrouter-ds.pdf. 48

[46] Generalized processor sharing, defined in: https://en.wikipedia.org/wiki/Generalized_processor_sharing. 48

[47] Weighted fair queueing, defined in: https://en.wikipedia.org/wiki/Weighted_fair_queueing. 48

ACRONYMS

API	Application Programming Interface
ATM	Asynchronous Transfer Mode
AMCC	Auxiliary Management and Control Channel
B2B	Business to Business
BNG	Broadband Network Gateways
CPqD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
CAPEX	Capital Expenditure
CDPI	Control to Data-Plane Interface
CORD	Central Office Re-architected as Datacenter
CPE	Customer Premises Equipment
CBQ	Class Based Queueing
DBA	Dynamic Bandwidth Allocation
DS	Downstream
DWDM	Dense Wavelength Division Multiplexing
DFB	Distributed Feedback
DBR	Distributed Bragg Reflector
DPID	Datapath ID
DB	Database
DHCP	Dynamic Host Configuration Protocol
EPON	Ethernet Passive Optical Network
ETH	Ethernet
FSAN	Full Service Access Network
FTTH	Fiber to the Home
FIFO	First In First Out
GPON	Gigabit Passive Optical Network
GEPON	Gigabit Ethernet Passive Optical Network
GEM	GPON Encapsulation Method
GUI	Graphical User Interface
GPS	Generalized Processor Sharing
HAL	Hardware Abstraction Layer
HEFT	Heterogeneous Earliest Finish Time
ITU-T	International Telecomunications Union - Telecomunication Standardization Sector
IEEE	Institute of Electrical and Electronics Engineers
ICPT	Inter Channel Termination Protocol
IP	Internet Protocol
IO	Input/Output
LSA	Link State Advertisement

MAC	Media Access Control
MGEN	Multi-Generator
MNGT	Management
MPLS	Multiprotocol Label Switching
MPCP	Multi-Point Control Protocol
NG-PON2	Next Generation - Passive Optical Networks 2
NBI	Northbound Interface
NFV	Network Function Virtualization
ONOS	Open Network Operating System
OS	Operating System
OLT	Optical Line Termination
ONU	Optical Network Unit
ODN	Optical Distribution Network
OMCI	ONU Management and Control Interface
OPEX	Operating Expediture
OPP	Optical Path Penalty
OAM	Operation, Administration and Maintenance
OMCC	ONT Management and Control Channel
ONT	Optical Network Terminal
OFC	OpenFlow Controller
OF	OpenFlow
OVS	Open vSwitch
ON.Lab	Open Networking Lab
OSI	Open Systems Interconnection
PON	Passive Optical Network
PMD	Physical Media Dependent
PtP WDM	Point-to-Point Wavelength Division Multiplexing
PLOAM	Physical Layer Operation, Administration and Maintenance
PM	Performance Monitoring
PHY	Physical
PBB	Provider Backbone Bridging
PCE	Path Computation Element
PC	Personal Computer
QoS	Quality of Service
RF	Radio Frequency
SDN	Software Defined Networking
SIEPON	Service Interoperability in EPON
SDU	Service Data Unit
SBI	Southbound Interface
SLA	Service Level Agreement
TC	Transmission Convergence
TWDM	Time and Wavelength Division Multiplexing
TDM	Time Division Multiplexing
TCP	Transmission Control Protocol
TLS	Transport Layer Security
US	Upstream
UDP	User Datagram Protocol
WRR	Weighted Round Robin
WFQ	Weighted Fair Queueing

APPENDICES

APPENDIX A. MIKROTIK CONFIGURATION

To make the MikroTiks work as desired some configurations were necessary to perform. Firstly, it was flashed of its default software and installed a version of OpenWrt containing the CPqD switch module. But this was not enough and some specific configurations to the network and openflow file were made.

It was necessary to add the VLANs 11 to 14 to the ports connected with the Raspberry Pis, and connect to the controller correctly without disturbances which proved more difficult than expected.

To solve the last issue the architecture of the MikroTik-OpenWrt was looked into, see Fig. A.1, and we discovered that the port 1 has a different network than the other ports so it was decided that port 1 will always be used to connect to the SDN controller.

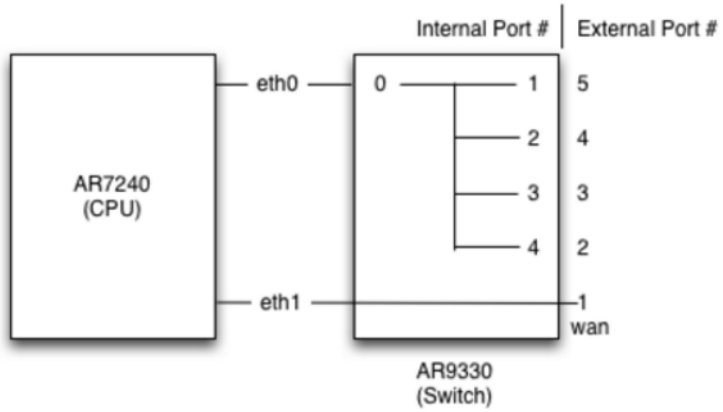


Figure A.1: MikroTik-OpenWrt architecture.

For the configuration to work it was necessary to disable the firewall, so the files located in *etc/init.d/firewall* and *etc/config/firewall* were removed.

Finally the *etc/config/network* file became:

```
config interface 'loopback'
option ifname 'lo'
option proto 'static'
option ipaddr '127.0.0.1'
option netmask '255.0.0.0'

config interface 'wan'
option ifname 'eth0.1'
option proto 'static'
option ipaddr '192.168.88.174'  #use prefered ip
option netmask '255.255.255.0'
option netmask '192.168.88.1'

config interface 'lan1'
option ifname 'eth0.2'
```

```
option proto 'static'

config interface 'lan2'
option ifname 'eth0.3'
option proto 'static'

config interface 'lan3'
option ifname 'eth0.4'
option proto 'static'

config interface 'lan4'
option ifname 'eth0.5'
option proto 'static'

config switch
option name 'switch0'
option reset '1'
option enable_vlan '1'

config switch_vlan
option device 'switch0'
option vlan '1'
option ports '0t 1'

config switch_vlan
option device 'switch0'
option vlan '2'
option ports '0t 2'

config switch_vlan
option device 'switch0'
option vlan '3'
option ports '0t 3'

config switch_vlan
option device 'switch0'
option vlan '4'
option ports '0t 4'

config switch_vlan
option device 'switch0'
option vlan '5'
option ports '0t 5'

config switch_vlan
option device 'switch0'
option vlan '11'
option ports '0t 2 3'
```

```
config switch_vlan
option device 'switch0'
option vlan '12'
option ports '0t 2 3'


config switch_vlan
option device 'switch0'
option vlan '13'
option ports '0t 2 3'

config switch_vlan
option device 'switch0'
option vlan '14'
option ports '0t 2 3'
```

Even though there also VLANs from 1 to 5, they are necessary for the switch to work as a CPqD switch but are not related to our management of wavelengths through VLANs.

And the *etc/config/openflow* file became:

```
config 'ofswitch'
option 'dp' 'dp0'
option 'dpid' '00000000008x' #change x for last number of ip
option 'ofports' 'eth0.2 eth0.3 eth0.5'
option 'ofctl' 'tcp:192.168.88.209:6633'
option 'mode' 'outofband'
```

It is also worth mentioning that the port 4 of all the MikrotTiks present on the laboratory do not work, the problem is not new and nobody knows how to fix it. Therefore, we are left with 4 working ethernet ports.

In the cases of the MikroTiks that represent the ONUs this is not a issue but for the one that represents the OLT it should have more available. This is the reason why this work only was two ONUs. It was adquired a 10-port MikroTik to solve this but not in time to make necessary adaptations and changes.

APPENDIX B. RASPBERRY PI CONFIGURATION

In the Intermediary Raspberry Pis it was necessary to configure them as bridges and add the VLANs, on the other hand, in the remaining Raspberry Pis there was no need of any specific configurations, just installing the necessary software and removing the DHCP, as explained previously.

The file changed was *etc/network/interfaces* where it was added:

```
auto lo br0
iface lo inet loopback

iface eth0 inet manual
iface eth1 inet manual

iface br0 inet static
        address 192.168.89.71
        netmask 255.255.255.0
        bridge_ports eth0 eth1
```

This way the intermediary Raspberry Pis work as bridges between the MikroTiks and the remaining Raspberry Pis, thus, giving us the ability to reach all hosts and exchange traffic among themselves.

APPENDIX C. SCRIPTS

Script countaining the commands for the tc tool to build the algorithm is here presented:

```
#!/bin/bash

# first add the multiple queuing disciples

tc qdisc add dev eth0 ingress

tc qdisc add dev eth0 root handle 1: cbq bandwidth 4Mbit allot 1514 cell 8
avpkt 1000 mpu 64

tc qdisc add dev eth0.11 root handle 2: cbq bandwidth 1Mbit allot 1514 cell 8
avpkt 1000 mpu 64

tc qdisc add dev eth0.12 root handle 3: cbq bandwidth 1Mbit allot 1514 cell 8
avpkt 1000 mpu 64

tc qdisc add dev eth0.13 root handle 4: cbq bandwidth 1Mbit allot 1514 cell 8
avpkt 1000 mpu 64

tc qdisc add dev eth0.14 root handle 5: cbq bandwidth 1Mbit allot 1514 cell 8
avpkt 1000 mpu 64



# then create a class for each aforementioned qdisc

tc class add dev eth0 parent 1:0 classid 1:1 cbq bandwidth 4Mbit rate 4Mbit
allot 1514 cell 8 weight 400kbit prio 8 maxburst 20 avpkt 1000

tc class add dev eth0.11 parent 2:0 classid 2:1 cbq bandwidth 1Mbit rate 1Mbit
allot 1514 cell 8 weight 100kbit prio 8 maxburst 20 avpkt 1000

tc class add dev eth0.12 parent 3:0 classid 3:1 cbq bandwidth 1Mbit rate 1Mbit
allot 1514 cell 8 weight 100kbit prio 8 maxburst 20 avpkt 1000

tc class add dev eth0.13 parent 4:0 classid 4:1 cbq bandwidth 1Mbit rate 1Mbit
allot 1514 cell 8 weight 100kbit prio 8 maxburst 20 avpkt 1000

tc class add dev eth0.14 parent 5:0 classid 5:1 cbq bandwidth 1Mbit rate 1Mbit
allot 1514 cell 8 weight 100kbit prio 8 maxburst 20 avpkt 1000



# and finally add a filter to each of the classes created

tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match u32 0 0 action
mirred egress redirect dev eth0.11

tc filter add dev eth0.11 protocol ip parent 2:0 prio 2 u32 match u32 0 0
action police rate 990kbit burst 10kbit conform-exceed pipe/continue
action mirred egress redirect dev eth0.12
```

```
tc filter add dev eth0.12 protocol ip parent 3:0 prio 2 u32 match u32 0 0
action police rate 990kbit burst 10kbit conform-exceed pipe/continue
action mirred egress redirect dev eth0.13

tc filter add dev eth0.13 protocol ip parent 4:0 prio 2 u32 match u32 0 0
action police rate 990kbit burst 10kbit conform-exceed pipe/continue
action mirred egress redirect dev eth0.14

tc filter add dev eth0.14 protocol ip parent 5:0 prio 2 u32 match u32 0 0
action police rate 990kbit burst 10kbit conform-exceed drop

exit 0
```

The scripts for mgen to generate the traffic with the desired traffic rates are presented next, the different scripts were run in the Raspberry Pi previously labelled as *Traffic Generator*.

For the rate of 0.8 Mbps:

```
#originate 2 UDP flows with the combined traffic of 0.8 Mbps

#0.0 ON 1 UDP SRC 5002  DST 192.168.89.62/5002 POISSON [100 500]
0.0 ON 1 UDP SRC 5002 DST 192.168.89.62/5002 PERIODIC [100 500]


#0.0 ON 2 UDP SRC 5000  DST 192.168.89.61/5000 POISSON [100 500]
0.0 ON 2 UDP SRC 5000 DST 192.168.89.61/5000 PERIODIC [100 500]


60.0 OFF 1
60.0 OFF 2
```

For the rate of 2.4 Mbps:

```
#originate 2 UDP flows with the combined traffic of 2.4 Mbps

#0.0 ON 1 UDP SRC 5002  DST 192.168.89.62/5002 POISSON [300 500]
0.0 ON 1 UDP SRC 5002 DST 192.168.89.62/5002 PERIODIC [300 500]


#0.0 ON 2 UDP SRC 5000  DST 192.168.89.61/5000 POISSON [300 500]
0.0 ON 2 UDP SRC 5000 DST 192.168.89.61/5000 PERIODIC [300 500]


60.0 OFF 1
60.0 OFF 2
```

For the rate of 4 Mbps:

```
#originate 2 UDP flows with the combined traffic of 4 Mbps

#0.0 ON 1 UDP SRC 5002  DST 192.168.89.62/5002 POISSON [500 500]
0.0 ON 1 UDP SRC 5002 DST 192.168.89.62/5002 PERIODIC [500 500]


#0.0 ON 2 UDP SRC 5000  DST 192.168.89.61/5000 POISSON [500 500]
0.0 ON 2 UDP SRC 5000 DST 192.168.89.61/5000 PERIODIC [500 500]
```

```
60.0 OFF 1
60.0 OFF 2
```

And finally, for the rate of 5.6 Mbps:

```
#originate 2 UDP flows with the combined traffic of 5.6 Mbps

#0.0 ON 1 UDP SRC 5002  DST 192.168.89.62/5002 POISSON [700 500]
0.0 ON 1 UDP SRC 5002 DST 192.168.89.62/5002 PERIODIC [700 500]


#0.0 ON 2 UDP SRC 5000  DST 192.168.89.61/5000 POISSON [700 500]
0.0 ON 2 UDP SRC 5000 DST 192.168.89.61/5000 PERIODIC [700 500]


60.0 OFF 1
60.0 OFF 2
```

Script to show the statistics of the queues, classes and filters of all the used interfaces, located in the intermediary Raspberry Pis:

```
#!/bin/bash

printf "################ Printout STATS: ###################\n"
printf "\n"
printf "++++++++++ qdisc stats ++++++++++\n\n"
printf "$(tc -s -d qdisc show dev eth0)\n"
printf "$(tc -s -d qdisc show dev eth0.11)\n"
printf "$(tc -s -d qdisc show dev eth0.12)\n"
printf "$(tc -s -d qdisc show dev eth0.13)\n"
printf "$(tc -s -d qdisc show dev eth0.14)\n"
printf "\n"
printf "++++++++++ class stats ++++++++++\n\n"
printf "$(tc -s -d class show dev eth0)\n"
printf "$(tc -s -d class show dev eth0.11)\n"
printf "$(tc -s -d class show dev eth0.12)\n"
printf "$(tc -s -d class show dev eth0.13)\n"
printf "$(tc -s -d class show dev eth0.14)\n"
printf "\n"
printf "++++++++++ filter stats ++++++++++\n\n"
printf "$(tc -s -d filter show dev eth0)\n"
printf "$(tc -s -d filter show dev eth0.11)\n"
printf "$(tc -s -d filter show dev eth0.12)\n"
printf "$(tc -s -d filter show dev eth0.13)\n"
printf "$(tc -s -d filter show dev eth0.14)\n"
printf "\n"

exit 0
```