Software Defined Telecommunication Networks

Frank Slyne

A Thesis submitted to the

University of Dublin, Trinity College

For the degree of

Doctor of Philosophy

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

Dated: 14th October 2016

Permission

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Dated: 14th October 2016

Summary

The paradigm of Software Defined Networking can have significant beneficial impacts on the provision of traditional Telecommunications services, but there is a possibility that networks may be oversimplified by removing hidden but important components.

We evaluate the impacts of applying Software Defined Network principles to constraints that have been built in, over time, into traditional Telecommunications networks.

We adopt a two stranded approach. The first strand evaluates the interaction between a state of the art control plane and a converged network (Long reach PON) architecture, through the application of a number of typical but important scenarios. The first strand gathers data from physical testbeds that were constructed specifically for the experiments. The second strand evaluates innovation in the Layer 2 data plane, made possible by the application of SDN principles, again through the application of a number of typical scenarios. The second strand relies on a mix of simulation, predominantly, and physical experimentation.

To evaluate the effect of SDN on the converged network architecture, we construct a number of testbeds involving substantial state-of-the-art components that create an end-to-end telecommunications network. A number of testbeds are used that facilitate different technological aspects of the network, as well as the skillsets of the centres involved. The complexity of the testbeds and their integrations developed over time to reflect the availability of components. The experiments that were execute involved the performance and capability in the provisioning of high capacity bandwidth, as well as the speed of failover of network paths across a wide area, that is both on the scale of a National Network as well as a Continental Network. The experiments are executed a number of times, to understand any underlying artefacts in the interaction between the control plane and the data plane. The Protection use case exemplifies how path integrity in the Core and TDM-DWDM LR-PON based Access Metro network of a Telecommunications network can be assured through logical protection. The protection experiment demonstrated a dual-homed LR-PON protection mechanism where backup OLTs are shared among PONs in an N:1 scheme and the service restoration is provided over an end-to-end Software Defined Network. The DWA use case exemplifies how capacity constraints in one PON channel may be overcome by re-allocating dynamically one or more end user ONUs to a different channel in order to assure quality of service. This could also be used for the opportunistic provision of high bandwidth services (on-demand video and big data transfers), to specific PON users on a dynamic basic.

To evaluate the Data Plane architecture aspects, we propose and model a design for a flat Telecommunications architecture that is theoretically more scalable and efficient when compared to traditional architectures. This architecture is called FLATLANd (short for Flat Layer Two Telecommunications Network). The proposed structure provides a number of

benefits. Firstly, the architecture is strictly flat and conducts all traffic at a single layer – that is layer 2 without the use of tunnelling, VPN nor labels. Secondly, the architecture is inherently Open Access in that no one network nor service provider dominates over the others, as is the case in traditional wholesale and retail models for broadband access networks. Thirdly, the addressing is extremely scalable and granular, accommodating many terminating nodes as well as service types. Rather than preserving legacy devices such as B-RAS in their physical or virtual form, we re-architect the entire network from first principles. We target in particular next generation optical broadband networks, and take into consideration the convergence of access and metro networks, using the Long-Reach PON (LR-PON) architecture.

Acknowledgment

First and foremost, I would like to thank my PhD supervisor Assistant Professor Marco Ruffini who has given me a tremendous amount of guidance and support in my journey of research in Optical Network Architectures. Along with Professor David Payne, he afforded me the opportunity and privilege to be associated with the DISCUS Long Reach PON European research project.

I would like to thank Seamas McGettrick for all his help in integrating the Xilinx FPGA boards with the testbeds and readying the LR-PON OLT and ONU for prime time. We spent long evenings attempting to capture alien packets and tracking down galloping packets.

I would like to thank Nattapong Kitsuwan, Christian Blumm and Aleksandra Kaszubowska for their help on the testbed integrations and the DISCUS demonstrations. And for the courtesy from the others in the ONA group and the wider CTVR / Connect organisation, in particular Catherine Keogh and Mark Cooney.

I would like to thank Giuseppe Talli, Nicola Brandonisio and others from the Tyndall Photonics Centre for all the support in integrating the photonic components with the PON protocol layer and SDN control plane. The results were some impressive demonstrations at ECOC 2015 and the DISCUS plenary in December 2015.

The pan-European Openflow testbed integration could not have been done without the support of Eoin Kenny and Andrew Mackarel of HEAnet, and Robert Szuzman, Peter Szegedi, Jerry Sobieski and Sebastiano Buscaglione from the GEANT partners.

I would like to thank Jose Manuel Gran and Victor Lopez from Telefonica for the support in integrating no less than three EU research projects DISCUS, STRAUSS and IDEALIST.

Lastly, but most importantly, I would like to express my appreciation to my daughters Katie and Hannah and my wife Marguerite, for enduring the time I committed in doing my PhD, and being resigned to my interests in engineering and all things technlology.

This material is based upon works supported by SFI Grant No. 10/CE/I1853 and No. 14/IA/2527

University of Dublin, Trinity College

September 2016

Table of Contents

Chapter 1 Introduction 1

1.1 Background to the research 1

1.2 Architectures for network convergence 6

1.3 Flat Layer 2 Networks 8

1.4 Overview of methodology 11

1.5 Key Contributions 13

1.6 Dissertation Outline 14

1.7 Publications arising from this work 16

Chapter 2 State of the Art 19

2.1 Software Defined Networks 19

2.1.1 SDN at Layer 2 and Layer 3 20

2.1.2 SDN at Layer 1 and Layer 2 21

2.1.3 Network Function Virtualisation (NFV) characteristics 23

2.1.4 Frameworks for Software Defined Networks 24

2.1.5 SDN in Access Networks 26

2.2 Network Performance and Quality of Service 27

2.2.1 Causes of Poor Performance 27

2.2.2 Remediating Bufferbloat 30

2.3 QoS Frameworks 31

2.4 Flow-based QoS Frameworks 33

2.5 How Network Performance is Benchmarked 34

2.6 Data Plane Design 36

2.6.1 Traffic Conditioning 37

2.6.2 Network Node Structure 38

2.6.3 Architectural Constraints 39

2.7 Tree Networks 41

Chapter 3 SDN Control Plane for Converged Architecture 44

3.1 Functional Components 45

3.1.1 Network Orchestrator 45
3.1.2 Core Network Controller 45
3.1.3 Access Network Controller 45
3.1.4 Open Flow Agent 50
3.1.5 PON ONU and OLT 53
3.1.6 Distributed Message Queue 54
3.2 Messages 55
3.2.1 Control Plane Messages 55
3.2.2 Openflow Messages 57
3.2.3 PON wrapper methods 58
3.2.4 Event Plane Messages 59
3.3 Sample Configuration 60
Chapter 5 Converged Architecture Fast Protection 62
5.1 1:1 Protection Scheme with Pan-European Core 63
5.1.1 Configuration 63
5.1.2 Results 64
5.2 N:1 Protection Scheme with Pan-European Core 66
5.2.1 Configuration 66
5.2.2 Testing Procedure 68
5.2.3 Results 71
5.3 N:1 Protection Scheme with PON Physical layer 72
5.3.1 Configuration 72
5.3.2 Results 73
5.4 Summary 76
Chapter 6 Converged Architecture DWA 78
6.1 Assured Capacity on a new Channel 79
6.2 Results 80
6.3 Interworking between DISCUS and IDEALIST testbeds 81
6.4 Results 82
6.5 Summary 84

Chapter 7 Performance Evaluation - NSIM 86
7.1 Generic Functionality 87
7.2 Network Functionality 91
7.3 TCP protocol 94
Chapter 8 FLATLANd Architecture 97
8.1 Outline of FLATLANd Architecture 97
8.2 Architectural Patterns 102
8.3 Functional Validation 105
8.4 Performance Scenarios 107
8.5 Performance Results 117
8.6 Protocol Efficiency 123
8.7 New Protocols 125
8.7.1 TCP over Ethernet (TCPoE) 125
8.7.2 UDP over Ethernet (UDPoE) 126
Chapter 9 Discussion 128
Chapter 10 Conclusions and future work 138
10.1 Recommendations for future work 140
Chapter 12 Appendix 142

Figures

Figure 1 - Today's FTTH telecommunications architecture 2
Figure 2 - DISCUS Architecture for Long-Reach PON (LR-PON) 6
Figure 3 - DISCUS metro/core node architecture 7
Figure 4 - State of the Art FTTH telecommunications 9
Figure 5 - FLATLANd FTTH architecture-level diagram 11
Figure 6 - Multi-layer Traffic Conditioning 36
Figure 7 - EU FP7 SPARC (left). Project NANDO (right) 41
Figure 8 - Layer 2 Datacentre Architectures (Trill, SPB, VL2, and Portland) 42
Figure 9 - Logical SDN Architecture 44
Figure 10 - Database Administration Interface 47
Figure 11 – OLT Fast protection mechanism 51
Figure 12 - OFPPortModPropOptical stream_id 55
Figure 13 - Example topology 60
Figure 14 - Install Red Traffic flows 61
Figure 15 - Install Green flows 61
Figure 16 - Release Red and Green Traffic flows 61
Figure 17 - Fast Protection Scenario 62
Figure 18 - Modelled combined LR-PON access and core network, with multi-tier Control Plane 63
Figure 19 - Fast Recovery in access and core 65
Figure 20 - Multi-tier protection events. 66
Figure 21 - Logical view of combined LR-PON access and SDN Core network 67
Figure 22 – Event plane based on distributed ZeroMQ Message Queue 70
Figure 23 - Switchover time (milliseconds) for 50 iterations of N:1 protection experiment 72
Figure 24 - Network level view of the demonstration 73
Figure 25 - Protection Experiment 73
Figure 26 - Protection Message Flow 74
Figure 27 - Protection Timings 75
Figure 28 - Service restoration time for the protection mechanism and the DWA through the implemented SDN control plane 75
Figure 29 - Timings Trace 76
Figure 30 – (a) VOD with Assured Capacity. (b) Assured capacity on new channel 78
Figure 31 - DWA Scenario 80
Figure 32 - Experimental Lab Set up 81
Figure 33 - Workflow Steps 83
Figure 34 - Whireshark capture 83
Figure 35 - JSON object for a COP service-call set-up 84

Figure 36 - NSIM Scheduler and Distributed Processes 87
Figure 37 - Duplex Process 88
Figure 38 - Stack 89
Figure 39 - NSIM example scenario A 90
Figure 40 - NSIM example scenario B 90
Figure 41 - Host Stack 91
Figure 42 - Datalink - based on Duplex block 92
Figure 43 - NSIM switch types 92
Figure 44 - NSIM example scenario C 93
Figure 45 - NSIM example scenario D 93
Figure 46 - TCPDump of link.pcap 93
Figure 47 - NSIM example scenario E 93
Figure 48 – NSIM example Scenario F 94
Figure 49 - NSIM Example Scenario G 95
Figure 50 - FLATLANd FTTH function-level diagram 97
Figure 51 - 48-bit Address Range 100
Figure 52 - FLATLANd Network Function Container 100
Figure 53 - Address Scheme 102
Figure 54 - FLATLANd Multi-service / Open Access Pattern 103
Figure 55 - FLATLANd Traffic Regulation Pattern 103
Figure 56 - FLATLANd Protection Pattern 104
Figure 57 - Network Function Pattern 104
Figure 58 - Service Registration carried out on SDN/NFV testbed 106
Figure 59 - Classic Model simulation 109
Figure 60- FLATLANd Model Simulation 110
Figure 61 - Shared Configuration 112
Figure 62 - Scenario 0 112
Figure 63 - Scenario 2 113
Figure 64 - Scenario 3 115
Figure 65 - Scenario 4 116
Figure 66 - Scenario 5 116
Figure 67 - Scenario 6 117
Figure 68 - Traffic Flows 118
Figure 69 - UDP performance metrics 119
Figure 70 - TCP performance metrics 121
Figure 71 - TCP Congestion Window 122
Figure 72 – TCP Round Trip Time (RTT) 122
Figure 73 - TCP over Ethernet packet trace 126

Figure 74 - UDP over Ethernet packet trace 126
Figure 75 - UDPoE host configuration 127
Figure 76 - UDPoE protocol stack utilisation 127
Figure 77 - UDPoE performance for scenario 4 127
Figure 78 - (a) Computer Engineering Curriculum. (b) Computer Science Curriculum 130
Figure 79 - Continuous Delivery life cycle [128] 132
Figure 80 - Downstream TCAM 145
Figure 81 - Upstream TCAM 146

Equations

Equation 1 - Packet drop probability 91

Tables

Table 1 - Loading Scenarios 39
Table 2 - Example of information in the PATHS table 47
Table 3 - Example of information in the WAVELENGTH table 48
Table 4 - Example of information in table “SW_PORTS” 48
Table 5 - Example of information in the HOST_IP table 49
Table 6 - Example of information in the BW table 50
Table 7 - List of Main Control Plane Messages 56
Table 8 - PON Methods 59
Table 9 - Association of Message Queue types and testbed components 71
Table 10 - DWA timings 80
Table 11 - Comparison of Flow based Networks 99
Table 12 - Simulation scenarios 111
Table 13 - Protocol Efficiency, Classic Architecture 123
Table 14 - Protocol Efficiency, FLATLANd Architecture 123
Table 15 - Network operations - Classic Architecture 124
Table 16 - Network operations - FLATLANd Architecture 124
Table 17 - Features and Benefits of SDN FLATLANd architecture 137
Table 18 - Internet Statistics 150

Chapter 1 Introduction

1.1 Background to the research

Telecommunications networks have been slow to adapt to meet the needs of high speed ubiquitous communication services. Telecommunications networks were originally commissioned in the first decades of the 20th century to provide POTS services (Plain Old Telephone Service). They comprised of large copper based cable networks extending from customer premises to local exchange buildings, where phone calls were switched through a hierarchy of national and international network transmission lines until they connected with their intended destination. Original switching equipment was mechanical and required significant building accommodation close to population centres, making the operation and ownership of the Telephony network and its assets significant responsibilities. In most countries, there was only one Telephone company, which operated as part of the function of the state, prohibiting other companies from providing telecommunications services.

Much of the architectural topology and network layers (Figure 1) has remained unchanged for many years. Typically, there is an Access Network which provides geographical reach, so all customers can have a network termination. The Access Network typically comprises the copper cable in the ground, which is costly to maintain and replace. The Metro Network concentrates traffic, so network traffic can be handled more efficiently. There can also be capabilities in the Metro Network to switch or redirect traffic between customers located off the same network portion. Typically, there might be a number of Metro Nodes in large geographical regions, towns and cities. The Core Network is at the top of the Telecommunications network hierarchy. It is here that there is the highest level of traffic concentration and requirement for network resilience and redundancy.

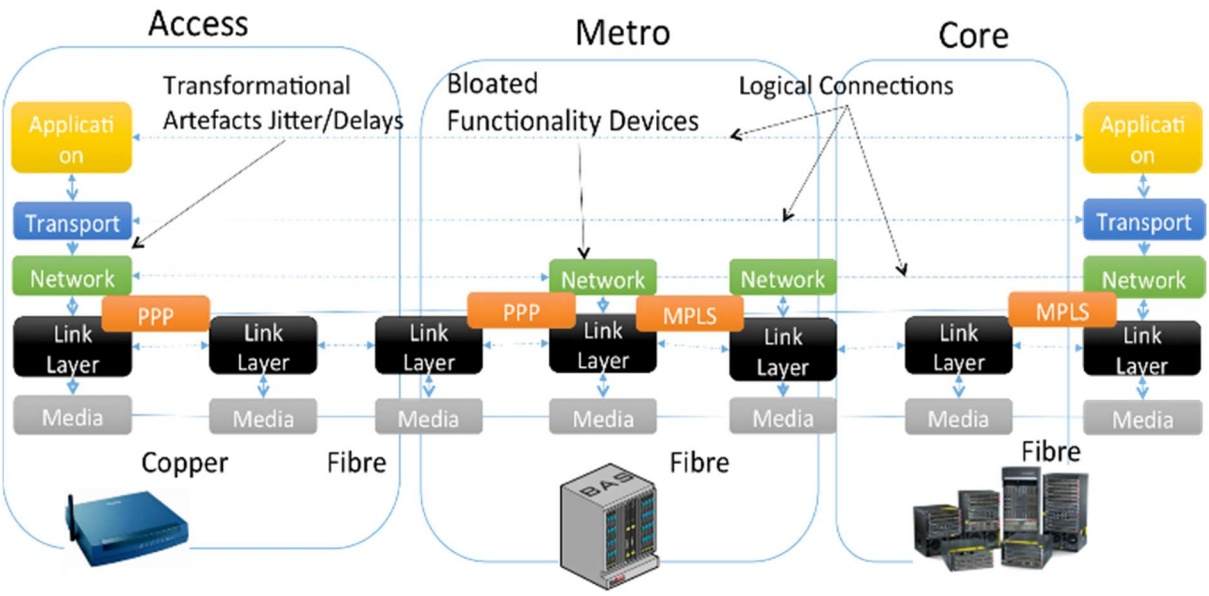


Figure 1 - Today's FTTH telecommunications architecture

The adoption of new technologies such as Fibre Optics has been most pronounced in the core network, since this is where there is most competition between incumbent (divested) telecommunication providers such as BT, France Telecom, Eircom, and wholly commercial companies such as Amazon, Google and Facebook. Unfortunately, migration to fibre optic has been slower in the access because there is more regulation and less competition. With less competition, there is less incentive for network operators to provide equivalent services such as high speed broadband, to both urban and rural customers.

With the advent of rudimentary data services such as public packet data service, a dedicated network was either built separately or over-laid on top of the existing telephony infrastructure. The building of a separate network made services expensive and thus not attractive to customers, while over-lay networks reinforced the existing sub-optimal architecture and topology. Newer services such as broadband access, Internet and GSM have been over-laid on existing telecommunications networks. Each service has required its own network components and management system, typically from different vendors. Different access and metro components, for telephone, broadband, IP (Internet Protocol) and GSM , serving similar customer or service groupings are split geographically, leading to inefficiencies. With each horizontal hop and with each vertical layer that data must transit through requires processing, more energy and computational processing must be expended, which in turn can cause performance artefacts such as latency, jitter and packet loss.

Stimuli for legacy networks upgrades

There are three types of forces that stimulate the upgrade of legacy networks. These are economic, policy and regulatory. In Europe, the main motivation for incumbent European

Telco's to invest in new technologies such as Fibre to the Home (FTTH) are Revenue Attrition, User Demand for Higher Bandwidth, New Application Devices, Competition, Political Will, Address High Cost Base, Future-proofing and Regulatory Relief [1]. In particular, the type and volume of services that customers consume will change considerably over next few years. In the course of 5 years from 2015 to 2020, the CISCO VNI index [2] predicts that the total number of Internet users in the UK will increase 10% to 62 Million, with the average fixed broadband speed increasing from 24.7 Mbps to 51.3 Mbps, and the average WiFi speeds increasing from 17.4 Mbps to 35 Mbps. The nett result is that the average combined Internet UK traffic will increase 2.9 fold from 5 Tbps in 2015 to 20 Tbps in 2020, with the Busy Hour traffic increasing 4.5 fold from 21 Tbps to 117 Tbps. At a European level, the Digital Agenda for Europe (DAE) [3] defines the policy objectives in relation to broadband infrastructure with which national government agencies such as the Department of Communications in Ireland should comply. The key targets of the DAE agenda are that all European citizens should have access to broadband internet with speeds of at least 30MB/s by the year 2020 with 50% of users subscribing to broadband with speeds of over 100MB/s. The short-term policy target was to have universal broadband provision by 2013. In Ireland, the objective of the Communications Sector of the Department of the Communications, Energy and Natural Resources is "*to contribute to sustained macro-economic growth and competitiveness and ensure that Ireland is best placed to avail of the emerging opportunities provided by the information and knowledge society, by providing a supportive legislative and regulatory environment and by developing a leading edge research and development reputation in the information , communications and digital technologies*".

Technological Environment

Other communication systems, most notably the Internet have also faced issues related to legacy technologies. This is surprising since the Internet is a much more recent and open network than a traditional telecommunications network. Central to the issues facing the Internet is the fact that the reach and importance of the Internet had grown exponentially in the 1990's and 2000's. From a base level of 200 hosts in 1980, the Internet grew substantially to 570 million hosts in 2008 [4]. Applications that are congestion-sensitive can hog bandwidth resources needed by other applications, which made it unattractive for companies to run commercial services over the internet. The poor returns for commercial use of the Internet lead to under-investment in capacity [5]. A flaw in the Internet's core routing algorithm in 1989 caused the entire Internet to fail. The impact was a mere inconvenience for the several thousand researchers who were the used the Internet then for academic purposes. This is in contrast to the effect of the SQL slammer attack in 2003 which caused over a billion dollars in damages to business [6] including the outage of

commercial airline flights reservation systems and thousands of Automatic Teller Machines lasting for days.

These catastrophic events had been predicted [7] in advance of them happening. This is because since its inception, the Internet has developed in an evolutionary and reactive manner, likened, at times, to patch being applied upon patch to network protocols and network. Physical components such as routers and switches which make up the core of the Internet must comply with in excess of 5400 RFCs. An RFC (Request for Comment) is a specification of a protocols or functionality, created by the Internet Engineering Task Force (IETF) that is essential to the operation of the Internet. This has made routers, which are critical to the functioning of the internet, bloated with functionality that is in many cases redundant. Each device requires code exceeding 20 Million lines, switching logic spanning 500 million gates and over 10 GBytes of RAM. Paradoxically, the Internet, which was initially designed to be open and free of regulation had itself become an impediment to Innovation [8]. This barrier to innovation is evident where important enhancements such as multicast, Mobile IP and Quality of Service sit on top of the IP layer [4] and have not been fully embedded in the Internet architecture. Incorporation of this functionality would require significant upgrades in the physical components such as routers and switches with a high risk that existing functionality would break. Typically, functionality may be provided as a patch by individual vendors to their equipment, which adds to the complex melange of functionality that routers and switches have to currently support.

The approach to how the stakeholders of the Internet have addressed these problems is different from how the operators of commercially run companies tackle the issues of legacy Telecommunication networks. There were a number of initiatives to both document the deficiencies of the current Internet and to define the architecture and functionality of the future Internet [7]. Some protagonists advocated an incremental or evolutionary approach so as to ensure compatibility with the current Internet. The NewArch [9] initiative advocated a revolutionary approach that would explore the technical consequence of a combination of top-down architectural reasoning and simulation and prototyping of a new architecture. This would speed up innovation and thus prevent legacy issues from been carried forward into the future Internet. In 2005, a panel of US academics instigated the NSF future Internet [10] project. This was followed by the development of the GENI experimental facility [6] and the NSF FIND programme [11]. NSF FIND was an important influence on Internet architecture concepts worldwide –in Europe [12], in Japan on the JGN+ testbed which supports the Japanese AKARI Next Generation Network [13], and in Korea on the KOREN2 experimental network. The EU-FP7 CaON cluster of Future Network projects were heavily influenced by the US NSF FIND/GENI initiatives.

Unfortunately, the GENI research plan was not universally admired [4]. Some elements in the research community criticised it for being too broad in focus. Others said it lacked a

classical scientific approach [14]. Stanford University were doubtful that a group of Computer Scientists could 'champion big ideas' such as the re-architecting of the internet. Instead, there could be better return on resource and effort through the embedding of research in so-called "CIO type" organisations. A CIO (Chief Information Office) has oversight of all technologies in an organisation, both network and IT. This would provide focus for the application of research and thus yield the necessary efficiency and effectiveness. While the GENI research network [6] proposed large infrastructure and a structured/formal approach to innovation, the Openflow initiative instigated out of Stanford University has gained significant community support.

Openflow is significant in that it has caught the imagination of both technology and commercial entities in the Internet, so much so that it is the stimulus behind the Software Defined Network initiatives. Openflow is a component (in terms of a protocol and a suite of applications) that can be evaluated and deployed by the research institutions, given their own network and resources. Openflow recognises that the transfer of IP packets is founded on flow and forwarding tables to be found in all switches and routers. While the structure of the flow tables may differ from vendor to vendor, the basic functionality is quite similar. Openflow separates the control plane decision making process from the action of passing IP. The control plane for all devices in a network can be aggregated and centralised where there are sufficient resources for path, switch and routing computation. The functionality of data plane components such as Switches and Routers can be simplified, and a common Optical infrastructure can be partitioned to provide virtual test bed resources, equivalent to those proposed by GENI [8]. Networks which are flatter and have fewer hops can be created using Fibre optics and end to end networking protocols such as the Internet Protocol. These protocols work at different levels in the Internet stack, and thus can work together.

Software Defined Network paradigm should be as applicable to the Telecommunications Industry as it is to the Internet Industry, with benefits to be applied at different levels in the technology stack.

1.2 Architectures for network convergence

Passive Optical Network architectures such as Long Reach PON (LR-PON) [15] make the Access network entirely fibre based. The migration to fibre access changes the characteristic of household Internet usage with households with fibre access consuming considerably more Internet (up to 20% more or 608.5 GB in total) than those with traditional copper access [2]. This removes legacy and redundant Optical-Electrical-Optical (OEO) transitions but also concentrates geographical functionalities and interfaces (such as Layer 1 to Layer 2, Layer 2 to Layer 3) for efficiencies. The reduction in OEO transitions has the benefit of reducing power consumption. The main drivers for power reduction research are usually economical (reducing the energy cost), technical (reducing the associated heat dissipation) and environmental (reducing the carbon footprint) reasons [16].

Figure 2 shows the Flat Core of the LR-PON architecture where the core switches are partially or fully meshed. Metro-Core Nodes perform traffic aggregation closer to the customers. Passive Optical Networks are composed of customer side ONU devices and Metro Access OLT devices, between which the PON protocol runs. In protocols derived from the GPON protocol, the upstream protocol is based on TDM (Time Division Multiplexing), whilst in the upstream traffic is statistically multiplexed. Fairness of usage is maintained using a Dynamic Bandwidth Algorithm (DBA). From a practical layer 2 perspective, the Ethernet protocol runs throughout the network.

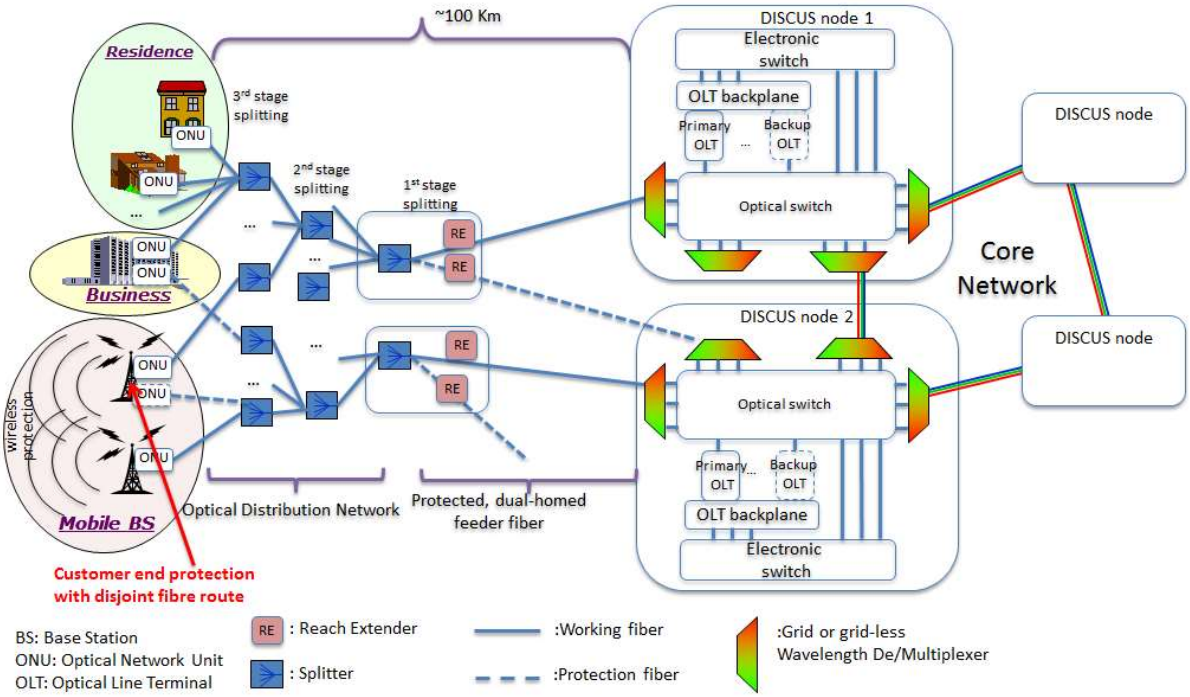


Figure 2 - DISCUS Architecture for Long-Reach PON (LR-PON)

In a LR-PON architecture (Figure 2), much of the currently protected metro network is replaced by long feeder fibres. The probability of failure is higher due to higher probability of the long feeder fibres being cut and optical amplifiers in the first splitter failing.

Remediating the broken fibre feeder is not easy and may take hours if not days. The impact of failure is also high, not only because many users are affected, but the types of services supported by the PONs may be of high value (i.e., can include backhauling and other business services). The LR-PON feeder fibres are replacing part of the current network that offers protection from failure and for this reason protection mechanisms become a requirement in LR-PON. Fast protection is required in order to fulfil user requirements for converged multi-service shared PON environment, particularly for enterprise and mobile backhaul applications.

The DISCUS metro/core nodes are core edge nodes in a similar architectural position in the network to what are often called metro-core nodes in classic telecommunication architectures. The DISCUS Metro/Core node are the only nodes in the network covered by a single optical island with traffic processing functions. The architecture of these nodes is flexible enough so that different (IP, Ethernet and Optical) layers can evolve and if necessary displace other layers minimising cost and energy consumption. The node architecture consists of an optical switching layer, an Ethernet layer and an IP layer. The optical switch provides flexible interconnect between theses layers and the optical channels from the access and core networks. The large port-count optical switch allows maximum flexibility as any incoming fibre can be terminated, after de-multiplexing, at any OLT (Optical Line Termination), or can be re-amplified and sent back to another ONU or regenerated and sent over the optical core network. Since every access PON can carry a large number of wavelengths, potentially 80 or more in the medium to long term, the optical switch must be highly scalable, while offering a very low optical loss (less than 2-3dB). The Optical Switch should have large switch matrices and a potentially be 3-stage switches capable of scaling to over 12000 ports.

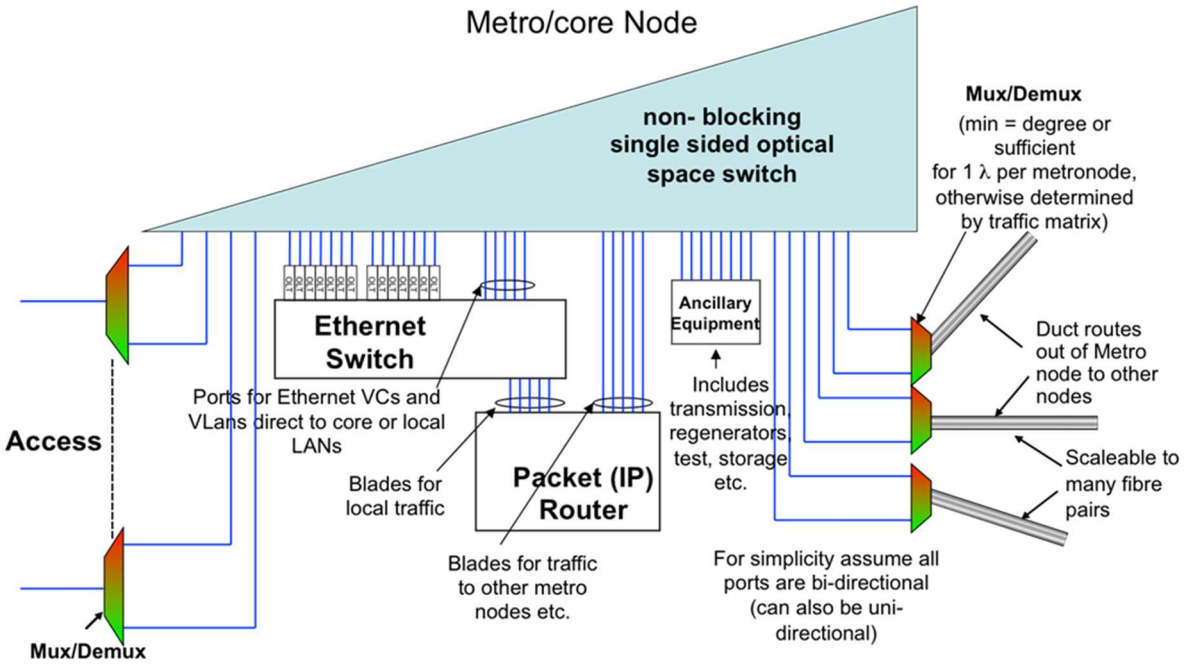


Figure 3 - DISCUS metro/core node architecture

The main approach to increasing availability of the LR-PON service is through redundancy of the feeder fibre and dual homing, which adds costs to the network and must be recouped through higher service charges. In addition, protection times may be reduced significantly by using 1+1 protection mechanisms, such as hardware optical monitoring, in the LR-PON. The downside of 1+1 protection is that downstream traffic must be replicated through both primary and secondary OLTs, so additional network ports, fibre and capacity are needed to duplicate downstream traffic. These downsides may be ameliorated if N:1 or 1:1 protection were possible in a granular, cost efficient and time responsive manner. The typical mechanisms used to provide protection in the core are based on routing (OSPF) or label switched paths (MPLS). Open shortest path first (OSPF), in which packets are routed through the shortest path, takes more than a second to recover. Recovery times of this order are not acceptable in many networks where target switch over time of 50 milliseconds are common for leased line traffic or 100 milliseconds for realistic internet scenarios [17]. Multiprotocol label switching (MPLS) provides fast rerouting by a protection mechanism that uses an alternative Label Switched Path (LSP) to reroute packets from a protection point to another node or to the destination. This mechanism has to be provided locally at each switch [18]. In the access network however, protection mechanisms have not been developed as much as in the metro and core. In 2008 an experiment was carried out using commercial GPON hardware and the restoration time was found to be in the order of 30 seconds [19]. The authors believed this could be reduced to approximately 500 milliseconds if they could optimise the switching, ranging and registration mechanisms of the GPON system. The same operator published in 2013 an updated protection mechanism using VLAN switching with an automated restoration solution, achieving protection times in the order of 4.5 s and with maximum values of 9.5 s. [20] . Fast PON protection also allows the implementation of protection load balancing schemes, such as those introduced in [21] which allow reducing substantially cost of both IP and PON backup resources by increasing the ability to share protection equipment across the network [17].

1.3 Flat Layer 2 Networks

Figure 4 exemplifies the complexity of providing Broadband service to a residential customer by a wholesale network operator through layered communications stacks. Typically, a Point-to-Point-over-Ethernet (PPPoE) tunnel extends from a B-RAS (Broadband Remote Access Service) through to a Residential Gateway located in the customer’s premises. Network designers typically use tunnels and VPNs to extend the reach of services such as PPPoE.

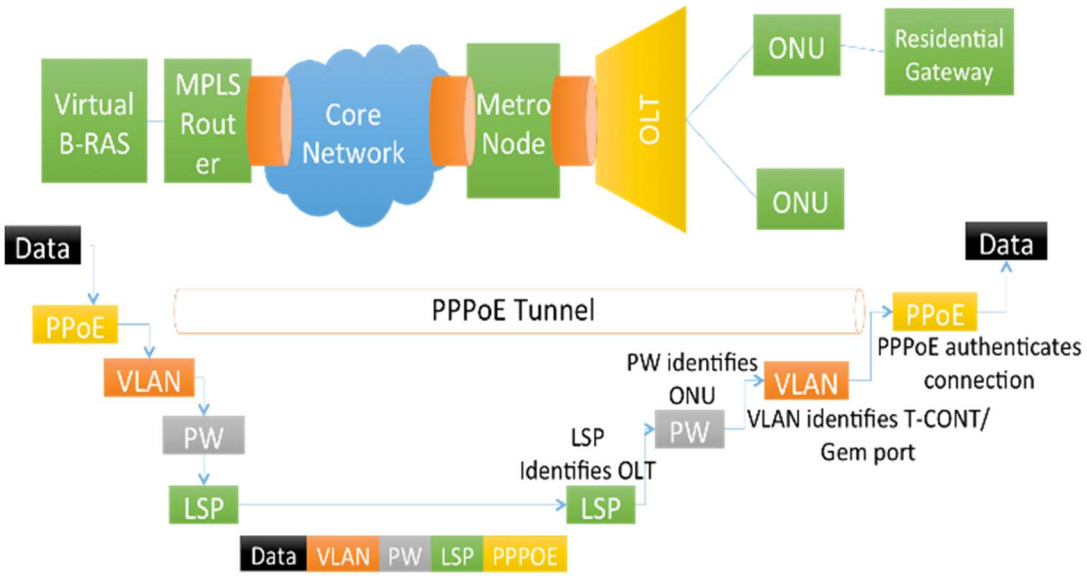


Figure 4 - State of the Art FTTH telecommunications

Here, an MPLS router tags the PPPoE tunnel with a Pseudo-Wire (PW) identifier and a Label-Switched Path (LSP) label. The PW is used to identify the path up to the Optical Line Terminal (OLT). For each OLT different PWs identify different SPs and within an SP different service types (Video-on-Demand and VOIP).After the OLT, towards the Optical Network Unit (ONU), a VLAN tag, together with the MAC address, is used by the ONU to direct traffic through a pre-determined Traffic-Container (T-CONT) and GPON Encapsulation Method (GEM) port. In the case of PPPoE, there are significant Virtual B-RAS load and capacity constraints. Tests done by BT (for example) [5] showed a maximum limit of 9,000 PPPoE sessions per virtual B-RAS (Broadband Remote Access Server). There are multiple manual configuration actions to set up new services and customers, largely due to the lack of integration between the management systems of the technology each layer and stack that underlies the service.

Excessive tunnelling and encapsulation for the transit of large connection volumes has significant downside such as restrictive network partitioning, slow reconfiguration times, and suboptimal dissociation between network platform and services. Each network layer and hop that is traversed has the potential to introduce artefacts such as jitter, Bufferbloat and cross-layer authentication requirements. Bufferbloat happens when excessively large (bloated) buffers are designed into network communication systems [22]. Systems suffering from Bufferbloat have bad latency under load under some or all circumstances, depending on if and where the bottleneck in the communication's path exists. Bufferbloat encourages network congestion; it destroys congestion avoidance in transport protocols such as HTTP, TCP and BitTorrent. Network congestion-avoidance algorithms depend on timely packet drop. Unfortunately, bloated buffers invalidate this design presumption.

The original intent of the Internet was to transmit IP datagrams over transmission links which were both unreliable and had limited by capacity. Intermediate IP routers would have to be

operationally autonomous. The TCP protocol was developed to cater for session properties such as statefulness, error control and congestion management [23]. Unfortunately, the underlying characteristics that made the Internet robust, have also been the ones that have made the Internet rigid [24] - Internet protocols such as IP, UDP and TCP do not have native support for Voice and Video Quality of Service (QoS); TCP flow control is inefficient because it is based on a slow-start mechanism; routing between large domains is cumbersome and unreliable; large-scale networks are difficult to manage.

We introduce the FLATLANd architecture [25] which uses an efficient hierarchy of low latency layer-2 switches and distributed Openflow tables (across ONU/OLT, electrical and optical switches in a LR-PON topology). In the FLATLANd architecture we apply the same concept to telecommunications networks. Any network that uses Ethernet as a layer-2 protocol can benefit from the FLATLANd architecture. From a practical layer 2 perspective, the Ethernet protocol runs throughout the network. A translation is performed between the real (physical) address of the end device and the internal structured (pseudo) addressing used within the network. In the case of LR-PON, this translation is performed at the ONU GEM port. The mechanism partitions the internal 48-bit address space of an Ethernet layer into a number of arbitrary subfields, each routed to a different part of the network.

Layer-2 Ethernet addresses of network devices and terminations are assigned during manufacturing and thus uncorrelated to their location and other devices in their vicinity. This restricts their use in switched LAN and WAN segments, due to the impossibility to create any kind of hierarchical structure in the addressing scheme and forwarding tables. Through the use of pseudo-MAC addressing, the FLATLANd architecture (Figure 5) overcomes such limitation by creating a structured Ethernet addressing domain that spans the entire network between the network terminations at the customer premise and the datacentre thus empowering wide area SDN at layer-2.

The FLATLANd architecture (Figure 5) creates a structured Ethernet addressing domain that spans the entire network between the network terminations at the customer premise and the Data Center thus empowering wide area SDN at layer-2. From a practical layer 2 perspective, the Ethernet protocol runs throughout the network. Layer-2 Ethernet addresses of network devices and terminations are assigned during manufacturing and thus uncorrelated to their location and other devices in their vicinity. This restricts their use in switched LAN and WAN segments, due to the impossibility to create any kind of hierarchical structure in the addressing scheme and forwarding tables.

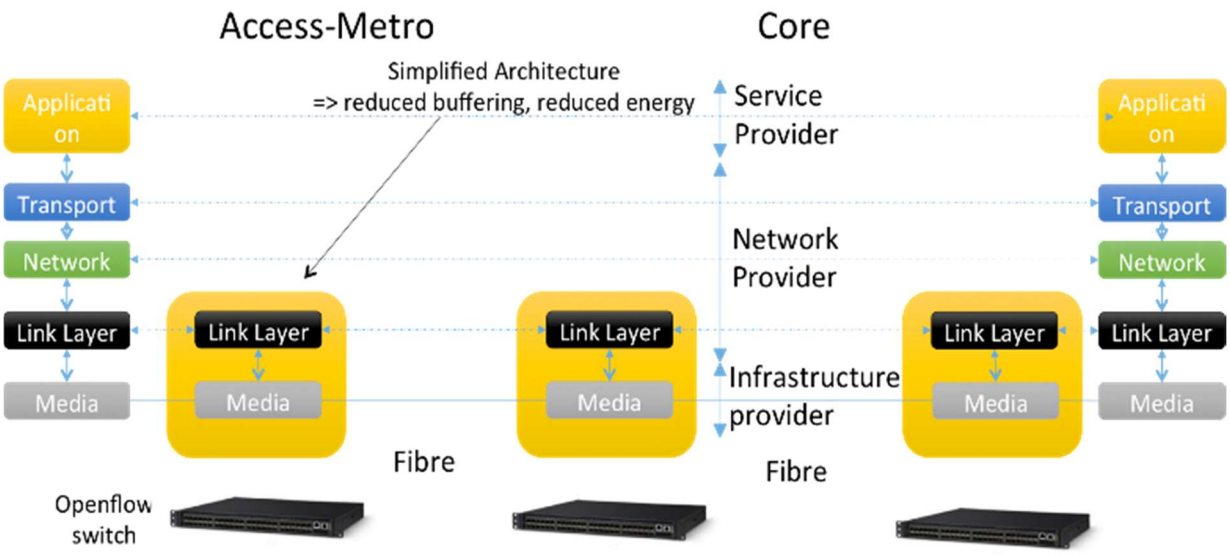


Figure 5 - FLATLANd FTTH architecture-level diagram

1.4 Overview of methodology

To conduct the research in a robust manner, we apply *Leedy and Omrod's* key principles for conducting research projects [26]. They are as follows.

Research should originate with a question or problem.

Software Defined Networking is having significant benefits for networking systems that underpin the Internet and Data Centres. Is the SDN paradigm of separating data and control planes applicable to the traditional Telecommunications Industry without oversimplification caused by the removal of hidden but important components?

Research requires clear articulation of a goal.

We evaluate the impacts of applying Software Defined Network principles to constraints that have been built in, over time, into traditional Telecommunications networks.

Research requires the collection and interpretation of data in an attempt to resolve the problem that initiated the research.

We adopt a two stranded approach [27, 28]. The first strand evaluates the interaction between a state of the art control plane and a start of the art Passive Optical Network, through the application of a number of typical but important scenarios. The second strand evaluates innovation in the Layer 2 data plane, made possible by the application of SDN principles, again through the application of a number of typical scenarios. The first strand adopt an approach similar to Action Research where data is gathered from physical

testbeds that were constructed specifically for the experiments. Action Research follows a closed cycle iterating through the steps Action Planning, Action Taking, Evaluation, Specific Learning, Diagnosing [29]. The second strand adopts an approach of Concept Implementation and Protocol Analysis and Simulation [30], in that we rely on a mix of simulation and experimentation.

Research requires a specific plan for proceeding.

We construct a number of testbeds involving substantial state-of-the-art components that create an end-to-end telecommunications network. A number of testbeds are used that facilitate different technological aspects of the network, as well as the skillsets of the centres involved. The complexity of the testbeds and their integrations developed over time to reflect the availability of components. The experiments that were execute involved the performance and capability in the provisioning of high capacity bandwidth, as well as the speed of failover of network paths across a wide area, that is both on the scale of a National Network as well as a Continental Network. The experiments are executed a number of times, to understand any underlying artefacts in the interaction between the control plane and the data plane. The Protection use case exemplifies how path integrity in the Core and TDM-DWDM (Dense Wavelength Division Multiplexing) LR-PON based Access Metro network of a Telecommunications network can be assured through logical protection. The protection experiment demonstrated a dual-homed LR-PON protection mechanism where backup OLTs are shared among PONs in an N:1 scheme and the service restoration is provided over an end-to-end Software Defined Network. The DWA (Dynamic Wavelength Assignment) use case exemplifies how capacity constraints in one PON channel may be overcome by re-allocating dynamically one or more end user ONUs to a different channel in order to assure quality of service. This could also be used for the opportunistic provision of high bandwidth services (on-demand video and big data transfers), to specific PON users on a dynamic basic.

To evaluate the Data Plane architecture aspects, we propose and model a design for a flat Telecommunications architecture that is theoretically more scalable and efficient when compared to traditional architectures. This architecture is called FLATLANd (acronym for Flat Layer Two Telecommunications Network). The proposed structure provides a number of benefits. Firstly, the architecture is strictly flat and conducts all traffic at a single layer – that is layer 2 without the use of tunnelling, VPN nor labels. Secondly, the architecture is inherently Open Access in that no one network nor service provider dominates over the others, as is the case in traditional wholesale and retail models for broadband access networks. Thirdly, the addressing is extremely scalable and granular, accommodating many terminating nodes as well as service types. Rather than preserving legacy devices such as B-RAS in their physical or virtual form, we re-architect the entire network from first principles.

We target in particular next generation optical broadband networks, and take into consideration the convergence of access and metro networks, using the Long-Reach PON (LR-PON) architecture.

Research is, by its nature, cyclical or more exactly helical.

Particularly with respect to the evaluation of the interaction between a state of the art control plane and a start of the art Passive Optical Network. For the protection experiment, we execute a number of iterations of the experiments that a 1+1, through 1:1 to N:1 protection scenarios. The protection experiments also evolve to include more physical layer components as they became available, and also encompassed different complexities of core network such as transcontinental core networks and national networks. The dynamic wavelength assignments also evolved from experiments on stand-alone testbeds to more complicated experiments across multiple geographically spread testbeds.

Research is guided by the specific problem, question or hypothesis.

The hypothesis is the application of Software Defined Networking principles with technological developments can encourage innovation, new services and approaches to old problems and bottlenecks in Telecommunications network architectures that have become stagnated,

1.5 Key Contributions

There are three key contributions within this work.

The First contribution is the development of the control plane mechanism for a metro-access network and its implementation and tests for experiments involving path protection and Dynamic capacity assignment (both in time and wavelength domains). This involved:

- development of SDN network control and network orchestration facilities
- development of message based event plane
- development of interfaces with devices such as OLT, ONU, EDFA, Optical Switches through an Openflow Agent so they could participate in the SDN framework
- development of timing and measurement experimentation tools
- optimisation of Openflow controllers for use in real-time protection experiments
- integration between testbeds including physical layer PON, GEANT testbed, IDEALIST testbed
- Execution of experiments for both path protection and dynamic capacity assignment.

The Second contribution is the introduction of the FLATLANd (Flat Layer Two Telecommunications Network) architectural concept which exploits Software Defined

Networking concepts to provide an alternative Telecommunications architecture. The most significant contributions within the design are:

- the principle of mapping from pseudo to real MAC addresses, enabling layer 2 routing across a wide area
- the use of Openflow switches and controllers to mimic network functions such as ARP, DNS and DHCP
- the development of architectural patterns for Network Function Virtualisation, Open Access, Traffic regulation and Path Protection

The Third contribution is that the NSIM network simulator which was developed to compare the performance of classic and FLATLANd architecture models. It has support for standard network protocols such as Ethernet, IP, TCP and UDP but also fractional layer protocols such as PPP, MPLS, Dot1Q, PPPoE. It has support for network characteristics such as buffering and latency. It supports Drop-Tail and Controlled Delay (CoDel) queuing disciplines. This allowed us to demonstrate hypothetical collapsed protocol stacks such as TCPoE and UDPoE.

1.6 Dissertation Outline

We structure the thesis into a review of the State of the Art for the application of SDN to Telecommunications networks. This is followed by chapter each that deals with major contribution of the thesis.

The review of the State of the Art was completed and maintained through a number of revisions until closure of the research. State of the Art informs precedents, constraints and developments related to technology and approaches.

The State of the Art has two main strands. The first strand evaluates the interaction between a state of the art control plane and a state of the art Passive Optical Network, through the application of a number of typical but important scenarios. The candidate scenarios are the protection scenario and the Dynamic Wavelength Assignment. Because the components we are working with are purpose built, there is a high level of flexibility around how to interface with them. This allows use to investigate interaction with a novel SDN control plane which we build.

We review candidate options for SDN frameworks. While Openflow is a dominant theme in the control of layer 2 (Ethernet) and layer 3 (IP) devices such as switches and routers, it is not apparent how relevant it is to physical and optical devices. Prior to Openflow, there has been precedence in the dividing data plane from control plane in optical networks. For the experimentation, we gather data from physical testbeds that were constructed specifically for the experiments. Firstly, we look at the performance of a protection scheme for a

flattened optical access, metro and core network. We see that a network failure such as a fibre break in the access network can be detected in a number of milliseconds, with the event being transmitted to an SDN control plane for corrective decision and action to be taken. This complexity of this use case evolves from a 1:1 protection regime in the access metro with diverse paths in the core through a N:1 protection regime with diverse paths in the core built on the GEANT European Research Testbed, to an N:1 protection regime on a TDM-DWDM PON physical layer in the Access network with an emulated national core. Secondly, we look at the implementation of a bandwidth on demand scheme through Dynamic Wavelength Assignment. A request for dedicated bandwidth, equivalent to an entire wavelength can be accommodated by an SDN control plane, incorporating a Network Orchestrator and multiple Network Controllers. The complexity of this use case was developed in two ways. We conduct the experiment with a TDM-DWDM PON physical layer in the Access network with an emulated national core. Secondly, we integrate our Metro Access network with the EU-FP7 IDEALIST core. This requires integration between our network controller and the IDEALIST ABNO orchestrator. The DWA use case exemplifies how capacity constraints in one PON channel may be overcome by re-allocating dynamically one or more end user ONUs to a different channel in order to assure quality of service. This could also be used for the opportunistic provision of high bandwidth services (on-demand video and big data transfers), to specific PON users on a dynamic basic. The use case in both case involves provisioning end-to-end dedicated bandwidth between a Video Server and a Video client. The Protection use case exemplifies how path integrity in the Core and TDM-DWDM LR-PON based Access Metro network of a Telecommunications network can be assured through logical protection. The protection experiment demonstrated a dual-homed LR-PON protection mechanism where backup OLTs are shared among PONs in an N:1 scheme [and the service restoration is provided over an end-to-end Software Defined Network.

The second strand evaluates innovation in the Layer 2 data plane, made possible by the application of SDN principles, again through the application of a number of typical scenarios. The second strand relies on a mix of simulation, predominantly, and physical experimentation.

We propose and model a design for a flat Telecommunications architecture that is scalable, efficient and economic, when compared to traditional architectures. The proposed Addressing structure provides a number of benefits. Firstly, the architecture is strictly flat and conducts all traffic at a single layer – that is layer 2 without the use of tunnelling, VPN nor labels. Secondly, the architecture is inherently Open Access [31] in that no one network nor service provider dominates over the others, as is the case in traditional wholesale and retail models for broadband access networks. Thirdly, the addressing is extremely scalable (at 2^48 or 281 trillion addresses) and granular, accommodating many terminating nodes

as well as service types. We look at the FLATLANd data plane performance, and the critical functions required from the constituent data nodes. We look at the state of the art of network node design, and understand the issues that are created when large volumes of traffic need to be switched at high speed. We look at the structure of a network node, the discrete functions which must be performed on packetised traffic. Depending on the level of flow processing that needs to be performed, the network node may experience constraints, due to fixed nature of the node architecture. At large traffic volumes, it is common for a network node to experience congestion which gives rise to artefacts such as Jitter, packet loss and latency. An anomalous behaviour can crop up where ingress buffers build up quickly on network nodes with large buffers, but do not dissipate normally. This behaviour is called BufferBloat.

The two strands are brought together in the section on conclusions and recommendations.

1.7 Publications arising from this work

The following is a list of papers to which I have contributed, which have been published or accepted for publication.

1. IEEE ICTON 2014 - An SDN-Driven Approach to a Flat Layer-2 Telecommunications network. Frank Slyne, Marco Ruffini
2. IEEE/OSA ECOC 2014 - Design and experimental test of 1:1 End-to-End Protection for LR-PON using an SDN multi-tier Control Plane. Frank Slyne, Nattapong Kituswan, Séamas McGettrick, David B. Payne and Marco Ruffini
3. IEIEC COMEX Letter - A Europe-Wide Demonstration of Fast Network Restoration with Openflow. Nattapong Kitsuwan, Frank Slyne, Seamas McGettrick,David B. Payne, and Marco Ruffini
4. IEEE/OSA Journal of Optical Communications and Networking. VOL. 3, NO. 2/FEBRUARY 2014 An Independent Transient Plane Design for Protection in Openflow-based Networks. Nattapong Kitsuwan, Seamas McGettrick, Frank Slyne, David B. Payne, and Marco Ruffini
5. IEEE 16th International Telecommunications Network Strategy and Planning Symposium. A Transparent Openflow-based Oracle for Locality-Aware Content Distribution. Emanuele Di Pascale, Frank Slyne, Marco Ruffini.
6. IEEE/OSA OFC 2015. Experimental End-to-End Demonstration of Shared N:1 Dual Homed Protection in Long Reach PON and SDN-Controlled Core. S. McGettrick F. Slyne, N. Kitsuwan, D.B. Payne, M. Ruffini.
7. IEEE/OSA OFC 2016, postdeadline paper. Demonstration of SDN Enabled Dynamically Reconfigurable High Capacity Optical Access for Converged Services. Giuseppe Talli, Stefano Porto, Daniel Carey, Nicola Brandonisio, Alan Naughton,

Peter Ossieur, Frank Slyne, Seamas McGettrick, Christian Blum, Marco Ruffini, David Payne, Rene Bonk, Thomas Pfeiffer, Nick Parsons, Paul Townsend.

8. [Invited] Elsevier Optical Fibre Technology special issue on Next Generation Access, Vol. 26, part A, December 201. Software Defined Networking for Next Generation Converged Metro-Access Networks. M. Ruffini, F. Slyne, C. Bluemm, N. Kitsuwan, S. McGettrick.
9. IEEE ONDM 2016. End-to-end Service Orchestration From Access to Backbone. J. M. Gran Josa, F. Slyne, V. Lopez, M. Ruffini.
10. IEEE ONDM 2016, best student paper award. FLATLANd: A Novel SDN-Based Telecoms Network Architecture Enabling NFV and Metro-Access Convergence. Frank Slyne, Marco Ruffini
11. IEEE/OSA Journal of Lightwave technology, vol. 34, No. 18, September 2016. Experimental End-to-End Demonstration of Shared N:M Dual Homed Protection in SDN-controlled Long Reach PON and Pan-European Core. Seamas McGettrick, Frank Slyne, Nattapong Kitsuwan,David B. Payne, and Marco Ruffini
12. IEEE/OSA Journal of Lightwave technology, in press. SDN Enabled Dynamically Reconfigurable High Capacity Optical Access Architecture for Converged Services. G. Talli, F. Slyne, S. Porto, D. Carey, N. Brandonisio, A. Naughton, P. Ossieur, S. McGettrick, C. Blumm, M. Ruffini, D. Payne, R. Bonk, T. Pfeiffer, N. Parsons, P. Townsend

The following is a list of papers to which I have contributed, which have been submitted for publication.

1. [Invited] IEEE/OSA Journal of Optical Communications and Networking. End-to-end Service Orchestration From Access to Backbone. V. Lopez, J. M. Gran Josa, V. Uceda, F. Slyne, M. Ruffini, R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, R. Martínez
2. [Invited] IEEE/OSA Journal of Optical Communications and Networking. FLATLANd: A Novel SDN-Based Telecoms Network Architecture Enabling NFV and Metro-Access Convergence. Frank Slyne, Marco Ruffini.

Demonstrations

1. Work contained in this document related to Fast Protection (section 5.3) was demonstrated at ECOC 2015 which was held in Valencia 27th – 30th September 2015.

2. Work contained in this document related to Fast Protection (section 5.3) and Dynamic Wavelength Assignment (section 6.1) was demonstrated at the EU-FP7 DISCUS plenary meeting held in the Tyndall Institute, Cork. 8th-10th December 2015.

The following is a list of Invention Disclosures to which I have contributed.

1. Invention Disclosure P11512GB at UK IPO. (The official filing details assigned to this UK Application are 1412069.5.). Metro-Core Network Layer and System.

Chapter 2 State of the Art

2.1 Software Defined Networks

The Open Network Foundation [32] defines Software Defined Network as a network "*architecture [that] decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services*." Traditional telecommunications networks are characterised by very long provisioning times and lack of flexibility in network bandwidth [33]. There are multiple manual configuration actions to set up new services and customers, largely due to the lack of integration between the management systems of the technology stacks that support the service. Legacy network architectures are embedded in the control plane with the data plane in network devices, while Software Defined Networks have the advantages of being "*dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications*." SDN separates control plane routing decisions, user plane forwarding engines and processing of individual flows. SDN enables Virtualisation thereby overcoming issues associated with multilayer and network segmentation thereby optimising infrastructure resource utilisation [34]. The impact of SDN on Telecommunications networks is forecast to have real tangible effects with AT&T implementing SDN in its core Telecommunications Network at 4500 Central Offices (CO's) as part of its Domain 2.0 programme. AT&T predicts a reduction of $95 to $85 per annum in service delivery cost per customer.

The concept of Software Defined Network (SDN) appears in different categories of networks ranging from carrier networks, data centres and central office networks through to home and wireless networks. Also, SDN is relevant to physical, link, network and transport layers of the OSI and TCP/IP stacks, both individually but also in an amalgamation. The impetus behind SDN is Openflow [35] which aims at replacing, or at least extending, current network equipment by a new type of "dumb switches" where the decision making is entirely assumed by Controller(s), giving the switches only a basic set of instructions: (a) Forward the packet, (b) Drop the packet, (c) Send the packet to Controller (after encapsulation) and (d) Overwrite part of the packet header. Openflow switches only need to look at their Flow Table(s) which contains the action(s) associated to a flow. To identify a flow, a switch can rely on a function which can match various fields in the frame (inbound port, VLAN ID, data layer or network address, transport protocol header, etc.). To register to a Controller, an Openflow switch goes through a procedure called a Handshake. During this exchange of messages, the two parties gather information about one another, such as the Data-path ID to uniquely identify the switch, the maximum capacity of the buffer and how many bytes of a packet to send to the Controller in case of an unknown flow. Once the switch is registered, it relies on the Controller to handle the management of the flows. When an inbound packet arrives, the

switch goes through its Flow Table(s) to try and match the different headers of the packet to an action. If one is matched, it carries the corresponding action. If not it sends the packet (or part of it depending on the configuration) to the Controller with a PACKET IN message. The Controller then replies back the final decision about the packet, whether it is to forward it with a PACKET OUT message or drop it entirely. It possibly writes an action in the switch's Flow Table with a FLOW MOD message in case another packet from the same flow comes up.

2.1.1 SDN at Layer 2 and Layer 3

While the Openflow protocols are synonymous with SDN for the configuration and management of flows at the data plane layer, it is one of a number of protocols that abstract the control plane from the data plane of network devices. The concept of the separation of control and data planes had been in existence for a number of years prior to Openflow catching the attention of first the research community, followed by switch manufacturers and software providers.

Rexford, Caesar, Feamster and Caldwell [36] first presented a Routing Control Platform (RCP) in which Border Gateway Protocol (BGP) inter- domain routing is replaced by centralised routing control to reduce complexity of fully distributed path computation. In the same year, IETF released the Forwarding and Control Element Separation (ForCES) framework, which separates control and packet forwarding elements in a ForCES Network. A ForCES Network Element (NE) consists of multiple Forwarding Elements (FEs) and multiple Control Elements (CEs). In 2007, Casado, Freedman, Pettit, Luo, McKeown and Shenker [37] presented Ethane, where simple flow-based Ethernet switches are supplemented with a centralised controller to manage admittance and routing of flows.

The Openflow Switch Consortium released the Openflow reference implementation (version 0.1.0) in 2007. In 2009, Openflow version 1.0 added multiple queues per output port for minimum bandwidth guarantees. In 2011, Openflow version 1.1 added multiple tables pipeline processing, VLAN's and rudimentary support for MPLS. In 2012, after stewardship for Openflow moved to the Open Network Foundation (ONF), Openflow version 1.2 was released and provided support for Flexible Matching and Rewrite, Multiple Controllers and IPv6. Openflow version 1.3 provided support for PBB tunnelling, Per-flow bandwidth tracking, traffic measurement and event filtering. The OF-Config 1.1 protocol was enhanced to allow configuration and management of Openflow switches and controllers. Openflow version 1.4 [38], released in 2014, supports Optical port parameters and Command Bundling. Command Bundling allows group of commands to be committed or rolled back in the event of success or failure of a given criteria. Openflow version 1.5 supports Layer 4 to Layer 7 processing through deep header parsing and execution of complex actions. There is support for a wider variant of Tunnels, as well as the stacking of tunnels. Prior to

version 1.5, flows were treated as unidirectional and stateless. With version 1.5 flows can be made persistent through the use of flow meta-data, as well as being paired as bidirectional flows in upstream and downstream direction.

The ONF has created a number of working groups to advance SDN in different areas. The Forwarding Abstraction Work Group is both standardising Openflow Switch hardware, but also improving interoperability between switches and controllers, through the use of Negotiable Datapath Modes (NDM) and Table Type Patterns (TTP). TTP describes a set of flow tables and the valid operations to be supported by an OF switch. Although the syntax and definition of TTPs is currently being defined, it is supported in rudimentary form in OF-Config v1.2. This allows some negotiation of the supported TTP at switch initialisation. The Optical Transport Working Group is looking at ROADM configuration in photonic enterprise networks and Network virtualisation for multi-layer networks and packet-optical integration. The Wireless and Mobile Working Group is responsible for proposing Openflow support and extensions for wireless transport, Mobile Packet Core and Mobile packet tunnels (for example GTP – GPRS Tunnelling Protocol).

Switch configuration may be performed through provisioning directly to the OVSDB database associated with each switch under the control of a controller, or through the Netconf based OF-CONFIG. OF-Config initiates the control channel, configures bridges, ports, meters and other facilities on a switch and (in version 1.5) negotiates the use of a particular NDM Network Device model, between the controller and the switch. OF-Config provides limited support for tunnels (such as IP-in-GRE and VXLAN). In future, because it is based on Netconf, OF-Config will support Yang based service model definition. Participation in a virtual machine / datacentre orchestrated network is catered for through a north-bound interface to Openstack Neutron. OpenStack Neutron is an SDN networking project focused on delivering networking-as-a-service (NaaS) in virtual compute environments.

2.1.2 SDN at Layer 1 and Layer 2

In 2006, the Path Computation Element (PCE) architecture was presented to compute label switched paths separately from actual packet forwarding in MPLS and GMPLS networks [39]. PCE and PCEP provide a mechanism for calculation and control and re-optimisation of MPLS Traffic Engineering tunnels (MPLS-TE). PCE is applicable to MPLS routers and GMPLS capable optical elements. Path Computation is the process of calculating route through a network that should be taken by an MPLS or GMPLS traffic engineered tunnel of a defined size, delay and jitter in order to meet the requirements of the bandwidth reservation that it is supporting. The path computation element is a computing function within the network that the MPLS Label Edge Route has elected to delegate this calculation to. The PCEP is the protocol that is run between the MPLS Label Edge Router (LER), known

as the Path Computation Client (PCC) and the PCE. This protocol supports the signalling of the path characteristics from the PCC to the PCE. To calculate the path, the PCE utilises the knowledge of the availability in the network based on its view of the Traffic Engineering Database (TED). The TED contains the set of all of the links within the MPLS domain, their characteristics and their available bandwidth. Elements of the PCEP protocol includes establishment of session between PCC and PCE, request for path computation, generation of keep-alive messages and definition of performance targets for resources and constraints. Ordinarily, the PCE is stateless, and plays no active part in the overall management of resources and bandwidth in the network. The IETF has defined a number of enhancements to the PCE architecture that permits a PCE to request that a PCC initiate an Label Switched Path (LSP), as well as an enhancement to ETSI's Resource Admission Control Subsystem (RACS) [40]. This allows application driven reservation of resources in the network and turns the PCE into a component of a fully-fledged bandwidth management implementation. The PCC still remains in control of the LSP and updates requests that violate the local policy held at the PCC may result in the PCE request being rejected. Because PCE has been specified to support both MPLS and GMPLS functions, this capability can be used by applications wishing to optimise the mapping of MPLS bearers to the optical layer. Velasco, Castro, King, Gerstal, Casellas and Lopez [41] demonstrated a PCE based optimisation tool that was used to prevent spectrum fragmentation in optical networks that support variable sized frequency slots. This was achieved by allowing a controller to adjust the allocation of light paths within the optical spectrum to group smaller light paths and free up larger contiguous blocks of spectrum. A stateful PCE facilitates a number of use cases such as Optimisation of network resources across optical and packet, re-optimisation, re-establishment and prioritisation of reservations after disruption, handling on-demand bandwidth requests from a bandwidth management function.

The Interface to Routing System (I2RS) provides access for external systems to the routing and topology information about a Layer 3 network[42]. It is also possible for external systems to modify the routing in the network. As such, it may act as an alternative to Openflow for conventional IP/MPLS carrier Networks. The objectives of I2RS are to be able to read from or write to the Routing Information Base (RIB), the provision of monitoring and control of BGP including policy enforcement, the control of routing in the network for given flows as well as the ability to extract topological information from a network. I2RS provides NetConf and RestConf (a restful version of NetConf) interfaces, over which Yang Service models may be defined [43].

Current MPLS based networks are characterised by thousands of Traffic Engineering LSP's and thousands of labels within the LDP (Label Distribution Protocol) database. Application states are contained within the network with the result that both convergence and recovery during a protection event can be slow. The IETF has drafted a standard for source routing

of traffic based on labels in both an MPLS and IPv6 based network. The scheme is called Segment Routing [44]. Each node in the network advertises labels to identify themselves as IS-IS via the Interior Gateway Protocol (IGP). Instead of the route being determined at each hop in the network, it can be specified as a sequence of labels that are applied at the ingress of the network. The stack of labels applied at the ingress may either be a specific sequence of next hops (to adjacent nodes) or a set of next segments (across multiple nodes). This facilitates a strict route, a loose routing path or a mixture of both. In the case of a loose routing scheme, this is equivalent to the use of Equal Cost Multi-path (ECMP) routing.

Because node adjacency can be used as a service identifier, adjacency labels only have meaning at a given node. This reduces the size of the LDP database and the number of TE LSPs in the network. Chaining of services is facilitated, by directing traffic through a predetermined set of functions (for example, firewalls). The Segment Routing scheme is compatible with existing IP routing infrastructure including IGP, BGP and MPLS control planes. Because RSVP-TE and LDP are not required in the operation of Segment Routing, protection across most if not all topologies is guaranteed to be less than 50 milliseconds. In the SDN paradigm, the Segment Routing scheme acts as a centrally co-ordinated control plane, with the MPLS or the IPv6 network acting as the data plane.

2.1.3 Network Function Virtualisation (NFV) characteristics

ETSI promotes the standardisation for Fibre to the CAB (FTTcab), VDSL2 and G.Fast,. Most recently, ETSI has looked at which traditional components may be virtualised [45]. These components include GPON OLT's, ONU's, DSL DSLAM and Broadband Remote Access Servers (B-RAS) and home gateway devices. ETSI has a number of objectives in promoting NFV. These include optimisation of cost, reduction in the power consumption of remote devices, the relocation of complex functionality that is currently located in the field, to the Head End, and the automation of provisioning of configuration and new services. ETSI have defined a number of uses cases for NFV services. These use cases relate to the provision of virtual CPE (vCPE), Fixed Access Network Function Virtualisation, virtual Provider Edge (vPE) and virtual Basestation (vBS). The Virtual Network Functions (VNF) forwarding graphs use case describes how services may be chained together. Service chaining is also described by the Broadband Forum document SD-326. The Broadband Forum has a number of working groups looking at SDN as part of Broadband (SD-313), Access Networks (WT-358) and as an enabler for Flexible Service Chaining (SD-326) and Network Function Virtualisation (WT-359) [46]. SD-313 is examining deployment scenarios where only some of the network equipment would support SDN functionalities, as well as possibility of supporting SDN capabilities by upgrading software only. EU FP7 project SPARC has successfully demonstrated the synergies between Software Defined

Networking and Network Function Virtualisation through the separation (and subsequent concentration) of forwarding and processing elements found in traditional telecommunications networks.

2.1.4 Frameworks for Software Defined Networks

A number of SDN frameworks have been created. These range from basic standalone controllers such as Floodlight [47], POX and RYU, that manage individual switches through to full architectures that administer entire data centres and telecommunications networks and WAN's. Generally, the standalone controllers are open-source, however, an exception is Onix which is closed source. Onix [48] is notable because it can provide a global architectural view of the switches under its control, and is also seen as an impetus for the ONF ONOS architectural framework.

OpenContrail is a tactical SDN framework, which has been adopted by Juniper as a control framework (Contrail) for its SDN compatible equipment [49]. Architecturally it is composed of four subsystems. vRouters handle network slicing, traffic steering and MPLS or VXLAN based overlay networks. The configuration subsystem manipulates the high-level service data model into a form for consumption by the devices. The Controller component manages and monitors network state. Lastly, the Analytic subsystem collects and collates data about system performance. OpenContrail uses XML based IF-MAP (Interface to Metadata Access Points) for model definition, which in time will be supplanted by YANG [43] based configuration format.

OpenDayLight [50] is an Opensource SDN architectural framework, based on the Cisco Extensible Network Controller (XNC), that is provided in three different guises or editions. Firstly, there is the basic core Controller edition. Secondly, there is the Virtualisation edition for Data Centres, which interworks with Openstack [51] and Virtual Tenant networks (VTN's). Thirdly, there is the Service provider edition with components for SDN in the WAN, LISP service and Defense4All (D4A) for DDOS protection [50]. The Service Provider edition has renderers for IETF's NetConf configuration, BGP and PCEP [40]. The Topology query for the purposes of discovery and host tracking and inventory management are performed through a REST API. SDN models are defined using YANG based MD-SAL (Model-Driven Service Abstraction Layer), where applications are defined as a data model and the API's required to access them can be auto-generated as part of the integration process. The OpenDayLight framework is made robust through the implementation of a distributed data store and a fail-over arrangement for its primary and stand-by controllers.

The Application-Based Network Operations [52] is an SDN framework that is unique in that it does not communicate using native Openflow to the data plane components [53]. Instead the focus of ABNO is MPLS and GMPLS multi-domain networks with PCE as the controlling agent and PCEP as the control protocol. ABNO also has a policy manager, an I2RS

(Interface 2 Routing System) client, a Virtual Network Topology Manager (VNTM) for multi-layer co-ordination and an Application-Layer Traffic Optimisation Server. Southbound communication with components such as Openflow are achieved using a provisioning manager. Statefulness is provided by an LSP-DB and TED database. ABNO has been used in the IDEALIST project [54] to demonstrate the multi-domain and multilayer configuration of commercial equipment (such as ADVA, Juniper nodes and OTN 400 Gbps channels) and the validation of the PCEP extensions to support remote GMPLS LSP set-up.

ONOS [55] is specifically a network operating system for Service providers, driven and supported by the ONF [56], which also maintains the Openflow standards [55]. ONOS is a specific ONF project with resources allocated to it by services providers such as AT&T and NTT, and research entities such as Internet2 and CREATE-NET. The objectives of the ONOS project are to provide a SDN platform with carrier-grade performance and availability. Overall, ONOS attempts to optimise Capex and Opex. The ONOS project has outlined a number of use cases to demonstrate the carrier capability of the system. These are an SDN IP Peering use case, a Network Function Virtualisation as a Service (NFVaaS) use case and a use case demonstrating failover using IETF Segment Routing (Spring Project). The NFVaaS use case demonstrates a virtual OLT (vOLT) solution for GPON. ONOS does not rely solely on Openflow as its SDN control plane technology, as demonstrated in the Segment Routing use case. The PCE [39] use case looks at the issue of over-dimensioning of current Packet Optical cores so as to handle both network outages and peak bursts. Usually Normal utilisation is kept at 30%, meaning a four to five fold underutilisation of capacity. The ONOS PCE application is used to configure, orchestrate and monitor the packet optical core to achieve much higher levels of utilisation without compromising on redundancy. The ONOS architecture and use cases demonstrates that there is accommodation for SDN protocols other than Openflow, particular for the orchestration of lower layers, as well as the co-ordination of multiple domains.

SPARC [57] reviewed three alternatives to implementing its Split Architecture, IETF's ForCES framework [58], IETF's GMPLS/PCE and Openflow supported at the time by Stanford University, but since then supported by the Open Network Foundation. AT&T are one of the sponsor operators of the ONF's ONOS SDN framework.

GMPLS was discounted by the SPARC project because it is, in essence, an intra-control plane signalling protocol, used for NNI (Network Network Interface) applications. GMPLS does not specify the interaction between the data and the control planes. While PCE recognises the decoupling of the data and control planes, the majority of control plane functions are delegated to distinct network elements. The PCE architecture does facilitate the concentration of control capability in a centralised system, however, with the PCEP protocol running between the Path Computation Element (PCE) and the Path Computation Client (PCC) [59] respectively.

ForCES and Openflow were directly compared because they clearly defined the control interface between the control and data planes. The strength of ForCES was that it was already, by the time Openflow was being created, a mature framework that allowed different technologies to be specified through the use of libraries. While Openflow was seen by SPARC as being less flexible than ForCES, the overall architecture for Openflow was simpler and provided a clearly defined nodal model. Openflow had more support than ForCES from both industry and research communities so it was less likely to be dominated by vendors or by theoretical academic interests. Openflow was selected by SPARC as the basis for its Split Architecture.

2.1.5 SDN in Access Networks

EU FP7 project SPARC (Split Architecture) [60] was an early project to demonstrate both SDN in the Access and Aggregation network as well as a prototype of Network Function Virtualisation, through a Virtual Home Gateway [61] and a Virtual BRAS. [45] There are two (so-called) splits in the SPARC architecture. Firstly, there is the *split* between the Control and Data planes that allows the data and control planes to evolve separately from each other. The data plane extends reach, connectivity and bandwidth, while the control plane enhances service creation, control and delivery. Secondly, there is the *split* between the forwarding and processing elements. In a traditional telco network, these functions are distributed throughout the network, for example at DSLAMs and customer home gateways with the result that these functions become isolated and degraded, though lack of manageability and enhancement. The split in forwarding and processing elements, is familiar in the concept Network Function Virtualisation, where simplified forwarding components at the level of data plane are located in the field or remotely, with the processing elements concentrated in either data centre or central office environments. SPARC respects the separation between access/aggregation and backbone/core networks, and leverages standard IP/MPLS control protocols such as OSPF, LDP, RSVP-TE and BGP to provide the necessary glue between control domains.

Another EU-FP7 project OFELIA, though while not primarily looking at SDN in the access network, demonstrated the evolving use of SDN and particularly Openflow in the Wide Area Networks and the effect on traditional carrier networks [62]. OFELIA demonstrated Optical Wavelength switching, Optical Flow Switching [63] and Multi-service technology control. Associated projects such as EU-FP7 project ALIEN presented a generic model using abstraction in the data plane to allow a wide range of access devices based on FPGA's and Network Processors to be controlled using Openflow.

An access network controller is associated with every metro/core node, where it controls the optical switch, access switch, and OLTs/ONUs. The access controller sends abstracted topological information about the resources available within its domain. Where access

protection is required, the controller handles incoming failure messages from the OLT to operate fast protection. Moreover, the access controller receives provisioning requests from the orchestrator and reports the service setup status. Finally, the access controller should be able to carry out path computation, because the network orchestrator can request a path computation. To do so, the physical domain information have to be obtained from the network elements and mapped in the abstracted view. Similarly, the provisioning of abstracted services is map in real configurations. Therefore, the access controller maintains information on bandwidth availability within the access switch and each OLT. Besides, it configures the network elements (access switch, optical switch, OLT, ONU/ONT) depending on its specific requirements.

2.2 Network Performance and Quality of Service

2.2.1 Causes of Poor Performance

TCP performance

TCP assumes that packet loss is caused by network congestion, and not by transmission errors. In the earlier variants of TCP, congestion was signalled by dropping packets [64]. TCP also assumes that there is only a small amount of jitter so Round Trip Time (RTT) is relatively constant. Any path alteration due to rerouting or switching needs to happen very quickly.

In most recent versions of TCP, a host can transmit a sequence of packets called a window. A new packet cannot be sent until a slot in the current window is available. Each TCP packet that is send has an associated count-down timer. If by the time the time expires an acknowledgment is not received, the sending host assumes that either the packet or its acknowledgment have been lost or corrupted. The sending host retransmits the packet. The purpose of Flow Control is to prevent flooding of a receiver's buffers. A sliding window is the mechanism which is used. A sender can send more data than the window advertised by the receiver, until the window is updated. A persist timer prevents TCP deadlock if the window is not updated by the receiver. The TCP sender will recover from a potential deadlock situation, when the persistence timer expires, by sending a small packet to the receiver so that the receiver can respond by sending an acknowledgement containing the new window size. Flow Control is similar in operation but contrasting in objectives to Congestion Control which prevents a transmitter from pushing too much data on to a network. With Congestion Control, a senders infers information about network conditions from the acknowledgements or lack of them between the sender and receiver.

TCP transmit and congestion windows increase with the increase in latency along the TCP path. With large windows, TCP can transmit a lot data with outstanding acknowledgments.

If the TCP transmit window is 2 MB, TCP will push 2 MB of data at the full network interface speed out to the network, so every device along the path experiences a high-speed burst of packets. Any issues related to buffering or packet loss causes TCP to back-off. This causes quite a marked decrease in throughput and performance. However, in low latency networks, TCP windows are small in comparison. Any similar issues related to queuing and packet loss are identified quickly, but also because the windows are smaller, the effect of TCP recovery is negligible. For a high latency network with inadequate buffers, there is a high chance of a buffer saturation due to large bursts of data. The TCP connection goes through a continuous cycle of recovery and congestion avoidance, or worse, may be in a continuous state of recovery with sub-optimal windowing and transmission rate. However, for low latency networks, while there might be frequent packet loss due to buffer saturation, identification and recovery is much quicker, so the effect is more negligible.

Buffer Congestion

A network node such as an Internet router or switch typically maintains a set of queues, generally one per interface, that hold packets that are scheduled to go out on that interface. The original queuing discipline was the Drop-tail discipline which enqueues the packet if the queue is shorter than its maximum size (measured in packets or in bytes). Otherwise it is dropped. A router buffers as many packets in a fixed length buffer. Any excess packets are dropped. Whenever the network is congested then router buffers are constantly full. The Drop Tail algorithm has a number of disadvantages. Firstly, some TCP flows, such as bursty traffic that use only a small portion of the bandwidth, may hog buffer space. Secondly, the Drop-tail algorithm on similar types of routers across the Internet can lead to TCP global synchronization where all TCP connections in a network are held back and then released through timeout, leading to the anomaly of the Thundering Herd. .

Random Early Detection (RED) is a congestion avoidance algorithm as well as an active queue management algorithm which attempts to overcome global synchronisation by dropping packets based on statistical probabilities[65]. Active queue management (AQM) drops packets with a probability proportional to how full the queue is. Even if the buffer is partially full, packets may be dropped, albeit with a small probability. As the buffer fills up, the probability of enforced packet drop also increases, however there is no fixed threshold at which packets are dropped. The more traffic a host transmits, the more likely it is that its packets are dropped, as the probability of a host's packet being dropped is proportional to the amount of data it has in a queue.

Explicit Congestion Notification (ECN) is a IP extension that is an alternative to dropping packets as a means of detecting congestion [66]. For it to work, it requires the co-operation of routers along the IP path as well as the terminating points. When an ECN-aware router detects impending congestion at its interfaces, it sets a flag in the header of transiting IP

packets instead of dropping a. The receiver of the packet echoes the ECN flag back to the sender, which should adjust it's transmit rate downwards.

Bufferbloat

There are many locations in a TCP data path, where traffic may be buffered. These include network devices such as core and edge router nodes, broadband Remote Access Servers, customer premises equipment such as broadband routers and laptop network stacks as well as hosts within data centres. Buffers are judiciously placed at ingress ports to help absorb (without packet drops) any transient bursts of bandwidth that may occur on the traffic links.

Logically, a buffer should be equal to the TCP congestion window which will vary with the Round Trip Time (RTT) of a TCP connection [67]. Typical RTT between sites within the same region is 20 milliseconds, between sites on the same continent is 100 milliseconds and between different continents 200 milliseconds [68]. The guideline for the network equipment manufacturers is to provide buffers large enough to accommodate at least 250 milliseconds worth of traffic passing through a device [69]. For example, the 1 Gbps Ethernet interface on a router would require a buffer of 32 MB in size. If buffers are not adequately large then TCP sessions with long Round Trip Time can experience excessive packet loss and TCP bandwidth reduction [70].

The TCP congestion avoidance algorithms rely on either packet Round Trip Times or packet drops to set the congestion window and the data throughput for a TCP connection. Where packets are buffered rather than dropped, the congestion algorithms do not alter their congestion windows appropriately. As a consequence packets which have been subjected to long or variable buffering may arrive with either high latency or jitter.[71]. The problem of continuously filled buffers which do not dissipate normally and function in a manner counter to their original purpose, that is, to improve Quality of Service, is called BufferBloat [22]. Since it is quite common in the downstream network path for network elements to high-bandwidth ingress links and low-bandwidth egress link, Bufferbloat problem is exacerbated by traffic bursts on the high-bandwidth ingress links that can fill up the buffers without giving them a chance to be drained by the low-bandwidth egress links [70]. For example, a buffer which is 1Mbyte in size takes 2 seconds (1000Kbytes/(8bits per Byte / 4Mbps) to empty through a 4 Mbps pipe.

This buffering effect affects UDP (that is non-TCP) applications as well, since application which require different mixes of latency and bandwidth all share the same traffic links. The effect is that Mpeg compressed video can suffer missed frame synchronisation, DNS resolver requests may time out. Gettys, the original proponent of the BufferBloat concept criticised equipment for including unnecessarily large buffers due to the availability of inexpensive high density Dynamic RAM (DRAM) [22].

2.2.2 Remediating Bufferbloat

Classical AQM algorithms based on RED try to identify Bufferbloat by gauging how full buffers become. There are two problems with approaches based upon RED based AQM algorithms [72] . Firstly, buffers may fill up for legitimate reasons other than through Bufferbloat. Buffers may fill up due to short spurts of high volume traffic and then dissipate normally. These are called Good Buffers. Secondly, such algorithms do not facilitate remedial actions for TCP traffic streams buffered for long periods, as they do not discriminate based on the age of data in the buffer.

CableLabs evaluated a number of solutions that remediate the Bufferbloat issue[73]. They looked at two Saturated Tail- Dropping approaches (Saturated Tail-Dropping Queues with large buffer depths and Saturated Tail-Dropping Queues with short buffer depths optimised with Buffer Control ECN, feature set to depths equal to the expected Bandwidth-Delay Product) [74]. Both of these algorithms perform simple dropping of packets whenever queues reach their maximum size, but they do not respond quickly to queue build-ups nor can they be forced to drop a sufficient number of packets once queue saturation is reached. As a result, Saturated Tail-Dropping systems cause BufferBloat for some latency-sensitive packets. The SFQ-CoDel establishes different queues per service group packet flows, which are identified as hash codes calculated from flow tuples. These hash codes direct different packet flows into different queues, which are serviced in a round-robin fashion. While performance is good, but performance suffers if there are hash collisions that causes two separate flows into a single queue. It is for this reason, SFQ has been enhanced with the CoDel algorithm which drops packets when performance may have degraded because of the issue of hash-collisions.

Both CoDel [71] and PIE [75] try to pre-empt buffer saturation by either dropping packets or throttling high-bandwidth flows. They do this well in advance of the Saturated Tail algorithms. Similar to RED, PIE randomly drops a packet at the onset of the congestion, however, congestion detection is based on the queueing latency (similar to CoDel) unlike the queue length in conventional AQM schemes such as RED. PIE uses a combination of latency moving trends and whether latency is increasing or decreasing to determine the true levels congestion. The CoDel (Controlled Delay) scheduling algorithm determines if a queue is good disciplined or bad based on the minimum age of packets in the queue. A good queue is where the minimum age of a packet is less than 5 milliseconds. For this algorithm to work, the timestamp of when the packet entered the queue must also be stored. When a packet is dequeued with an age greater than 5 milliseconds for a given window, the algorithm drops the packet. CoDel can infer buffer depths from the measured packet delays. The advantages of the CoDel algorithm are that the monitoring and the action of the algorithm require little processing overhead and require no additional configuration parameters. No action is taken against packets within a Good disciplined queue. The

disadvantage of CoDel is that it requires changes to data structures within the queuing mechanisms of host and routing devices. The CoDel queuing discipline has been available from Linux version 3.5 (2012).

The BDP (Bandwidth-Delay Product) is defined as the maximum amount of data that has been transmitted but not yet acknowledged on network connection at any point in time. It is calculated as a product of a data link's capacity measure in bits per second and its RTT (round-trip delay time measured in seconds). A network with a large BDP value is called a Long Fat Network. On a homogeneous network, the BDP would be equivalent to the product of the transmission speed of the egress port on the network element multiplied by the RTT currently being experienced by the TCP connection (with units of Bytes). Each elements assumes that its own egress link bandwidth capacity is the highest transmission rate that the TCP session will experience, and sets its buffer depth accordingly. However, this is an incorrect assumption since links, port speeds and Round Trip Times are not homogeneous along the path of the TCP connection. With one buffer size being defined for a single shared buffer which caters for multiple flows, the buffer is typically not apportioned based on the RTT of each flow [22].

2.3 QoS Frameworks

Besides specific events such as buffer saturation and Bufferbloat, network termination devices require the ability to request and be given a particular Quality of Service. Examples of Quality of Service parameters are that jitter, delay or packet loss are within give bounds. QoS frameworks are typically categorised by how they deal with this complexity and scalability as well as service differentiation. IntServ is the model architecture for IP based QoS guarantees. IntServ (RFC 1633)[76] configure every router in a small network run by a single operator, where end users traffic patterns are predetermined [76]. The Resource Reservation Protocol (RSVP) adapts the IntServ model for dynamic QoS provision of real-time/interactive traffic over larger and more complex networks [77].The RSVP protocol uses signalling messages along the network path between sender and receiver, with each node along the path interpreting RSVP and storing QoS state information for each flow requesting resources. End hosts (or their proxies) should also interpret the RSVP protocol. IntServ (Integrated Services) provides three levels of Class of Service, which are Guaranteed Service, Controlled-load Service and Best Effort. The down side of IntServ is that it is complicated and resource intensive [76].

In contrast to IntServ which deals with single flow instances, DiffServ [78] reduces the volume of the required flow state information in routers by dealing with flow aggregates [79]. Each edge device must set the appropriate DSCP bits based on the network’s current QoS policy. DiffServ enabled nodes are required to inspect the DSCP and respect the required QoS appropriate for that particular class of service. Exterior nodes of a DiffServ domain

implement may admission control blocking. Interior nodes do not track individual flows but must be provisioned to handle the actual classes of service which are provided inside the domain. Overall, DiffServ assurances are statistical in nature so there is not an explicit alignment between the QoS requirement requested by an end application and the QoS delivered by the network. This makes DiffServ appropriate to networks with larger cores compared to IntServ. DiffServ (Differentiated Services) Blake, Black, Carlson, Davies, Wang and Weiss [79] provides three levels of service also, Expedited Forwarding, Assured Forwarding, Default Forwarding. Expediting Forwarding is employed where there is a need for low loss, low latency, low jitter, and assured bandwidth end-to-end services. IntServ treats different classes of packets in a different manner.

In contrast to IntServ and DiffServ which are still criticised for remaining dumb and increasing protocol management and overheads and make QoS decisions based on the IP packet header, MPLS makes QoS (and routing) decisions, based on short fixed length (shim) label in the packet header [4] [80]. Where the label matches an entry in a routers forwarding table, the packet may be forward along an explicit Label Switched Paths (LSPs) [81]. An LSP may support a class of service or aggregate particular network resources. MPLS configures an end-to-end path between routers and simplifies QoS classification and management [82] MPLS Switching based on Label forwarding enables a higher packet processing rate because the forwarding component of the router is simpler. However, every node along a network path must know what MPLS labels map to a particular class of service. This is similar to every node in a DiffServ network being aware of the mapping between DSCP bits and the Class of service.

The NGN Flow-State-Aware Transport mechanism uses DiffServ flow-aggregation in the Core and QoS mechanisms at the edge that are based on individual flows. While the Flow-state-aware transport technology is relatively similar to IntServ, it uses flow aggregations and is thus more scalable and less complex. NGN typically separates services from the underlying separating transport layer, so when a transport link carries QoS guaranteed traffic, an FSA node needs to guarantee a certain part of the link capacity for the flow-state-aware traffic [83]. Flow-Aggregate-Based Services enhances NGN Flow-State-Aware architecture and addresses three distinct types of congestion - instantaneous (packet-level) congestion, sustainable (flow-level) congestion and congestion avoidance. Instantaneous congestion is mitigated through the proper aggregation of flows and discard of packets. Sustainable congestion is resolved through rate limiting, and admission flow discards. Flow-aggregate-based services introduces inter-domain flow aggregation and endpoint implicit admission control. DiffProbe is used to estimate congestion in the network.

Overall, QoS frameworks may be distinguished by whether they require signalling or not. Both Connectionless approach and FAN do not require any signalling and do not offer much service differentiation. IntServ, Flow-State Aware and Flow-Aggregated-Based Services

outline in depth how to use signalling and offer greater differentiation, with numerous parameters to be assigned to each flow and multiple classes of service. Signalling is relatively complex limiting the scalability.

2.4 Flow-based QoS Frameworks

Almost all flow-based QoS architectures understand the concept of a traffic flow, either as the object or component of an object subject to defined quality metrics, such as a stream of related packets from a single-user activity such as a single video stream or voice transmission.

IETF defines a flow as a unidirectional sequence of packets with some common properties that pass through a network device, with flow classifiers based on the 5-tuple of the source and destination addresses, ports, and the transport protocol (either TCP or UDP) used for transmission.

In the IntServ QoS framework, traffic related to a single service is classed as a flow. Routers in the traffic path must treat all packets within the flow equally with the same QoS. In the Connectionless QoS framework, a flow is defined as a stream of packets between two client server applications. A user may create multiple flow instances in the network which must be treated individually. In both the DPS and Feedback and Distribution QoS frameworks, there is a clear distinction between UDP and TCP flows, with a single user session defined by the standard 5-tuple. In flow-based differentiated QoS frameworks, a flow is regarded as an aggregate of all transmissions between the same end users, defined by unique pair of source and destination IP addresses that belong to the same class of service defined by a value of the DiffServ field (DS field) [84]. Where NAT obfuscates multiple sessions within a single flow, the source destination pair would be identical. Similarly, NGN-based flow QoS frameworks, such as Flow State Aware transport and Flow-Aggregated-Based Services define a flow based on a unique pair of source and destination IP addresses that belong to the same class of service defined by a value of the DiffServ field or MPLS field. The Connectionless Approach overcomes the scalability issues of the IntServ model, by using an Automatic Quality of Service mechanism instead of using the RSVP protocol [85]. The AQS mechanism profiles the network traffic in real-time and defines the end-to-end QoS along the path of the traffic. The approach is scalable since it does not use signalling between nodes. However, in order to manage router bandwidth it retains the IntServ Model classifier, admission controller and scheduler. As a result, Traffic handling capability is reduced because of the complex processing performed at each node. Because the QoS logic is executed autonomously at each individual node, it is not possible for end users to differentiate their bandwidth in advance, through service upgrade nor downgrade. Connectionless approach is also open to abuse by users that try to imitate other traffic types.

Dynamic Packet State (DPS) adopts the IntServ QoS admission control and scheduling and obviates the need for per-flow states in core routers [86]. The edge router inserts per-flow QoS classification into the IP packet header which can then be read and updated by all (including the core) routers in the path of the traffic. DPS approach is scalable in the core of the network, but has a number of limitations. The IP header is being modified according to the CJVC (Core-Jitter Virtual Clock queueing algorithm) and requires proprietary router firmware. The manipulation of the IP header makes the real-time data handling more complex. Because the router firmware for the modified IP functionality must be pervasive throughout the network, the architecture cannot be introduced gradually into the network, but done in so as part of a step change.

The Feedback and Distribution Method is a hybrid QoS framework similar to DPS, but designed specifically for a client server network, with traffic being generated predominantly from the server side [87]. It has the per-flow traffic regulation of IntServ and the simplified core architecture of DiffServ and does per-flow-based QoS differentiation, by marking the traffic and the server side and profiling the traffic and the receiving client side. The traffic marker assigns one of two levels of priority, either high or low, to a flow. A profile meter gauges if a flow is received with the required priority. When a high priority flow starts to experience congestion, the profile meter feeds a signal back to the traffic marker to drop packets related to low priority flows. This lasts until such time as the quality of service related to the high priority flows is re-established. Flow-Based Differentiated Services implement a flow based proportional QoS scheme based on three additional modules: a flow estimator for the number of active flows; a dynamic weighted fair queuing scheduler and a queue manager [78]. While Flow-based DiffServ has the advantage of retaining the scalability features of the basic DiffServ, it also retains the disadvantages of a limited number of Class of Service (CoS) and the difficulties in maintaining CoS service across domains.

Flow-Aware Networking provides differentiation based on the current flow peak rate while protecting low-rate flows[88]. Admission Control maintains the quality of existing flows while restricting new flows (of all priorities) until network congestion has improved. The functionalities for measurement-based admission control and fair scheduling with priorities that control link sharing and other traffic management mechanisms are implemented in a custom router called a Cross-Protect (XP) Router [89]. The XP Router does not require signalling between routers, the QoS calculation algorithms are lightweight so there is low processing overhead [90].

2.5 How Network Performance is Benchmarked

Many operators provision their network, and thus prioritise capital investments, based on either peak period traffic volume or a measure of the per-subscriber bandwidth at peak (or peak hour, or peak period) [91]. A number of operators use additional metrics to optimise

where and when capital investments should be made [92]. Two metrics that suit this objective are round-trip time (a measure of network latency) and video quality of experience. Video quality can test all dimensions of the quality of delivery of service in terms of Display Quality (how good the picture looks e.g. the target bitrate and resolution) and Transport Quality (how often the picture stalls and rebuffers). However, not all Internet video behaves in the same manner. Progressive video takes the user's request for a particular level of quality and starts downloading the file. In a progressive download, the video usually does not start playing until the buffer has grown large enough to ensure stall-free playback. Adaptive video takes a different approach, achieving transport quality at the expense of the display quality (to the viewer, this manifests as down-shifts and up-shifts in display quality). The effect of poor network performance on businesses can be quite stark. Quantitatively, Amazon has estimated that each 100 milliseconds of network latency between its customers and its services costs them 1% in sales annually [93].

There are two main standards for benchmarking throughput for internet (packet) based devices: IETF RFC2544 and ITU-T Y.1564. RFC2544 is the base standards for determining Throughput, Latency, Frame Loss and Back-to-back frames performing tests, on Devices Under Test (DUT), for a range of standard frame size (64, 128, 256, 512, 1024, 1280 and 1518 bytes). Back-to-back frame testing involves sending a burst of frames with minimum inter-frame gaps to the DUT and count the number of frames forwarded by the DUT. If the count of transmitted frames is equal to the number of frames forwarded the length of the burst is increased and the test is rerun. If the number of forwarded frames is less than the number transmitted, the length of the burst is reduced and the test is rerun. The back-to-back value is the number of frames in the longest burst that the DUT will handle without the loss of any frames. Some of the criticisms of RFC6815 are that its main purpose is to benchmark network equipment not to turn up services, it can't be used for determination of QoS characteristics such as Committed Information Rate (CIR) and it does not measure Inter-frame delay variation (IFDV) commonly known as Jitter. The more recent ITU-T Y.1564 (EtherSAM) standard was created within the context of Ethernet service activation based on the service attributes used by service providers to define their SLAs. EtherSAM is comprised of two phases, the service configuration test and the service performance test. The service configuration test consists in sequentially testing each service. It validates that the service is properly provisioned and that all SLA parameters (throughput, frame delay, frame loss, frame delay variation) are met. A ramp test and a burst test are performed to verify the committed information rate (CIR), excess information rate (EIR), committed burst size (CBS) and excess burst size (EBS). Once the configuration of each service is validated, the service performance test simultaneously validates the quality of all the services over time. In this phase, all services are generated at once at their CIR, and all KPIs are measured for each service.

2.6 Data Plane Design

Figure 6 shows the logical architecture of the state of the art Data Plane that spans the network between two end points which may be a Data Centre Traffic source and an end user.

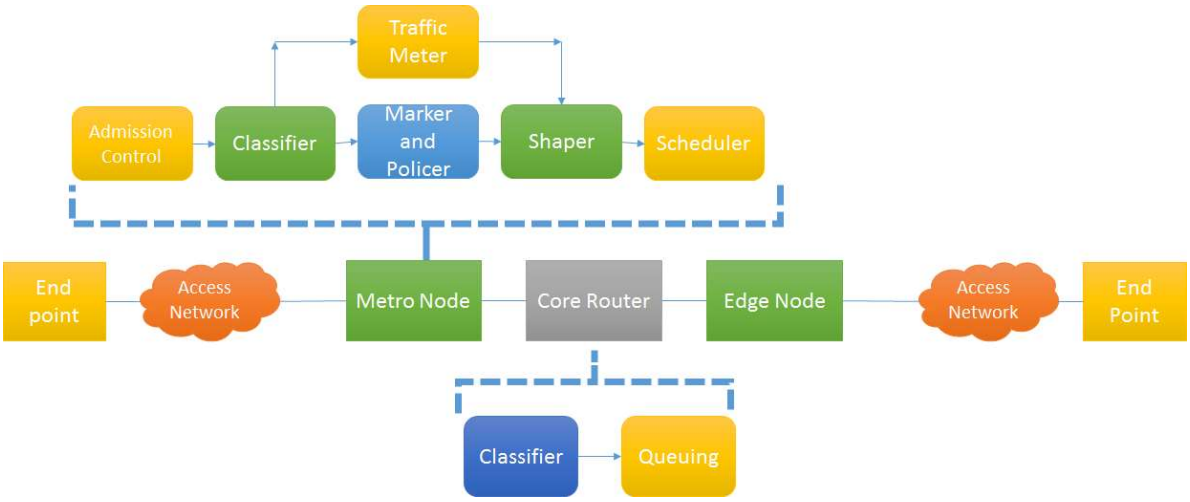


Figure 6 - Multi-layer Traffic Conditioning

Traffic management parameters and rules that are applied to user generated traffic streams are defined by a TCA (Traffic Conditioning Agreement). The TCA describes the various networking mechanisms required in order to handle packets according to a required QoS (Quality of Service). TCA is usually subject to an underlying Service Level Agreement (SLA) provided by the Network Layer [79].The primary mechanisms used by QoS include traffic classification, call admission, regulation, policing and shaping. Secondary mechanisms include signalling, routing and flow control. These mechanisms are invoked in the provision of a typical CoS (Class of Service) scenarios. In particular, admission control, traffic policing and shaping, packet scheduling, and buffer management, are used, and are coupled with flow and congestion control and routing [94]. Admission Control is the function that allows connection to the network. Packet queuing involves the buffering, queuing and servicing of packets throughout the buffers along the length of the network. Depending on the appropriate servicing discipline or policy, Queued packets may be dropped or scheduled. In a multi-layer network composed of metro, access, edge and core layers, it is appropriate to apply QoS (and traffic management) functions at particular layers.

Admission Control, traffic policing, shaping and sometimes buffer management are found in the access or edge layers. Scheduling, buffer management and sometimes shaping and regulation can be found in the metro or the core networks. Packet flow handling is found throughout the various layers.

2.6.1 **Traffic Conditioning**

Class of Service, Type of Service, QoS and Traffic Management are used to balance utilisation of constrained processing resources with meeting the demands of concurrent differing service streams, usually in a packetised environment. Generally, they interwork with each other, but are invoked under different conditions.

Traffic Management provides congestion management, queuing algorithms, prioritisation and merging of network traffic for large numbers of flows [95]. It forwards traffic according to a user-defined set of rules pertaining to priority levels, latency and bandwidth guarantees, and varying congestion levels. Traffic Management prevents network congestion using the techniques of traffic measurement, policing and shaping. At a granular level the mechanisms such as Transmit priority, bandwidth allocation, Call Admission Control (CAC), congestion avoidance, and selective packet loss are employed. Traffic management can sometimes be called Traffic Conditioning or Traffic Access Control [79]. On the ingress line card, WFQ (Weighted Fair Queuing) allows packets from lower priority queues to be interleaved with higher priority traffic into the switch fabric. This prevents the higher priority traffic from completely blocking the lower priority traffic, since the queues are guaranteed access to the switch fabric for a predefined proportion of the time.

Traffic policing prevent either inadvertent or deliberate traffic surges which overload network end-points and intermediate network elements. It does this by analysing and measuring traffic characteristics in real-time. There are a number of responses that are possible should the requisite policy limits be breached. The traffic may be tagged and routed separately from other traffic or the traffic may be dropped in extreme circumstances.

Traffic engineering avoids or reduces congestion by controlling traffic paths in a network and routing traffic along non-default routes in a congested network. This has the benefit of optimising network resources such as link bandwidth utilisation in and out of the Metro Node. In order to do this, Traffic Engineering need to be capable of measuring the capacity of possible flows or maximizing the flows in a given network [96].

Admission control is a type of traffic policing that prevents traffic with a particular characteristics, to regions of the network. Two end-points of a transport link agree flow control parameters that ensure that both stations are not over-burdened by traffic, particularly to the extent that packets are dropped. Traffic shaping by an intermediate component such as a Network Processor supports desired flow-control traffic characteristic such as desired rate and burstiness. It does this by regulating the volume of packets released into the network using a combination of buffering, metering and smoothing. Depending on a given shaping or scheduling algorithm, packets may be forwarded to appropriate queues, and then scheduled for transmission according to the conditions of the lines, the receivers, and the priorities appropriate to that these packets [97].

2.6.2 Network Node Structure

In a network node such as an Internet router or switch, traffic needs to transit the node with the minimum of delay and interference as possible. Flow processing is the function where the characteristics of flows of very many packets are manipulated over time. These characteristics include QoS (Quality of Service) and CoS (Class of Service), which align to a cut-through switching architecture. Cut-through switching is a method for packet switching systems, wherein the switch starts forwarding a frame (or packet) before the whole frame has been received, normally as soon as the destination address is processed. It is only in rare circumstances that entire individual packets are processed (for example, where they rewritten or compressed) and require store and forwarding.

Modern switch architectures employ Virtual Output Queuing (VOQ). VOQ, in conjunction with a scheduling algorithm to eliminates blocking issues, such as Head-of-Line blocking, input blocking, and output blocking [98]. HOL blocking wastes nearly 50% of crossbar switch's bandwidth if the cells waiting at each input are stored in a single First-In, First-Out (FIFO) queue. To implement scheduling algorithms requires signalling and a switch fabric runs faster than the line speed of the interfaces. A 10GE line card that supports 15 Gbps to the switch fabric offers 50 percent “speedup”. Where the crossbar switch runs twice as fast as the external line, the traffic manager can transfer two cells from each input port, and two cells to each output port during each cell time.

The functionality that needs to be implemented at a Network Node includes Admission Control, Classification, Marking/Policing, Shaping and Scheduling. All incoming packets needs to be Error Checked and (re-)assembled. Rudimentary address lookup must be performed and Traffic management polices applied. For outgoing packets, checksums must be calculated and traffic shaped prioritised and queued.

In order to perform switching, routing and access control validation, the packet needs to be classified by performing a match against classification tables in its local control plane processor. The classification table is a special memory used by the packet processor, and contains routing table determines where to route incoming packets, Access Control Lists (ACLs) which grant or deny permission to specific users or groups and flow classification table about a particular user or group of users, protocols, and applications. A Simple Layer 2 or Layer 3 switch classifies traffic based on the Layer 2 Header (such as the Ethernet header and the VLAN tags) or the Layer 3 Header (which may be either IPv4 or IPv6). Older architectures would use sequential look-up trees to perform the match, while newer architectures store the tables in TCAM (Ternary CAM) which allows matches to be executed in a few clock cycles.

2.6.3 Architectural Constraints

There are a number of situations in the transit of a packet through a Router or Switch where it may suffer processing constraints, most notably in the Ingress packet buffer, during packet classification, Crossbar switch and backplane interconnect, Traffic Management, Multicast replication and queues and during interaction between the Route processing and the Control plane [99]. Table 1 shows the loading effect on the central process by a number of typical layer 2 and layer 3 traffic processing scenarios.

Scenario	Activity	Action	Level of Loading
A	Full Duplex layer 2 performance and latency	Packet Forwarding	Very Low
B	Layer 2 QoS throughput & latency test	Traffic Management	Medium
C	Layer 3 (IPv4) with ACL performance & latency test	Ingress Packet Buffer Packet Classification	Medium
D	Layer 3 (IPv4) with QoS & ACL performance & latency test	Ingress Packet Buffer Packet Classification Traffic Management	Medium
E	Mesh L3 IPv4 with ACL performance & latency	Packet Classification CrossBar	Medium
F	Mesh L3 IPv4 with QoS and ACL performance & latency	Packet Classification Traffic Management CrossBar	

Table 1 - Loading Scenarios

Scenario A demonstrates full duplex with traffic transmitting in both directions. The DUT (Device under Test) must perform packet parsing and Layer 2 address look-ups on the ingress port and then modify the header before forwarding the packet on the egress port. This scenario does not present loading on any components of the router or switch. Scenario B determines the DUT's maximum Layer 2 forwarding rate with packet loss and latency for different packet sizes. The DUT must perform a Layer 2 address lookup, check the 802.1p priority bit value on the ingress port, send it to the designated queue, and then modify the header before forwarding the packet on the egress port. This scenario presents medium loading on the Traffic Management module. Scenario C determines the DUT's maximum IPv4 Layer 3 forwarding rate with packet loss and latency for different packet sizes. The DUT must perform packet parsing and route look-ups for both Layer 2 and Layer 3 packets on the ingress port and then modify the header before forwarding the packet on the egress

port. The ACL test involves blocking or allowing traffic through, based on user-defined classifiers such as IP addresses or Layer 4 port numbers. This scenario presents Medium loading on the Ingress Packet Buffer and Packet Classification modules. In Scenario D, QoS values in each header will force the classification of the traffic based on IP Type of Service (TOS) field settings. On the ingress side, this QoS policy could also be used for assigning a packet to a specific queue, packet metering, and policing; on the egress side, it could be used for packet shaping. This scenario presents Medium loading on the Ingress Packet Buffer, Packet Classification and Traffic Management modules.

It is apparent that every packet that crosses a router's interface must be read at Layer 3 and a new MAC header must be created. Reading a packet's Layer 3 addressing information and creating a new MAC header causes latency. In contrast, when a packet is switched through a network, the Layer 2 address is read and the packet is forwarded, filtered, or flooded. The MAC header is not recreated and this dramatically reduces latency. To emphasise the scale of computing that a processor would have to complete, Giladi [96] gives an example of 1 Gbps Ethernet handling about 1 million packets per second. For each packet, classification based on complex parsing would be executed. Typically, this might involve retrieving both the destination port and its IP address. In some cases, for some destination ports this could mean identifying field in layer 7 Protocol Data Unit (PDU), at an offset depending on the destination port. On from which, there could be one or two searches to be executed to retrieve a destination IP address and port. A search would be conducted among hundreds of thousands of possible addresses, and a longest prefix match is matched. Ignoring packet modification and forwarding processing times, all these parsing and searching activities would have to take place in less than 1 microsecond. The Von Neumann architecture does not efficiently support this set of sequential and parallel miniature processing steps and data flows.

Jitter, packet loss and latency can be caused by a build-up of data in the ingress packet buffer. The packet buffer is a temporary repository for arriving packets while they wait to be processed by Packet processing function. This may be caused by sub-optimal efficiency and architecture of the packet processor, or multiple ingress ports on a switch/router contending for an egress port. To prevent buffer overflow, the packet processor issues a flow-control instruction to the upstream MAC device, instructing it to stop passing packets, which then transmits a "pause" to remote ports requesting them to suspending sending packets. Where the buffer continues to fill-up, the MAC device will start to drop packets. An ingress packet buffer being fed by a 10GE MAC device continuous packet stream would have to dequeue packets every 67 ns in order not to saturate. During times of congestion, the traffic manager may need to make discard decisions based on the availability of queue space, priority, or destination port, using a packet discard algorithm like Random Early Detection (RED) or Weighted RED (WRED) for IP traffic. The worst case performance for

small (64 Byte) Ethernet packets through a 10G/E interface are 14.88 Million IPV4 Packets/Sec, and 12.25 Million MPLS IPv4 Packets/Sec.

2.7 Tree Networks

We see examples of research into large scale layer 2 networks in the Telecoms world as well as in Data Centres. In Figure 7, EU FP-7 Project *SPARC* uses MPLS labels to bind service provider and customer groupings together [60]. In Project *NANDO* (Neutral Access), network slicing is created using VLANs [31]. This provides the benefit of simplicity from the perspective of encapsulation and working across different media, however it has a significant downside. Inclusion in the VLAN is based on a service provider-supplied secondary MAC address that must be associated with each device that requires access.

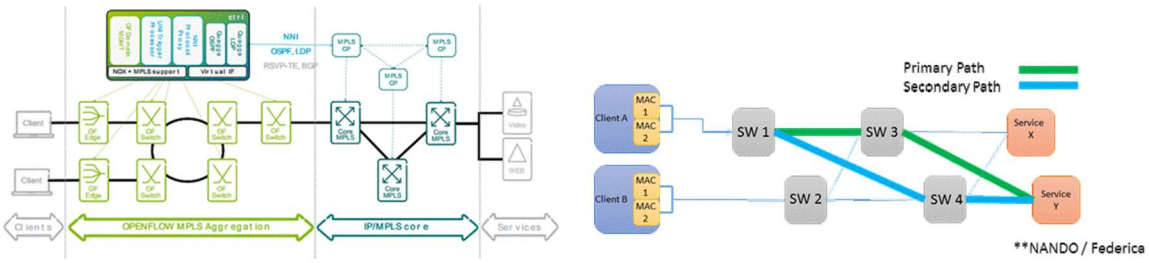


Figure 7 - EU FP7 SPARC (left). Project NANDO (right)

Substantial progress on creating flat, large-scale Layer 2 networks has been achieved in the area of Data Centres. In modern Data Centres, not only are there tens if not hundreds of thousands of physical machines, but each machine may have up to twenty tenant virtual machines. Each of these virtual machines must be addressable through a distinct layer 2 MAC address. There are different strategies referenced in the literature [101]. These include *IEEE TRILL Shortest Path Bridge (SPB), VL2, Portland, SEATTLE, Hedera, and BCube* which span the gamut of what is currently being tested and going through standardisation. Figure 8 shows the topology of the first four of these. TRILL uses a layer 2 link state protocol to identify the shortest paths between switches on a hop-by-hop basis, and load balance across these paths. This enhances scalability, allows loop-free multipath topologies and reduces excessively large MAC address tables (approaching 20,000 entries) that must be discovered and updated in conventional Ethernet networks. Shortest path bridging (*SPB*) is a layer 2 standard (IEEE 802.1aq) that attempts to address the same basic issue as *TRILL*, albeit in a slightly different approach. It uses the IEEE 802.1ah PBB provider link state bridging. The 802.1ah frame format provides a service identifier that is completely separate from the backbone MAC addresses and the VLAN IDs. This separates the connectivity services layer from the physical network infrastructure.

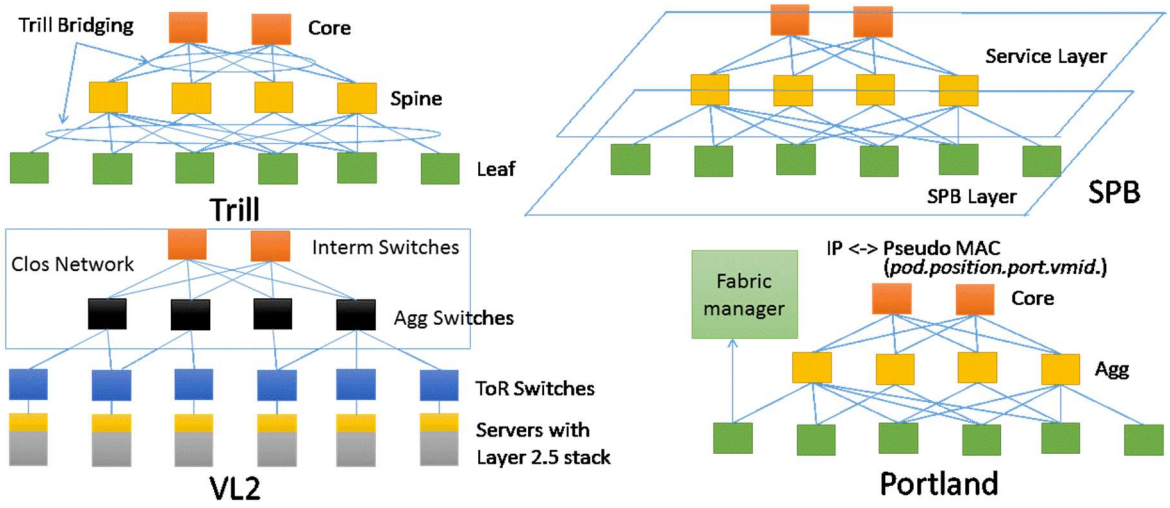


Figure 8 - Layer 2 Datacentre Architectures (Trill, SPB, VL2, and Portland)

VL2 uses a CLOS network topology with Valiant Load Balancing (VLB) with traffic being sent to random intermediate switches, resulting in small forwarding tables and hosts which can be independent of location in the data centre. Each Core switch is given the same Anycast address and ECMP is used to select a random shortest paths. OSPF builds forwarding tables between the switches, each of which is assigned a location-specific IP address. Real servers have Application-specific IP addresses so a centralised address manager is needed to maintain the mappings between Application and Local IP address mappings. In order to route on Local Addresses and deliver based on Application specific Addresses, IP-in-IP encapsulation is used. This requires a layer 2.5 stack which runs on each host in the *VL2* regime that consults the Address Manager for the mapping between the Application and Local IP address mappings prior to transmitting packets.

Portland uses a fat tree topology with pseudo or position-based MAC (PMAC) addresses to achieve a very compact routing state [102]. Top-of-the- Rack (ToR) aggregation switches are grouped together in pods, with every core switch being connected to every pod through a single link. Each Virtual Machine and real host is assigned a pseudo MAC address with information embedded related to the Pod identifier, its position in the pod, the port identifier and lastly it's Virtual Machine Identifier. Typical this of the form: ***pod.position.port.vmid***. By having a structured and predictable regime, as opposed to a typically random layer 2 addressing, opens up the possibility of best of Layer 2 and Layer 3 worlds. Wild carding of addresses can be used to route to pods, at a layer 2 level. Longest prefix-matches can be used to reduce forwarding state. Three components are needed to make the Portland strategy work. Firstly, in order to obviate the need for hosts and VMs to be aware of the top-level addressing structure, switches must rewrite between pseudo and real MAC addresses. Secondly, in order to calculate routes locally, switches must maintain a matrix of full link-connectivity. Lastly, a centralised fabric manager is required to maintain the mappings between pseudo and real MAC addresses.

0.

Chapter 3 SDN Control Plane for Converged Architecture

Software Defined Networks (SDN) separates the control and data planes in network components such as switches, bridges and routers. The SDN control plane can enable highly dynamic service and capacity provision over the LR-PON in response to changing demand by implementing agents in the network elements. The Logical SDN control plane architecture [103], shown in Figure 9, is derived from the Open Network Foundation SDN model [56], and is based on a hierarchical structure of controllers. The access network controller controls the access network elements and the core network controller controls the core transmission elements. The network orchestrator handles requests from application plane and translates them into high-level commands for the access and core network controllers.

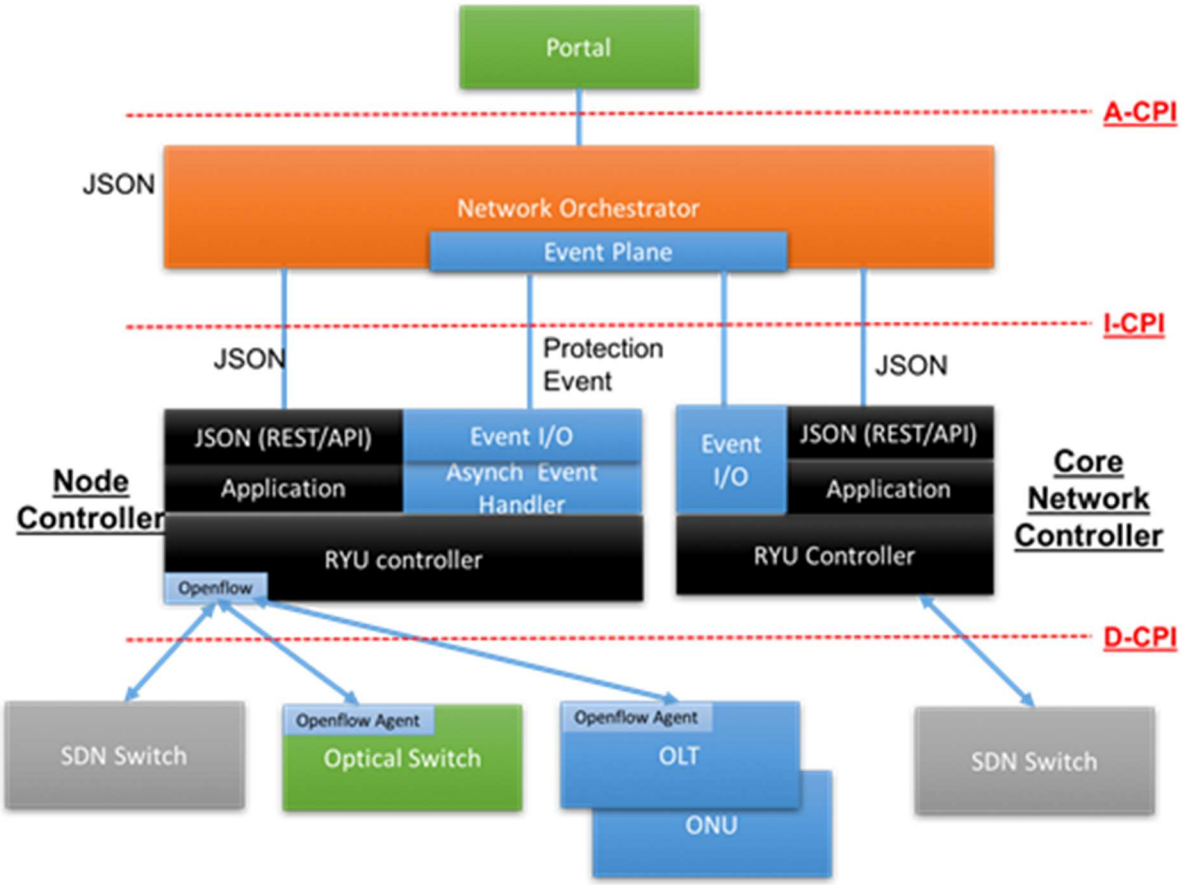


Figure 9 - Logical SDN Architecture

There are three main interfaces defined, the A-CPI, the I-CPI and the D-CPI interfaces. The A-CPI interface describes the interaction between the control plane and the application plane. This is the interaction between the service provider and the network orchestrator. The I-CPI interface operates between the network orchestrator and the access and core network controllers. Lastly, the D-CPI interface operates between the access and core controllers and the physical devices [104]. There are different protocols to cope with the functionalities in each interface.

In this section, we present the design of an SDN Controller for a Converged Architecture. We present the functional components (section 3.1) and detail the messages (section 3.2 that are exchanged between these components.

3.1 Functional Components

3.1.1 Network Orchestrator

The network orchestrator is defined as a parent controller or a centralised “controller of controllers”, which handles the automation of end-to-end connectivity provisioning, working at a higher, abstracted level and covering inter-domain aspects between the access and the metro/core network. The network orchestrator interfaces with the network controllers to get topological information about the resources in each controller’s domain. Each controller may have different interfaces, which requires the orchestrator to have a method to support multiple technologies or interfaces. When an application, such as Network Management System (NMS) or Operation Support System (OSS) requests a service, the network orchestrator must deal with the end-to-end path computation. This process can be done by the orchestrator or may be delegated to the access and core controllers. Once the services are set-up, the network orchestrator is in charge of update its status and notifying the application plane.

3.1.2 Core Network Controller

The core controller is in charge of receiving commands from the network orchestrator and transforming them in the D-CPI for the metro/core network. Similarly, it exports the topology to the network orchestrator, so it can have a view of the resources in the core network. The network orchestrator can request a path computation to the core controller, so it must support path computation within its domain. A core controller is used as an entity which is in charge of the specifics of the underlying core technologies. The technologies under the core controller are Optical Transport Network (OTN), Wavelength Switched Optical Network (WSON), Spectrum Switched Optical Network (SSON) networks, which are based on the GMPLS distributed control plane. If the GMPLS is enables, the best interface to interface with the nodes is Path Computation Element Protocol (PCEP), as demonstrated in ABNO [53].

3.1.3 Access Network Controller

The Access Network Controller translates requests from the Network Orchestrator into instructions for the physical devices, such as access switch, optical switch and OLT. The Access Network controller uses Openflow to manage the various network components in the access network, such as the optical switch, access switch and PON components. The Access Network Controller consists of a JSON RESTful API, an Application module, a

Database, and a RYU Openflow controller, which we will describe in detail. The JSON RESTful API module is an interface that translates a JSON request, which is received from the Network Orchestrator via the I-CPI interface which is then sent to the application module. JSON is a syntax for storing and exchanging data. It is written in a lightweight data-interchange format, which is less verbose than XML. JSON also describes data structures which includes arrays, whereas XML does not.

Application Module

The application module processes the incoming request, based on the state information present in the database. This module implements functionalities such as path calculation, path recovery, wavelength selection, bandwidth assignment, PseudoWire (PW) assignment and Link State Protocol (LSP) assignment. Based on the request and the network state, the application module determines whether the request can be satisfied or should be declined. The application module triggers the appropriate OpenFlow commands using the RYU OpenFlow controller. Thus the communication to the controller is carried out over the Openflow v1.4 protocol using the RYU Openflow controller application programming interfaces (APIs). The Network Controller sends an acknowledgment message back to the Network Orchestrator via I-CPI. The Network Controller maintains Openflow rules and meters in the access switch. A two-stage hierarchy of meters implements Peak Information Rate (PIR) and Committed Information Rate (CIR) Quality of Service characteristics for each flow. PIR is implemented by discarding packets that exceed the first meters bitrate. CIR is subsequently implemented by marking non-dropped packets as low priority (prec_level=0) that exceed the CIR bitrate defined by a second meter.

Database Module

The database module stores all information from the Metro Core node on routing, wavelengths, capacity, MPLS labels, and detail of flows and meters being used. Infrastructure information is held in database tables and relates to topology and the definition of ports, paths and host configuration (Figure 10).

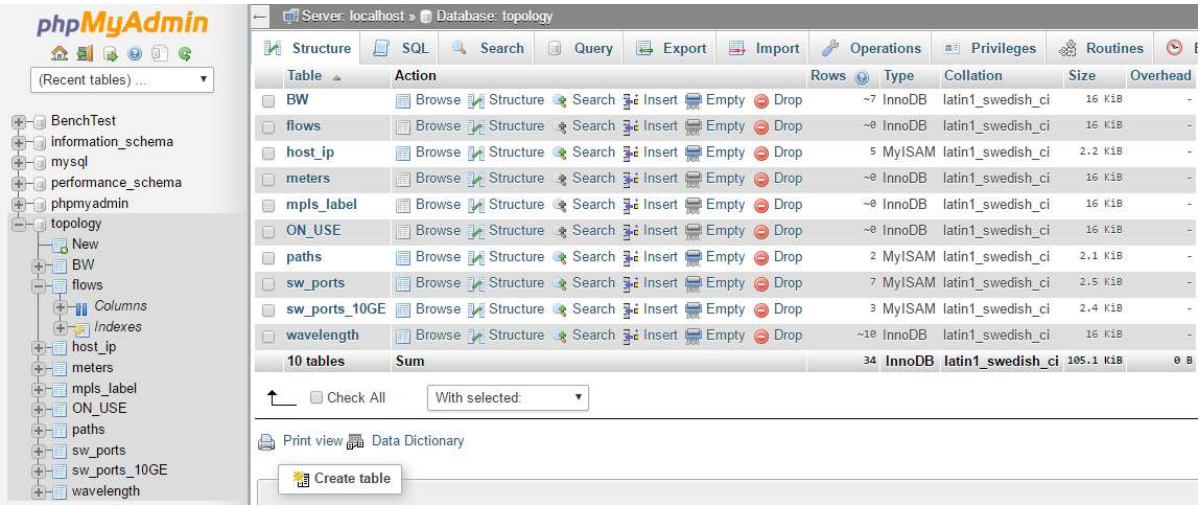


Figure 10 - Database Administration Interface

Logically, the database is broken into two distinct sub-databases, the Infrastructure Database and the Network State Information Database.

Infrastructure Database

The Infrastructure database stores information that is statically defined by a management or network control system and consists of the following tables: **PATHS, WAVELENGTHS, SW_PORTS** and **HOST_IP**. The **PATHS** table (Table 2) defines the route between sources and destinations.

id	src_IP	dst_IP	pri_path
48	10.0.0.77	10.0.0.103	105,7461338682660421663,103
49	10.0.0.99	10.0.0.102	101,7461338682660421663,102

Table 2 - Example of information in the PATHS table

Here, *src_IP* indicates an the source IP address of the path, for example the address of a SP video server, *dst_IP* indicates the IP address of the destination path, for example the address of an OLT that terminates a PW paths, *pri_path* defines a path between *src_IP* and *dst_IP*

Private IP addressing is used which is mapped to the HOST-IP table. The designation SW indicates a packet switch, while optical switches are designated as OSW.

The **WAVELENGTH** table (Table 3) associates wavelength ITU-T IDs with the wavelength ID recognised by the SFP+ tunable component.

wavelength_id	ITU_ID	Wavelength (nm)	FPGA_ID
1	21a	1560.61	21
2	21b	1560.2	22
3	22a	1559.79	23
4	22b	1559.33	24
5	23a	1558.98	25
6	23b	1558.58	26
7	24a	1558.17	27
8	24b	1557.77	28
9	25a	1557.36	29
10	25b	1556.96	30

Table 3 - Example of information in the WAVELENGTH table

Here, *wavelength_id* defines an ID number for the wavelength, *ITU_ID* is the standard wavelength ID defined by the ITU and *Wavelength* defines the value of the wavelength expressed in nanometres. *FPGA_ID* is the wavelength identifier used for controlling the tuneability of the SFP+ transceiver through the FPGA.

The **SW_PORTS** table (Table 4) defines the port connectivity of each switch which is statically allocated. In more complex implementations this information could be retrieved through discovery protocols.

SW_ID	ports_connection
7461348157366609941	101,14560991494939486229,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
14560991494939486229	102,7461348157366609941,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
7461338682660421663	0,102,0,0,0,0,0,103,0,0,0,0,0,105,0,0,0,0,0,101,0,0

Table 4 - Example of information in table SW_PORTS

Here, *SW_ID* shows the id number of a switch, *ports_connection* defines the port number of the switch and SW_ID that connects to the switch. In the example, switch with dpid 7461348157366609941 is connected to host 101 at port 1 and to switch with dpid 14560991494939486229 at port 2

The **HOST_IP** table (Table 5) defines the IP address for each host.

host_id	ip	PRI_OLT_IP	BK_OLT_IP
102	10.0.0.102		
104	10.0.0.88		
103	10.0.0.103		
101	10.0.0.99		
501	10.0.0.123	10.0.0.102	10.0.0.103

Table 5 - Example of information in the HOST_IP table

Here. *host_id* indicates the ID number of the host, which could be an SP server, ONU, or OLT. *ip* shows the IP address of the host, *PRI_OLT_IP* shows an IP address of a primary OLT (if the device is an ONU, *BK_OLT_IP* shows an IP address of a backup OLT if the device is an ONU.

Network state information database

This database contains the information used to identify the dynamic state of the network, including the flow table, the meter table, and the capacity table. This information is added, removed or modified dynamically following network operations. The **MPLS_LABEL** table defines LSP and PW numbers for each flow. Here, *src_IP* indicates the IP address of the source of the MPLS Pseudowire path, *dst_IP* indicates the IP address of the destination of the MPLS Pseudoswire path, *lsp* defines the higher level Link State Protocol that contains the PseudoWire. *pw* defines the label for the path and *traffic_type* indicates the traffic type carried by the PW. Here, traffic type is one of Internet, Video on Demand or Bandwidth on Demand.

The **SERVICES** table stores information on active services in the Metro Core node. *ID_Operation* is a reference ID number associated to an incoming request and is automatically generated after the Network Controller receives the request. *Operation_Type* indicates the type of service. *flow_id* refers to the *id* in the FLOWS table. *meter1* and *meter2* are the meter numbers that define the PIR and CIR respectively. *wavelength_no* refers to *walength_id* in the WAVELENGTH table. *pw* refers to the Pseudowire *pw* field in table MPLS_LABEL. *dst_host_id* refers to the destination ONU ID. The **METERS** table stores the configuration of the OpenFlow meters. Meters are unidirectional, thus if QoS is required on both direction of a connection, it is configured into two separate entries. Here, *ID_Operation* is the reference ID number for the request associated to the meter. *SW_ID* is the ID number of the switch where the meter is configure. *stream* indicates the direction for the meter, for example from source IP to destination IP address. *meter_id* is an identifier for the meter and is automatically generated after a request. *band_type* defines the action on the packet, which may be DROP for PIR, or DSCP_REMARK for CIR); *rate* is the capacity limit for the meter considered; *arguments* indicates an optional argument for the

meter (i.e., following the OpenFlow syntax). **The FLOWS** table stores the flows in use on the OpenFlow switches, indicating both matching condition and meters. *id* is a reference number for the flow; *SW_ID* is the ID number of the switch where the flow is installed; *ID_Operation* is the reference ID number of the request; *stream* indicates the direction of the flow (from source IP to destination IP address); *match_condition* refers to match field used in the OpenFlow table for the switch; *meter1* and *meter2* are meter numbers attached to the flow, controlling, respectively, PIR and CIR capacity. The **BW** table (Table 6) stores the available capacity for each wavelength on each link in the network, for both the PON channels and for the capacity between switches or between switches and other hosts. It is calculated and updated by the Network Controller upon every new request.

s_link	d_link	MAX	wavelength_ID	available_capacity
101	3492832460723070997	10000000	1	10000000
104	7461348157366609941	10000000	1	10000000
7461338682660421663	101	10000000	2	10000000
7461338682660421663	105	10000000	1	10000000
7461338682660421663	103	10000000	1	10000000
7461338682660421663	102	10000000	1	10000000
102	501	10000000	2	10000000

Table 6 - Example of information in the BW table

s_link is device_id at the beginning of the link; *d_link* is device_id at the end of the link; *MAX* is the total capacity of the link in bps; *wavelength_ID* is the identifier for the wavelength used; available_capacity shows remaining CIR bandwidth in bps. The BW table is used to assess the current available capacity on an end-to-end connection. For example, in the case of a VoD request with defined CIR and PIR parameters, the Network Controller will first determine the end-to-end path from the PATH table. It then checks the available capacity on each link making up the end-to-end path in the BE table.

3.1.4 **Open Flow Agent**

A number of network elements such as the PON ONU and OLT and the optical switch cannot be controlled through a native OF v1.4 protocol. We have developed an OF agent (Figure 11), running collocated with the controller, that emulates the Openflow protocol and provides a mediation interface between the upper Openflow Control plane and the lower non-native Openflow network devices such as the FPGA-based LR-PON OLTs and ONU. The OF agent interprets the Openflow commands coming from the Access Network controller and communicates various changes to the PON network via a UART control link in the FPGA hardware. The OLT and connected ONUs can be controlled via the access network controller like any other network components. The OLT uses a proprietary messaging system over a UART serial interface. The optical switch can be controlled through a TL1 session. Transaction language 1 (TL1) is a man-machine management

protocol defined by Bellcore and is commonly used to manage the optical broadband and access equipment [105].

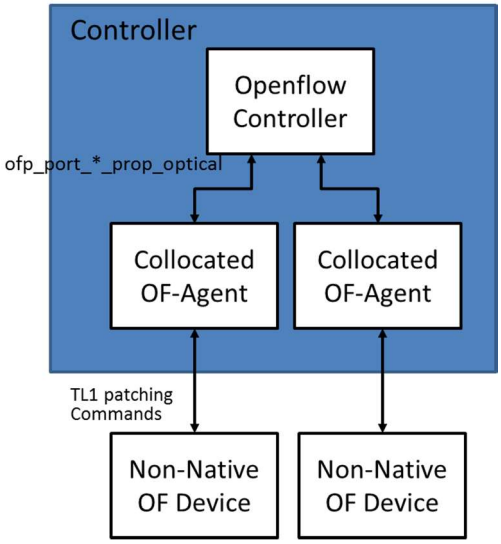


Figure 11 – OLT Fast protection mechanism

To control the Openflow Agent, the standard Openflow V1.4 syntax is used. Firstly, the Feature Mask is set to include capabilities for **OFPPC_PORT_DOWN**, **OFPPC_NO_RECV**, **OFPPC_NO_FWD** and **OFPPC_NO_PACKET_IN**. Secondly the **OFPPortModPropOptical** properites is built up based on **freq_lmda, fl_offset, grid_span, tx_pwr**. Lastly, the **OFPPortMod** method is called on the Openflow Agent datapath ID (or the MAC address) with the **OFPPortModPropOptical** properites and Feature Mask just described. The **OFPPC_PORT_DOWN** bit indicates that the port has been administratively brought down and should not be used by Openflow. The **OFPPC_NO_RECV** bit indicates that packets received on that port should be ignored. The **OFPPC_NO_FWD** bit indicates that Openflow should not send packets to that port. The **OFPPFL_NO_PACKET_IN** bit indicates that packets on that port that generate a table miss should never trigger a packet-in message to the controller.

The Openflow standard V1.4 introduced support for querying and managing the characteristics of Optical ports. This includes the ability to configure and monitor transmit and receive frequency of a laser, as well as its power. The Optical ports may be Ethernet based optical ports (i.e. Ethernet through SFP+ interfaces) or optical ports on circuit switches. This allows an Openflow controller to **configure** the optical ports through the Optical port mod property (**ofp_port_mod_prop_optical**), **monitor** the optical ports through Optical port stats property (**ofp_port_stats_prop_optical**) and to **describe** the optical ports through the Optical port description property (**ofp_port_desc_prop_optical**).

To support the mapping between the real device, Openflow Agent has a number of data structures defined on a per-port basis to handle the attributes required by the Openflow Controller. The **ofp_port_desc_prop_optical** data structure stores attributes for Maximum

and Minimum Transmitted Frequency , Maximum and Minimum Received, Maximum and Minimum Transmitted Power and Features. Features is a bit mask, with the bits representing the capabilities of the particular port. The right-hand most bit (**OFPOPF_RX_TUNE**) denotes if the Receiver is tuneable or not (**OFPOPF_TX_TUNE**), the second bit if the transmitter is tuneable or not (**OFPOPF_TX_PWR**), the third bit of the Power is configurable and lastly, the fourth bit denotes whether the to use frequency or wavelength (**OFPOPF_USE_FREQ**). For the Openflow Agent, all four bits are set to the on position. The minimum, maximum, and grid spacing are specified for both transmit and receive optical ports as either a frequency in MHz or wavelength (lambda) as nanometres scaled up by a factor of 100. For ports that are not tuneable, the minimum and maximum values need to be identical and so specify the fixed value.

The overall behaviour of the Openflow Agent is described by the Openflow Agent class. It has two main routines or methods, **wait_on_controller()** and **get_optical_message()**. On instantiation, the Openflow Agent makes a network connection to the standard TCP listening port for Openflow (6633). The **wait_on_controller()** routine is a loop that waits for Openflow messages. Once the upstream Openflow controller accepts the TCP connection, it issues an Openflow Hello message to which Openflow Agent responds with **Hello**. Thereafter, **wait_on_controller()** handles responses to upstream controller requests for message types **OFPT_ECHO_REQUEST**, **OFPT_BARRIER_REQUEST**, **OFPT_FEATURES_REQUEST**, **OFPT_PORT_MOD**, **OFPT_MULTIPART_REQUEST** and **OFPT_FLOW_MOD**. **Wait_on_controller()** uses the **OFPT_PORT_MOD** message with subtype **OFPT_PORT_MOD_OPTICAL** as a trigger to generate a message (**get_optical_message())** to the downstream Optical device, for which the Openflow Agent is acting as agent. **get_optical_message()** is a stub, which is over-ridden when Openflow Agent is instantiated.

The Openflow Agent is composed of 2 main sets of Classes. The first set of classes details the messages which are exchanged between the Openflow Agent and the upstream Openflow Controller. The Openflow Agent acts as a client, and the Openflow Controller acts as the server in the relationship. It is the function of the Openflow Agent to initiate the relationship, by contacting the controller. It is the function of the Openflow Agent to sufficiently mediate between the non-native Openflow device and the controller, and behave in a manner that will give the impression that the device is a real Openflow Switch. The base class of Openflow Agent message is **ofPkt**. All messages derive their attributes for transaction ID and payload from the **ofPkt** Class. **OfPkt** also details the versions of Openflow which the Openflow Agent can support. It is possible for the Openflow Agent and the Controller to negotiate to the most optimal version of Openflow to be used in the relationship. All messages have a common header with fields for the size of the message as well as detailing the message type. Depending on the message type, the payload of the

message will vary in length as well as format. Depending on the switch characteristics the size of messages can vary also. For example, there is a common format of data structure and command structure to manipulate a single port, however, different switches have different numbers of ports so the size of the total structures is sufficiently expandable.

3.1.5 PON ONU and OLT

The LR-PON Protocol is implemented over three Xilinx VC709 FPGA boards acting as primary and secondary OLTs and ONU. The LR-PON protocol is a partial implementation of the XGPON standard, the major differences being that the LR-PON protocol must work over a longer feeder fiber (125Km in LRPON as opposed to 20Km in XGPON) and across a higher split ratio (512 versus 64). The PON backplane connection to the core network contains a 10G Ethernet physical layer and Media Access Control Layer, allowing it to be plugged into any 10G capable network element. In this experiment the PON backplane is connected to a 48-port 10G Openflow Access/Metro switch. A Microblaze soft processor, which is collocated on the Virtex FPGA board, provides a (North Bound) UART management interface to the PON OLT and ONU hardware. Through this interface most PON functionality can be controlled such as resetting the hardware, viewing hardware status, simulating hardware failure, loading bandwidth map and setting XGEM mappings.

The OLT's and ONU's do not present native Openflow interfaces, but instead are controlled over a high-speed serial UART running at 406kbps. The Microblaze on the host FGPA boards presents an interface for directly programming and interrogating PON control registers, which are then accessible over the high-speed UART interface. Run-time control of the PON is executed through the interfaces on the OLTS which in turn relay control instructions to the remote ONU using PLOAM messages. The run-time functionality (see section 3.2.2) includes configuration of the laser frequencies of the OLT and ONU tuneable lasers, the configuration of Alloc_id's at the ONU for appropriate XGEM packets and the rehoming of ONU from one OLT to another.

An Openflow agent wrapper around both OLT1 and OLT2 was developed so as to present an Openflow v1.4 compatible interface to the Metro-Access Controller. Openflow v1.4 facilitates the control of optical parameters of Openflow compatible switches and devices through the **OFPPortModPropOptical** method. These parameters include the transmission centre frequency or wavelength, a frequency offset from the centre frequency and the transmission power level (dB) and are a subset of those which we are looking to control within the PON. Because we need runtime control of additional non-standard additional parameters, we enhanced both the v1.4 protocol and the agent to allow configuration of the XGEM, Alloc_ID and PseudoWire tags associated to a given flow through the Metro-Access Controller.

Logically, using the paradigm of Software Defined Networking, the Metro Access controller communicates with the OLTs through the Openflow agent translating Openflow commands into proprietary control messages sent through the UART interface, such as the set-up of wavelengths and the set-up of protection paths.

3.1.6 Distributed Message Queue

The Openflow protocol was originally intended as a means for the controller plane to push flow updates to the data-plane devices, and was not intended for responsiveness to real-time events. As a consequence, early controllers prioritised the fulfilment of the functional aspects of the Openflow standards rather than performance. There was a trade-off between developer productivity and performance when early Openflow controllers were developed to replicate the functionality of the original NOX controller [106]. A common means of increasing productivity (and making the applications portable across multiple platforms) was to abstract the functionality of the controller using class/object hierarchies. A number of controllers (such as RYU and POX) were rendered in Python in such a fashion. Because Python is a dynamically typed and semi-interpreted language applications requiring CPU (i.e. real-time) responsiveness may appear sluggish in the manipulation of both simple data structures and particularly more complex class/objects. Figure 4 of Erickson [106] shows the low throughput and high latency of standard Python based controllers such as POX and RYU, in comparison to compiled controllers.

In order to accelerate some of the time-critical messages at the I-CPI level, we created an asynchronous module to pick up the failure event form the OLT. This is then passed to an Event Plane, implemented using the ZeroMQ libraries [107]. which bypasses the slower JavaScript object notation (JSON) /API interfaces. We elected to use the open source ZeroMQ library which can handle up to 2.8 million messages per second and can open a TCP socket and process data within 28.45 microseconds. Any elements throughout the test bed can either publish or subscribe to topics on the event plane, through the use of the lightweight ZeroMQ API. This API is available for scripting and programming languages such as Perl, Python, Java and C. A key feature of ZeroMQ is that the Message Queue is logically centralised however; there is no physical hub through which all messages flow. This removes both single points of failure and performance bottlenecks. The event plane allows all major elements in the test bed such as the Openflow Controllers and the PON components to publish events using a common message format, as well as to subscribe to system wide broadcast events. We implemented four sets of messages which have a common format for control and co-ordination of events within the test bed. Our development optimised the real-time event-handling capability of the standard Python controller and extended the functionality across multiple controllers. We retained the basic shell of the

Python controller for the purposes of a standard Interface to the south-bound Openflow devices. This allowed us to leverage our existing code-base.

3.2 Messages

3.2.1 Control Plane Messages

Table 7 show the messages type implemented by the SDN controllers and orchestrator. The D-CPI messages shown in Table 7 are those between the OF agent and the physical device. The '**Patch_Connect**' and '**Patch_Disconnect'** messages are those that invoke connection between optical switch ports. These are translated over the interface to the TL1 '**ENT-PATCH::inport,outport**:;' and '**DLT-PATCH::inport,outport**:;' messages. Also the D-CPI messages from the Network Controller to the OF agent for the optical switch and OLT follow the OF v1.4 standard. The only extension to the standard OF protocol is that at the OLT for the wavelength selection at the OLT.

```
# Initialise variables. stream_id = 1 -> up, 0->down
c=3; fl=2000; gs=3000; tx=50; frq= 1500;stream_id=1;
# Set Feature Mask
mask = (ofp.OFPPC_PORT_DOWN | ofp.OFPPC_NO_RECV | ofp.OFPPC_NO_FWD |
ofp.OFPPC_NO_PACKET_IN)
# Set Properties
properties = [ofp_parser.OFPPortModPropOptical(1, configure=c, freq_lmda=frq,
fl_offset=fl, grid_span=gs, tx_pwr=tx, stream_id=stream_id)]
# Mod Port
req = ofp_parser.OFPPortMod(datapath, port_no, hw_addr, config,  mask, properties)
```

Figure 12 - OFPPortModPropOptical stream_id

The '**ofp_port_mod_prop_optical**' data structure and the '**OFPOPF Configure**' method which are defined in the OF v1.4 standard haves been enhanced to include an FGPA stream identifier (stream_id in Figure 12) used to differentiate between the upstream and downstream directions of the port considered.

These are then mapped to the parameters of the '**Set_US_lambda'** and '**Set_DS_lambda**' D-CPI functions (Table 8) from the OF agent to the FGPA Microblaze controller interface in the OLT.

Interface Type	Command Type	Source	Destination	Use Case	Main Parameters
D-CPI	Patch_Connect	Network Controller	Optical Swtich	Protect, DWA	Input-output ports
D-CPI	Patch_Disconnect	Network Controller	Optical Switch	Protect DWA	Input-output ports
D-CPI	Status_report	OLT	Network Controller	Protect, DWA	Status Values
D-CPI	Failure_Detect	OLT	Network Controller	Protect	ID of pre-set failure event
D-CPI	Create_Flow	Network Controller	OLT	Protect, DWA	MAX, xgem_port, mpls_tag
D-CPI	Delete_Flow	Network Controller	OLT	Protect,DWA	Flow_ID
D-CPI	Set_DS_lambda	Network Controller	OLT	DWA	ONU_ID, channel
D-CPI	Set_US_lambda	Network Controller	OLT	DWA	ONU_ID, channel
I-CPI	Failure_Detected	Network Controller	Network Orchestrator	Protect	
I-CPI	Invoke_Failover	Network Orchestrator	NC, CNC	Protect	
I-CPI	Client Failure Recovery	Network Orchestrator	Network Controller	Protect	
I-CPI	Create_Path	Network Orchestrator	Network Controller, Core Network Controller	DWA	Source, Destination, QoS params
A-CPI	Resource_Request	Portal	Network Orchestrator	DWA	Source, Destination, QoS params
A-CPI	Resource_Confirmation	Network Orchestrator	Portal	DWA	Request_ID

Table 7 - List of Main Control Plane Messages

The '**Status_Report'** is a general message to report the status of a given parameter. The '**Failure_Detected'** message indicates that the OLT has identified a failure and it triggers the protection action at the Network Controller. The '**Create_Flow'** creates an entry on the OLT flow table, providing flow association between the device MAC address, the ONU XG-PON encapsulation method (XGEM) port and the Multiprotocol Label Switching (MPLS) label (this isused to identify a Pseudo-Wire in this part of the network). The Network Controller uses the '**Set_DS_lambda'** and '**Set_US_lambda'** for indicating the downstream

downstream and upstream upstream transmission wavelengths to the ONU. Once the ONU has received the corresponding XG-PON physical layer operation, administration and maintenance (PLOAM) message, it sends a '**write WL<lambda>**' over the UART interface to change the centre wavelength of its tunable filter. This last command is sent to the OLT, which then generates a PLOAM message to communicate with the ONU.

While our implementation of the Network Controller to OLT is proprietary, the Broadband forum has initiated an effort to standardise define a D-CPI OF interface for PONs [109]. The I-CPI layer takes charge of the messaging between the Network Orchestrator and node controller NCs. The messages used in our case are: "**Failure_Detected**" reporting from the Network Controller to the network orchestrator that one of the OLT connections has failed; '**Invoke_Failback'**, used by the network orchestrator to activate the pre-configured protection path on the NCs; "**Client_Failure_Recovery**", is used by the network orchestrator to inform the Network Controller that the protection was successfully established. Alternatively, the DWA use case employs instead a '**Create_Path**" message with source, destination and quality of service (QoS) parameters (triggered by the '**Resource_Request'** message described below). In the protection experiments [107, 110, 111], we have used a proprietary interface for the I-CPI, although in [112] we have demonstrated the interoperability of our controller NC with the Control Orchestration Protocol (COP) [113].

The A-CPI interface operates between a user portal and the network orchestrator. In the DWA use case, a "**Resource_Request**" is sent by the portal to the network orchestrator indicating the source and destination points as well as the relevant QoS parameters (e.g., CIR and PIR) such as committed information rate (CIR) and peak information rate (PIR).The demonstrated architecture also integrates an SDN control plane for the access and core network elements, showing a fast protection mechanism, in the case of primary backhaul link failure, with service restoration and the dynamic reassignment of an ONU wavelength in response to increased traffic demand.

3.2.2 Openflow Messages

All Openflow messages follow a common format, however, the format has grown from a simple one format in the 0x01 wire standards to the more complex formats of wire standards 0x04 and later. The increase in complexity of the messages reflects the complexity of the type of functions and controls which more recent type of Openflow switches are required to exercise. Switches are required to be stateful, and have wider varieties of port attributes. An example of complex messaging introduced is the Barrier Message, which must be interpreted by Openflow Agent. When the controller wants to ensure message dependencies have been met or wants to receive notifications for completed operations by Openflow Agent, it uses an **OFPT_BARRIER_REQUEST** message. This message has no

body. Upon receipt, the Openflow Agent must finish processing all previously-received messages, including sending corresponding reply or error messages, before executing any messages beyond the Barrier Request. When such processing is complete, the Openflow Agent must send an **OFPT_BARRIER_REPLY** message with the transaction id of the original request.

Messages are characterised as either synchronous or asynchronous. Asynchronous messages can be sent by either the Openflow Agent or the Controller, and elicit a response from the other party. Example of asynchronous messages are Hello and EchoRequest. Hello elicits Hello in return, and is used to initiate the relationship between the Openflow Agent and the Controller, however in practice, this usually is initiated by the Openflow controller. **EchoRequest** elicits **EchoReply** in response and is typically used as a health check or keep-alive message. Synchronous messages generally are initiated by the controller, once the relationship has been created. Synchronous messages are used to elicit information from the Openflow Agent by the Controller. Examples of Synchronous messages are **OFPT_FEATURES_REQUEST**, **OFPT_PORT_MOD** and **OFPT_FLOW_MOD**. **OFPT_FEATURES_REQUEST** is the request for the characteristics of the switch as well as the capabilities of all ports.

3.2.3 PON wrapper methods

Software-wise, communication with the OLT's and ONU's is accomplished through a single Class FPGA, through which a number of methods are defined. Instantiation of the class opens a TTY serial interface port to the particular device. The sendcmd() method is the base method to issue a string to the interface and receive back a response. sendcmd() is used by almost all other FGPA methods to issue commands and gather responses. The raw FPGA interface presents a menu structure, which is invoked by the FPGA class. The following table outlines the range of based methods which may be issued on the PON devices.

Method	Explanation
device.reset()	Resets Device
device.enable_mpls()	Enables MPLS tagging interpretation on the device. This command is issued at the head-end OLT.
device.disable_mpls()	Disables MPLGS tagging interpretation on the device. This command is issued at the head-end OLT.
device.set_ds_laser(fpga_channel)	Sets downstream laser to channel designated by fpga_channel (Skylane mapping). This command is issued at the head-end OLT.
device.set_us_laser(onu_id, fpga_channel)	Sets downstream laser to channel designated by fpga_channel (Skylane mapping) on the ONU onu_id. This command is issued at the head-end OLT.
device.set_alloc_id(onu_id, alloc_id)	Sets the alloc id / XGEM port on the ONU onu_id
device.create_flow(mac, xgem, mpls_tag, cam)	Creates a flow denoted by mac mac address, XGEM port xgem, MPLS tag mpls_tag on cam cam. This command is issued at the head-end OLT.
device.delete_flow(cam)	Deletes flow in cam cam_id
device.dwa_set()	Completes the set up of the DWA mapping on the PON. This command is issued at the head-end OLT.
device.dwa_reset()	Resets the DWA mapping on the PON. This command is issued at the head-end OLT.
device.getstatus()	Returns the status of the device. This may be on any deivce
device.readreg()	Returns the register value. This may be on any device

Table 8 - PON Methods

3.2.4 Event Plane Messages

The message format is composed of a major category (called a ZeroMQ topic); a global timestamp which is synchronized to Dublin time; a minor category which is used for a command or message payload. The four messages types are as follows:

i. unsolicited events of large significance such as the failure of major nodes and links: an example of a primary PON failure event showing the major category, timestamp and minor category is *NetEvent 1422736912.30 OLT_P_Failure,*

ii. reactive control messages, which are generated in response to unsolicited events, for example, messages that trigger takeover of service by a standby piece of equipment or Service. An example of an Optical Switch control message showing the major category, timestamp and minor category is *GlimEvent 1422737623.07 upSdownP*.

iii. proactive configuration of elements or sub-systems within the testbed. An example of a sub-system restart event showing the major category, timestamp and minor category is *SysControl 1422737623.07 Restart*.

iv. The control of or alerting within test routines. These messages serve to co-ordinate the actions of a number of agents involved in a test cycle which are located across the wide area testbed.

3.3 Sample Configuration

The example of executing a resource request through the Access Controller A-CPI interface is given as follows. The sample topology in Figure 13 shows two traffic sources 10.0.0.88 and 10.0.0.99 sending traffic streams to a common ONU (10.0.0.123) along paths which are designated green and red respectively. The red flow is tagged with pseudowire labels 2001 and the green flow is tagged with pseudowire label 2002. The pseudowire labels are popped at the OLTs.

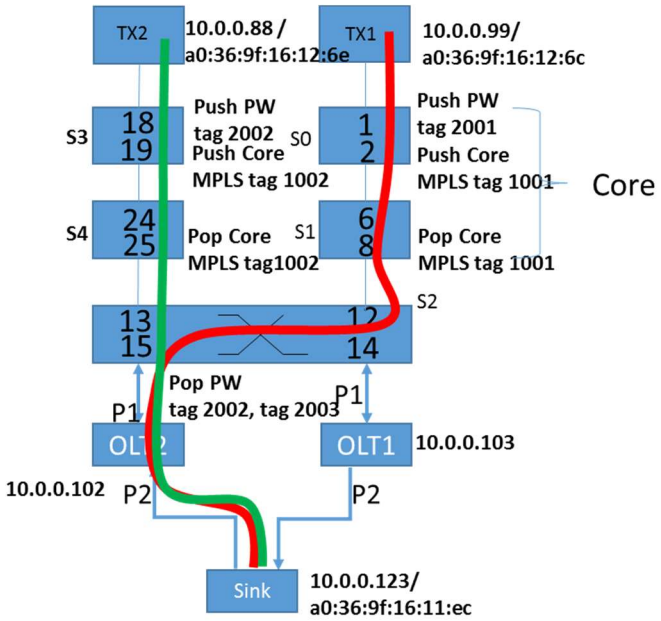


Figure 13 - Example topology

To emulate the use of MPLS tunnelling across the core network, the core switches S0 and S1 push and pop mpls tag 1001 on the red stream. Similarly, core switches S3 and S4 push and pop mpls tag 1002 on the green stream. Figure 14 and Figure 15 show the RESTful JSON API calls to the Network Controller to install the Red and Green Traffic flows respectively.

```
curl -X POST -d '{"Direction_Type": "Unidirectional", "Wavelength_Range_from": "1556", "Destination_Node": "501", "PRI_OLT_IP": "10.0.0.102", "ONU_IP": "10.0.0.123", "Wavelength_Range_to": "1561", "Bandwidth": {"CIR": "4000000", "PIR": "6000000"}, "BK_OLT_IP": "10.0.0.103", "Source_Node": "10.0.0.99", "New_Lambda": "None", "Operation_Type": "PWMPLSProvisioningWF", "Traffic_Type": "Internet", "Operation": "add"}' http://127.0.0.1:8080/stats/flowentry/add
```

Figure 14 - Install Red Traffic flows

The Red Traffic flow has Traffic Type Internet and requires a guaranteed bandwidth of 4Mbps with an ability to peak up to 6 Mbps. The Green Traffic flow has Traffic Type VoD (Video on Demand) and requires a guaranteed bandwidth of 8Mbps with an ability to peak up to 10 Mbps.

```
curl -X POST -d '{"Direction_Type": "Unidirectional", "Wavelength_Range_from": "1556", "Destination_Node": "501", "PRI_OLT_IP": "10.0.0.102", "ONU_IP": "10.0.0.123", "Wavelength_Range_to": "1561", "Bandwidth": {"CIR": "8000000", "PIR": "10000000"}, "BK_OLT_IP": "10.0.0.103", "Source_Node": "10.0.0.88", "New_Lambda": "Necessitated", "Operation_Type": "PWMPLSProvisioningWF", "Traffic_Type": "VoD", "Operation": "add"}' http://127.0.0.1:8080/stats/flowentry/add
```

Figure 15 - Install Green flows

For the purposes of demonstrating Assured bandwidth, a dedicated wavelength may be necessitated. Figure 16 shows the release of the Red and Green Traffic flows

```
curl -X POST -d '{"pw": "2001", "ID_Operation": "", "Operation": "release"}' http://127.0.0.1:8080/stats/flowentry/release

curl -X POST -d '{"pw": "2002", "ID_Operation": "", "Operation": "release"}' http://127.0.0.1:8080/stats/flowentry/release
```

Figure 16 - Release Red and Green Traffic flows

Chapter 5 Converged Architecture Fast Protection

In a classical telecommunications architecture, each portion of the metro network serves potentially hundreds of residential and business users. In a converged architecture such as Long-Reach PON, this metro network is being replace by backhaul links which must be made dual parented using a secondary backhaul. Fast protection of these links is required in order to fulfil requirements by enterprise and mobile backhaul applications. Fast PON protection enabled by SDN control also allows the implementation of protection load balancing schemes, which allow substantial cost reduction in both IP and PON backup resources by increasing the ability to share protection equipment across the network.

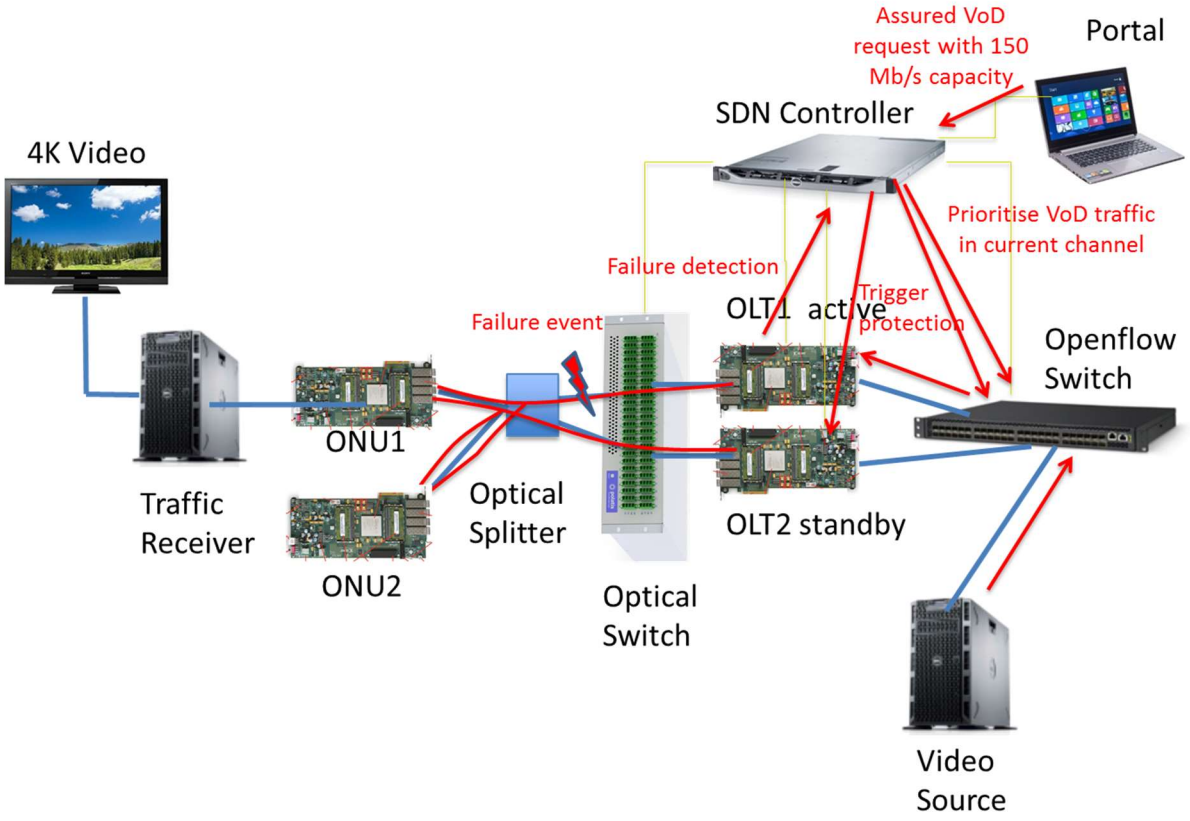


Figure 17 - Fast Protection Scenario

Figure 17 portrays the fundamental SDN enabled protection scenario which we demonstrate in this section. A traffic generator transmits traffic to a traffic receiver across a PON. The ONU is homed off a primary OLT through a fibre distribution path that involves an Optical Switch. There is a cut of the fibre, which is sensed by the primary OLT. The primary OLT sends an in-band alert to the upstream SDN controller, that then instigates a number of steps to fail the PON over to a secondary path. We execute a number of iterations of the protection experiments from a 1+1 (Active-Active) , through 1:1 (Active-Standby) to N:1 (Active-shared Standby) protection scenarios. The protection experiments evolve to include more physical layer components as they became available, and also encompass

different complexities of core network such as transcontinental core networks and national networks.

5.1 1:1 Protection Scheme with Pan-European Core

5.1.1 Configuration

We investigated a 1:1 protection scheme, where a primary OLT had a dedicated backup OLT, but without traffic duplication in the core. Our objective was to implement an end-to-end protection switching scheme across the access and core networks that operates in the tens of milliseconds [110].

The experiment for the combined access and core networks spanned the optical architecture test bed in Trinity College Dublin and the GEANT Openflow facility. The GÉANT Openflow facility is a test-bed environment deployed on top of the GÉANT pan-European research and education network and provides network resources such as software-based OVSwitch soft Openflow switches and interconnecting network links. The GEANT Openflow facility is collocated with five of the GEANT network Points-of-Presence in Vienna (AT), Frankfurt (DE), London (UK), Amsterdam (NL) and Zagreb (HR). The OFELIA Control Framework (OCF) is used by the GÉANT Openflow facility to manage requests for slice submission, instantiation, and decommissioning [114]. OCF is a set of software tools for testbed management, which controls the experimentation life cycle such as resource reservation, instantiation, configuration, monitoring. Connectivity between the two access and core portions of the network was achieved over the Internet. While this connectivity would ideally be over a dedicated fibre link, this setup allowed us to replicate latency effects between diverse network elements and the higher-level control layers [115].

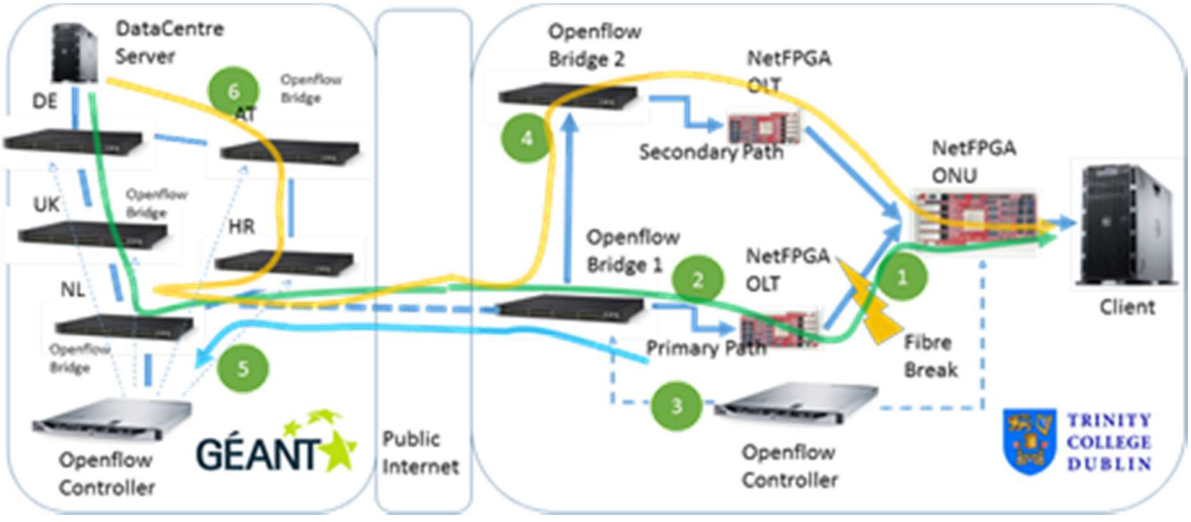


Figure 18 - Modelled combined LR-PON access and core network, with multi-tier Control Plane

In GEANT, we created a five-node network topology, with nodes in Amsterdam (NL), Frankfurt (DE), Hungary (HR), Austria (AT) and London (UK). Collocated with Node DE was a server which acted as a Data Centre. The primary path in the core is through nodes DE,

UK and NL, with the diverse fall back path from nodes DE through AT and HR to NL. Since node NL was the only gateway to our access network, both primary and protection paths pass through this node, while in a more realistic scenarios the node hosting the backup OLT would be connected directly to a node on the secondary path. However, this served the purpose of carrying out core network redirection. NL also hosted the Openflow core network controller.

For our access network, the configuration was comprised of a Pronto 3780 switch with 48 10G interfaces, running release 2.2 (Openflow v1.3 compatible firmware), and a Hitech Global 10G NetFPGA board acting as twin OLTs and ONU. The Pronto switch was configured with multiple virtual bridges. A VPN tunnel extended between the access and the core network gateways. A Dell R320 acted as the Openflow access controllers running RYU. In our 1:1 scheme, data travelled through the primary OLT and the primary bridge. The standby path, through the secondary OLT and secondary bridge did not carry any traffic until it was invoked, at which stage all primary traffic was redirected. Link availability was determined by the transmission of a UDP packet every 1 milliseconds between the Data Centre (DE node) and the Client which was terminated on the ONU in the TCD testbed.

5.1.2 Results

In our 1:1 protection experiment described in Figure 18, the feeder fibre between the primary OLT and first stage splitter on the PON was cut (event 1). This stopped all data from being transmitted upstream or downstream on the Primary link. A hardware unit in the primary OLT FPGA monitored the upstream data. The hardware detection unit alerted the OLT controller which sent an in-band upstream alarm (event 2). An upstream alarm was required, because the Openflow bridge did not physically terminate the connection between the ONU and the OLT, which meant that it was not possible for Openflow path switching rules (such as Group based port protection) to be invoked due to the fibre cut. Once any ONU has been registered on a particular PON, the upstream fibre should be quiescent for no longer than a single quiet window [116]. For a LR-PON of 125 Km, as proposed by the DISCUS project, this would be equivalent to no more than 1.3 milliseconds. Taking round trip time into account, the hardware failure detection unit (Figure 11) would detect a break in the fibre in approximately 2.5 milliseconds. Next, the data plane of the Openflow-based primary bridge intercepted the upstream alarm, which it then forwards to the access node controller (event 3). The Openflow controller instructed the Openflow switches to route traffic through the secondary bridge and the backup-OLT (event 4). The upstream alarm was also sent to the Openflow based controller for the core network (event 5). The core network controller built the backup path in the core from the nodes DE through AT and HR to NL (event 6). While it was not possible for Openflow path switching rules to be invoked directly by interception of this alarm (that is, solely within the data plane), we had devised

an Openflow relay (OF-Relay), located on-board the Pronto switch, that performed fast updating of the access fast recovery rules on the switch, as well as forwarding the alarm to the higher layer control infrastructure. The fast recovery paths were invoked and revoked by an application of a single goto_table statement injected into the primary switch. Results were measured over 50 failure – restoration cycles. Figure 19 shows the variance in fast recovery times in the access and core networks.

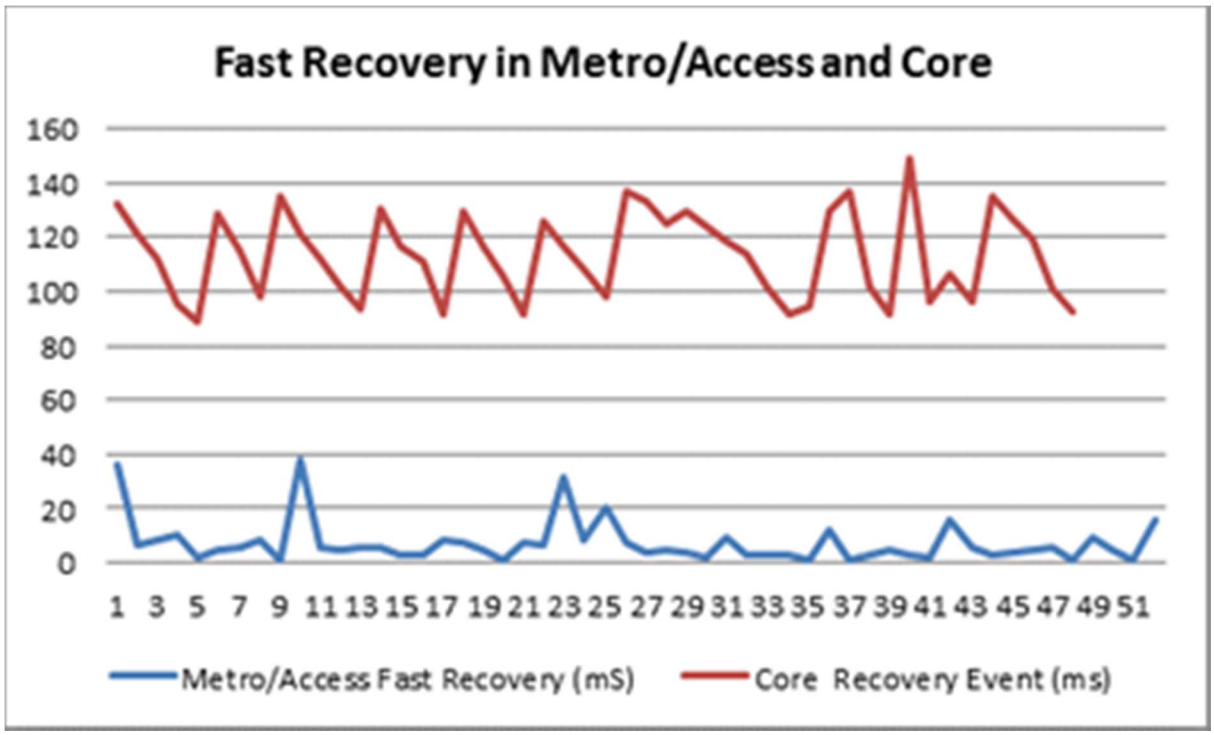


Figure 19 - Fast Recovery in access and core

Figure 20 shows that full recovery took place over an elapsed time period of 124 milliseconds. This was composed of 3 individual time periods - a period in which traffic in the access was failed over from the primary path to the secondary path (7.2 milliseconds); a period in which core traffic was being redirected before the service could be restored (25 milliseconds) and lastly an intervening period in which the end to end link was in flux (92 milliseconds).

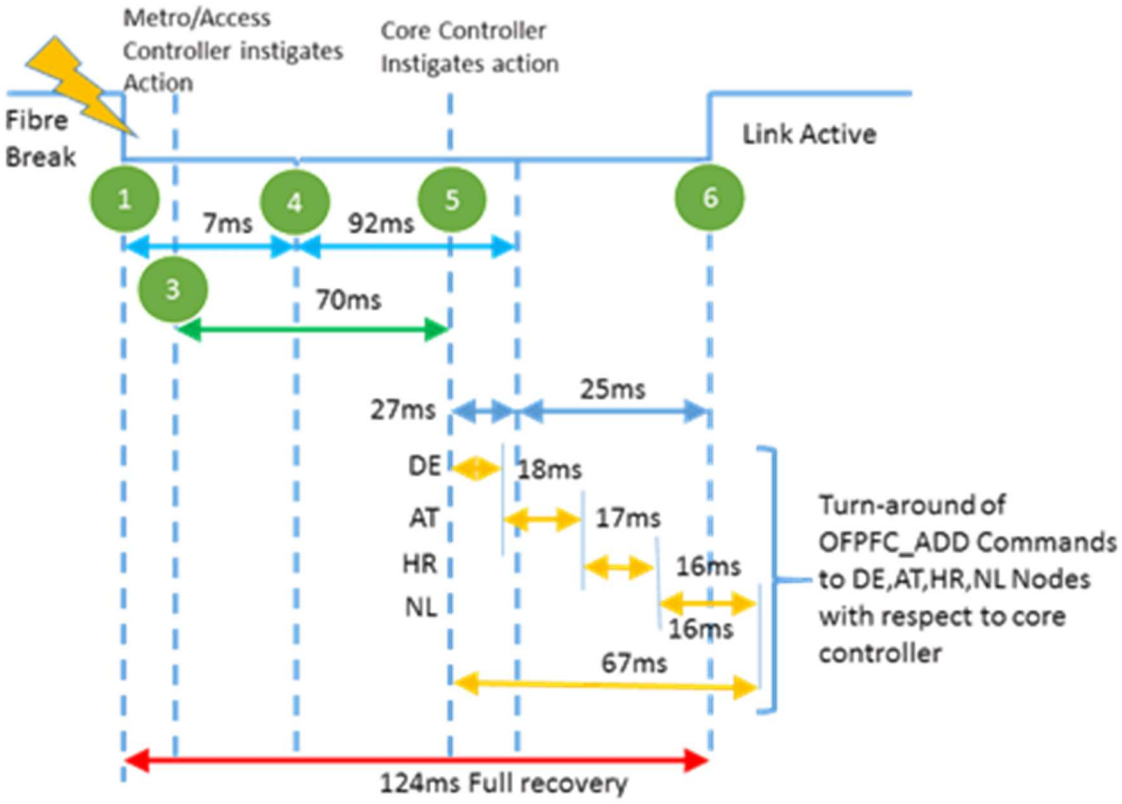


Figure 20 - Multi-tier protection events.

5.2 N:1 Protection Scheme with Pan-European Core

5.2.1 Configuration

The N:1 protection with Pan-European core experiment [107] evaluated a dual-home Long-Reach Passive Optical Network (LR-PON) protection mechanism where backup OLTs were shared among PONs in an N:1 scheme, and the service restoration was provided over an end-to-end Software Defined Network (SDN) controlled core network . In our test scenario, we simulated a cut in the feeder fibre between the primary OLT and first stage splitter on the PON. This stopped all upstream and downstream data on the Primary link. A hardware unit in the primary OLT FPGA monitored the upstream data path. Even in cases were the two directions of communication were operated over separate fibres, a cut in the downstream fibre prevents all ONUs from receiving messages form the OLT. In this case all ONUs automatically stopped transmitting, so that the OLT does not receive any upstream data. This upstream silence activates a timer. If this timer expires an alarm is raised to initiate a protection switchover. The duration of this timer would take into account all normal silences on the PON, namely quiet windows and normal roundtrip time, in order to make sure that an alarm was only raised when a failure occurred. Thus on a LR-PON of 125 Km, like the one proposed by the DISCUS project, failure detection could take approximately 2.5

milliseconds in worst case conditions (Figure 19). The 2.5 milliseconds is composed of 1.25 milliseconds of round-trip fibre delay and 1.25 milliseconds for quiet window.

Since our aim was to investigate end-to-end network protection times, we incorporated the control plane for the access and core nodes. For our experiment, we used Openflow-controlled access and core networks, each using an independent Openflow controller. Thus, when a failure was detected, the hardware detection unit alerted the OLT controller which sent an in-band upstream alarm to the Openflow access network controller. An upstream alarm was required, because the Openflow Bridge did not physically terminate the connection between the ONU and the OLT, which meant that it was not possible for Openflow path switching rules, such as Group based port protection, to be invoked due to the fibre cut.

We tested our end-to-end protection service with dual-homed, N:1 backup OLT sharing by combining the optical architecture testbed in Trinity College Dublin and the GÉANT pan-European research network, as shown in Figure 21. The testbeds were connected through two dedicated Gigabit Ethernet links. Although this link was well below the 10Gb capacity of the LR-PON, having dedicated data links allows us to reliably evaluate latency effects between diverse network elements and the higher-level control layers. The experiment replicated both the metro-access and core networks of a high-speed fixed line telecommunications network. The end-points replicated a Data Centres generating traffic (located in Frankfurt) and a reception or termination point located on a PON ONU in Dublin. The Metro-Access portion of the network was created in the Optical Network Architecture (ONA) lab in Trinity College Dublin. The Core network was replicated using the GÉANT Openflow testbed facility, which spanned continental Europe.

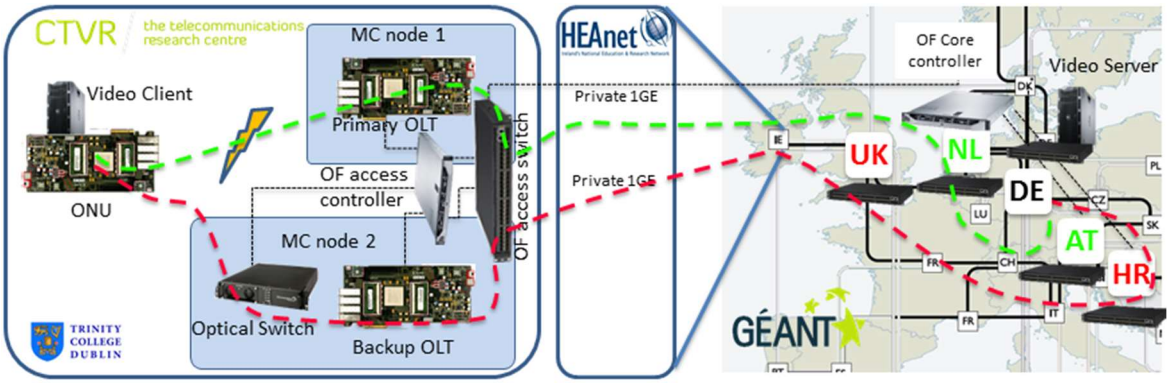


Figure 21 - Logical view of combined LR-PON access and SDN Core network

A server co-located at the DE node acted as the source for data in this experiment. The primary data path in the core is between nodes DE, AT and NL (shown in green in Figure 21). The backup path followed the route: DE, HR and UK (shown in red Figure 21). We implemented two paths to emulate dual-homed PON network where the primary and the backup OLT were in different locations. Data on the primary data path was routed to the

Primary OLT and data on the backup link was routed to the backup OLT. The NL node also hosted the Openflow core network controller which could be used to control which of the two paths data takes to our access network.

The TCD ONA (Optical Network Architecture) was setup as two Metro/Core nodes together with an LR-PON access network. Although we used one physical switch and server, they were both virtualised to represent independent MC node switches and controllers. The Metro/Access network comprised of a pronto 3780 switch, running release 2.2 (Openflow v1.3 compatible firmware), three Xilinx vc709 development boards acting as primary OLT, backup OLT and ONU, a Glimmerglass MEMs-based optical fibre switch, A Dell T620 with 10G SFP+ cards acting as client machine attached to the ONU and a separate Dell R320 Server acting as access Openflow controller. The Pronto switch was configured as multiple virtual bridges to act as standalone bridges each with a separate Openflow controller. These were connected to a gateway machine on the core side of the network and one of the Primary or backup OLTs on the access side. The two testbeds, TCD and GÉANT were connected via two dedicated 1Gb links to UK and NL respectively.

In our previous experiment in 1:1 protection (see section 5.2), we utilised tunnelling over the internet for these links which added a variance to our results that was very hard to measure and account for in our results. Although the dedicated 1Gb links were well below the 10Gb capacity of the GÉANT and TCD testbeds they do offer a stable link that enables us to carry out the protection experiments to the desired precision. The Glimmerglass optical switch was connected between the backup-OLT and the ONU which allowed the backup OLT to switch between a number of different PONs allowing us to test the N:1 protection timing. In order to extend Message Queue from TCD testbed to GÉANT, we implemented tunnels to NL and DE.

Our test scenario used two independent Openflow controllers, the core controller and the access controller. The OLT issued an alarm to the access Openflow controller when a failure was detected. The access controller enabled a data route through the backup Openflow Bridge, activated the backup OLT and tuned the optical switch to route data from the backup OLT to the failed PON backup fibre. In parallel it communicated with the core controller to activate the pre-calculated protection route in the core, which connected the remote server with the MC node where the backup OLT was located.

5.2.2 Testing Procedure

The TCD ONA testbed was designed to ensure all tests were easily reproducible, regardless of when they are run and by whom. All testbed components were completely programmable using a Python (v2.7) based object Framework. This allowed us to centrally control all testbed components so that test scenarios could be set up quickly and consistently. Likewise all test components logged information to a central repository, with

clear and consistent messages and time stamps. This allowed us to run test scenarios repeatedly allowing for statistical analysis of means and deviations of measurements.

In this scenario, a Python scenario script was initially used to check the status of all relevant components. It uploaded a bootstrap configuration to an Openflow Switch, set up the Openflow controllers, set up the event logger, configured the optical switch and flashed the FPGA images. The OpenVSwitch (OVS) was restarted and connected to the controller. The switches and ports were defined and associated with the OVS instance. For each switch, the required flows were configured. The testbed controller then linked to the OLT and ONU microprocessors to ensure the PON were operational. Finally, the data service on the DE node of the GÉANT network were enabled and end to end operation of the system were confirmed.

Once the experiment were started, data start flowing over the primary link from node DE through AT and NL to our testbed and through the PON to the data sink connected to the ONU. After some time a trigger signal were sent to the primary OLT FPGA which resetted the optical channel to simulate a fibre dig up. At this stage the protocol hardware was unaware of the break and packets are lost. The failure detection timeout timer started and when it expired the primary OLT issued an alarm which was sent in band upstream to the Openflow controller. The Openflow access network controller notified the optical switch to connect the backup OLT path to the failed PON and finally the management controller notified the backup OLT to take control of the PON. The ONU meanwhile entered the Loss of Downstream Sync state and remained there for 100 milliseconds or until the backup OLT began to send synchronization words downstream. If the backup OLT did not take over before the 100 milliseconds time out the entire PON would have to be reactivated, and re-ranged to resume transmitting data. Once the backup OLT had taken control of the PON the PON was ready to start receiving data again.

In parallel with the backup OLT taking control of the PON, the Openflow access network controller passed a message to the core network Openflow controller. This caused the core network to redirect data from the primary path to the backup path to emulate more closely the dual homed nature of the Long-Reach PON. When data services resumed, data was then flowing over the backup-path from DE to HR, UK through our backup OLT to the ONU. Since each of the packets being sent on the PON had a sequence number the ONU could easily work out how many packets were dropped during the switchover. Once this number had been calculated the scenario script was ready to restart a new iteration of the experiment.

In the first 1:1 experiment, (see section 5.1) we had noticed a significant variation in the time for the protection path to be fully activated, ranging from 79 milliseconds up to 133 milliseconds. On analysis, this high level of variation had been caused by two factors. Firstly, there was variation in the time between the receipt of a PON failure alarm by the Optical

Network Architecture (ONA) Openflow controller and the subsequent action taken by both the ONA and GÉANT POX controllers. Secondly, there had been significant variation in the relaying and tunnelling of signals between the ONA and GÉANT POX controllers. In order to reduce the latency and variation within both the controllers and the tunnels between the controllers, we implemented an event plane based on a fast, low latency distributed message queuing architecture (See section 3.1.6). Figure 22 shows the logical configuration of the Test bed event plane based on the distributed ZeroMQ Message Queue described in section 3.1.6

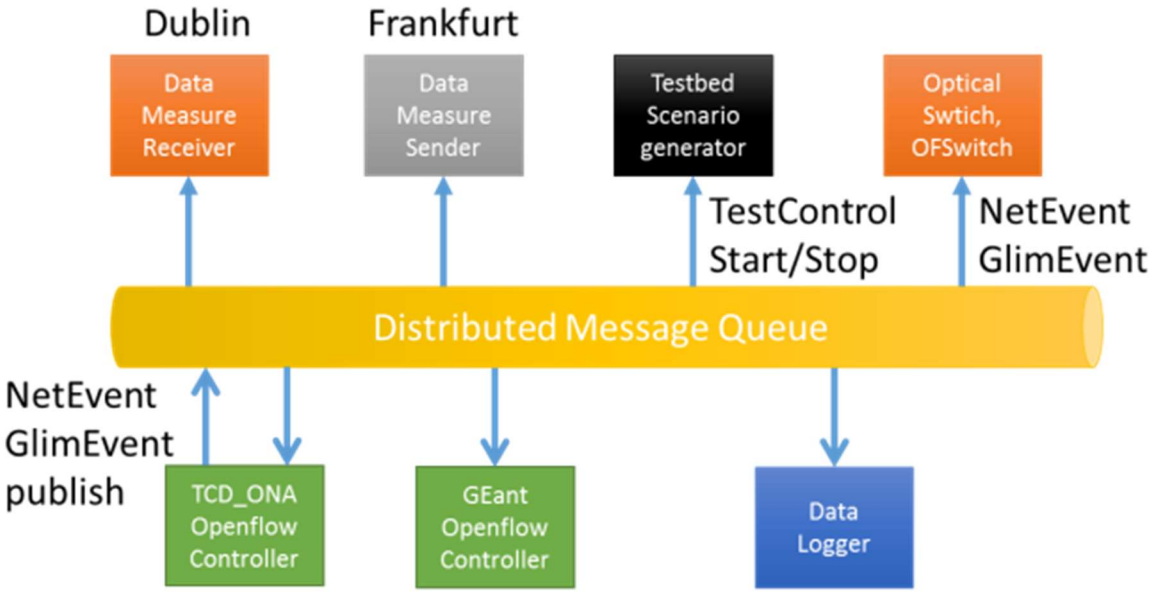


Figure 22 – Event plane based on distributed ZeroMQ Message Queue

The TCD_ONA Openflow Controller intercepts the downstream failure alarm in the primary PON. As well as triggering Openflow switching rules in the Metro Core network, the TCD_ONA controller also publishes Message Queue NetEvent and GlimEvent messages. The NetEvent broadcasts to all subscribed components about the PON failure. One such subscribe component is the GÉANT Openflow Controller which executes secondary routing in the network core. The Message Queue was extended between the TCD_ONA controller and GÉANT Openflow Controller using an SSH tunnel. The GlimEvent triggers Optical Switch path selection or protection Path in the Access Optical Switch. The Optical Switch subsystem was developed to provide a concurrent Message Queue interface to the TL1 interface of the Glimmerglass Optical Switch. Table 9 shows the array of test bed components on the Message Queue and the type of messages which they publish and to which messages they subscribe.

Subsystem	Location	Topic	Direction	Purpose
TCD Openflow controller	TCD_ONA controller	NetEvent	Publish	Interpret failure signal from PON and broadcast event on Message Queue
TCD Openflow controller	TCD_ONA controller	GlimEvent	Publish	Trigger Primary or Secondary Path in Optical Switch
Testbed Controller	TCD_ONA controller	TestControl	Publish	Broadcast signal for stop, start, restart of test cycles.
GÉANT Controller	GÉANT Openflow controller (NL)	NetEvent	Subscribe	Execute secondary routing in network core.
Data Logger	TCD_ONA controller	TestControl	Subscribe	Aggregate and format results of test events throughout Test bed.
Data Traffic Gap Measurement Sender	GÉANT DataCentre (DE)	TestControl	Subscribe	Ascertain when test scenario has started or stopped, and which one
Data Traffic Gap Measurement Receiver	TCD_ONA Testbed	TestControl	Subscribe	Ascertain when test scenario has started or stopped, and which one
Optical Switch (Glimmerglass System 100 16 port)	TCD_ONA Testbed	GlimEvent	Subscribe	Execute Optical Switch path selection

Table 9 - Association of Message Queue types and testbed components

5.2.3 Results

Figure 23 shows the end to end N:1 dual homed protection time of the LR-PON and SDN core over 50 experimental iterations, using the initial break in the fiber as a reference point. The figure also shows the timing of various events that occur during the protection switch for all 50 experimental iterations.

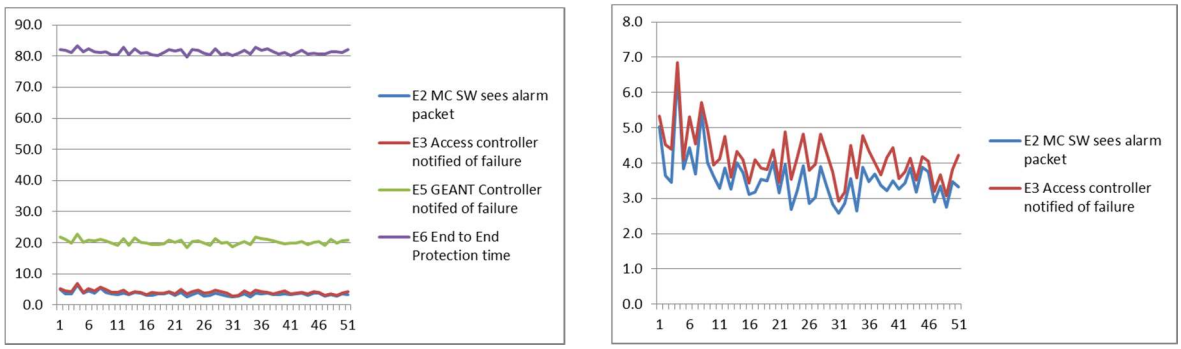


Figure 23 - Switchover time (milliseconds) for 50 iterations of N:1 protection

Since the trigger failure event was issued to the FPGA board over a UART it was not possible to read an absolute time value from the FPGA boards for when the break in the primary fibre occurred. However, we were able to work back from the restoration point of traffic by subtracting the outage duration within each cycle. On average, the alert that identifies loss of the primary PON (the E2 event in the figure) occurs 3.5 milliseconds after the break. The Openflow controller within the TCD ONA testbed sees the alert 0.59 milliseconds (E3) after this and publishes a NetEvent failure alert as well as a GlimEvent event. The NetEvent alerts the GÉANT controller to invoke the alternate path through the core. The GlimEvent event invokes the secondary path in the optical switch. Within this experiment, the GÉANT controller sees the NetEvent event 20.3 milliseconds after the initial failure (E5). Separately, we have measured the asynchronous switchover of the Glimmerglass optical switch as 23 milliseconds. Overall, Restoration time of the data traffic is measured as 81.29 milliseconds.

5.3 N:1 Protection Scheme with PON Physical layer

5.3.1 Configuration

For the N:1 Protection with Physical Layer experiment [117], we implemented end-to-end software defined networking (SDN) management of the access and core network elements of a time-division multiplexing (TDM) dense wavelength division multiplexing (DWDM) long-reach passive optical network (PON). The physical layer in Figure 24 demonstrated co-existing heterogeneous services and modulation formats such as residential 10G PON channels, business 100G dedicated channel and wireless front haul on the same long reach TDM-DWDM PON system. This worked with both erbium doped fibre amplifiers (EDFAs) or semiconductor optical amplifiers (SOAs) for a TDM-DWDM PON up to 100km reach, 512 users and emulated system load of 40 channels.

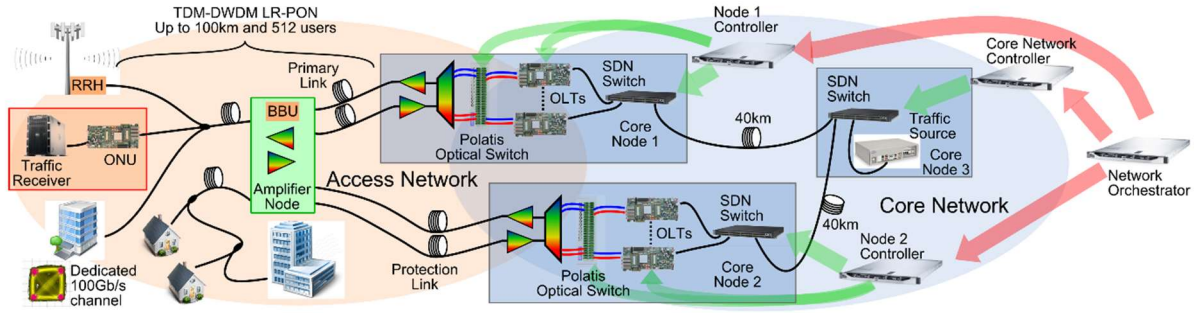


Figure 24 - Network level view of the demonstration

The Use case shown in Figure 25 exemplifies how path integrity in the Core and TDM-DWDM LR-PON based Access Metro network of a Telecommunications network could be assured through logical protection. The protection experiment demonstrated a dual-homed LR-PON protection mechanism where backup OLTs are shared among PONs in an N:1 scheme [107] and the service restoration is provided over an end-to-end SDN. The system carried out an initial phase of path-precomputation, where it sets up a backup path associated to the failure of a specific PON. The pre-calculation considers the input and output ports at the optical switch, the flow table configuration of the OF SDN switch (both access and core) and the configuration of the OLT flow table. The Failure Event (1) was caused by the feeder fibre between the primary OLT and first stage splitter on the PON being cut. This stops all upstream and downstream data on the Primary link. A hardware unit in the primary OLT FPGA monitors the upstream data path.

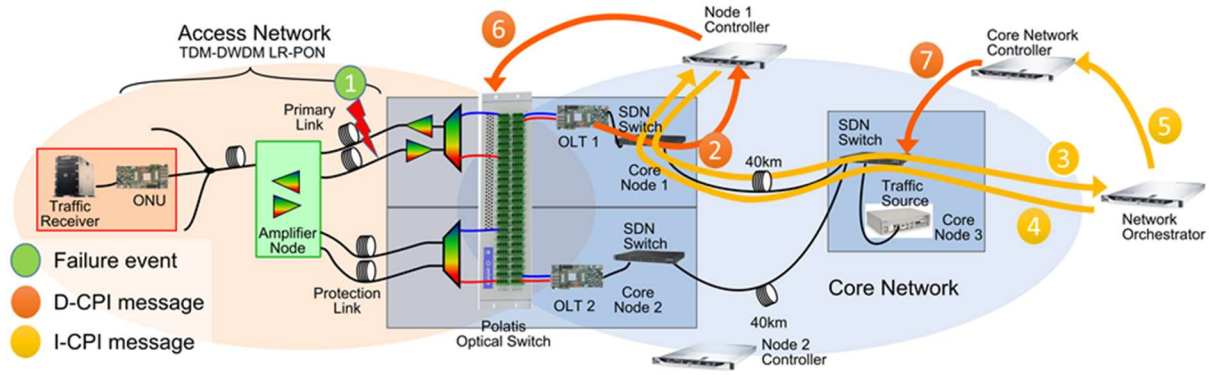


Figure 25 - Protection Experiment

5.3.2 Results

The first test of the protection experiment was the failure event emulated by using the optical switch to simulate a fibre cut in the backhaul fibre link between the primary OLT and the Access Network. Silence in the upstream activated a countdown timer in the primary OLT, which on expiry generated a failure detection and an in-band alarm to the controller of node 1. The duration of this timer took into account all normal silences on the PON due to the

1.25 milliseconds quiet windows and 1.25 milliseconds roundtrip time over maximum distance supported by the protocol of 125km, for a total of 2.5 milliseconds. The node 1 controller then alerted the overarching Network Orchestrator which calculated a path to restore services to the ONUs according to its knowledge of the full end-to-end topology covering the core and access networks. The Network Orchestrators were then instructed by the Network Orchestrator to provision the protection path through the backup OLT. Figure 26 shows a capture of the message flow for one of the protection experiment runs.

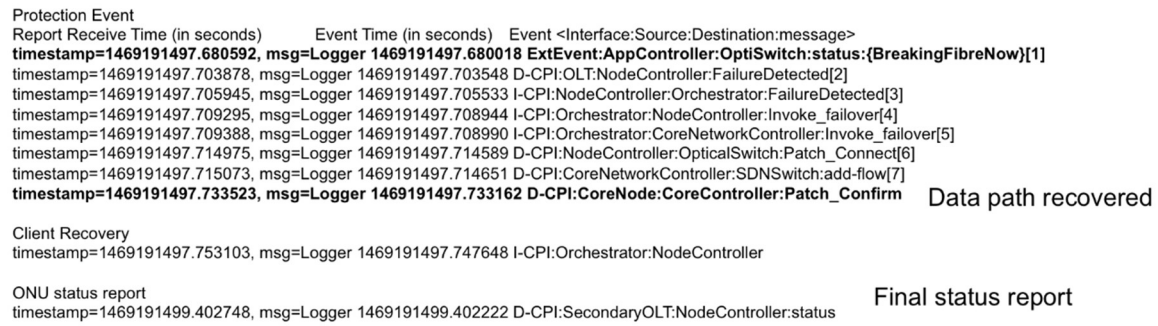


Figure 26 - Protection Message Flow

Figure 27 shows the service restoration time for the protection mechanism conducted where backup OLTs are shared among PONs in an N:1 scheme [117]. The baseline time between the two paths of approximately 50 milliseconds is given when the switchover is proactively triggered by the controller, without waiting for a failure event. In contrast, the protection results show the restoration time when a failure event is caused by a cut in the backhaul link between the primary OLT and the Access Node. Silence in the upstream activates a countdown timer in the primary OLT controller, which on expiry generates a failure detection and an in-band alarm to the Openflow access Network Controller. The access Network Controller alerts the Network Orchestrator, which provisions the protection path. The average protection time is measured at 64 milliseconds, with variations between 50 and 100 milliseconds attributed to the random delay in the failure detection.

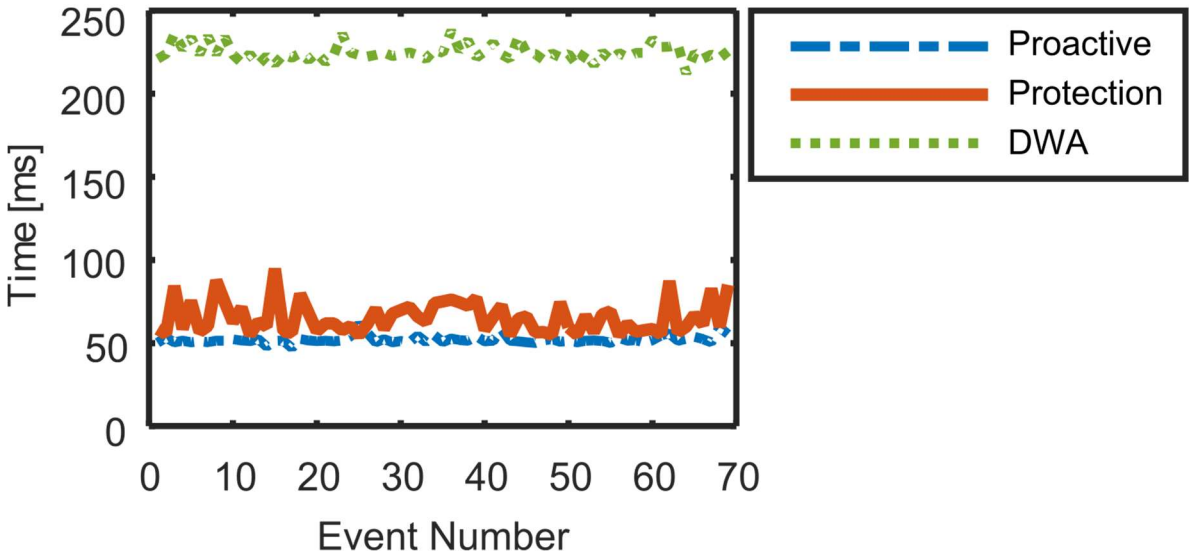


Figure 27 - Protection Timings

The N:1 + physical layer experiment was executed secondly [118], where the detection response was optimised. The results of the service restoration time for the SDN control plane based protection mechanism are shown in Figure 28. The average restoration time over 70 measurements was 41 milliseconds.

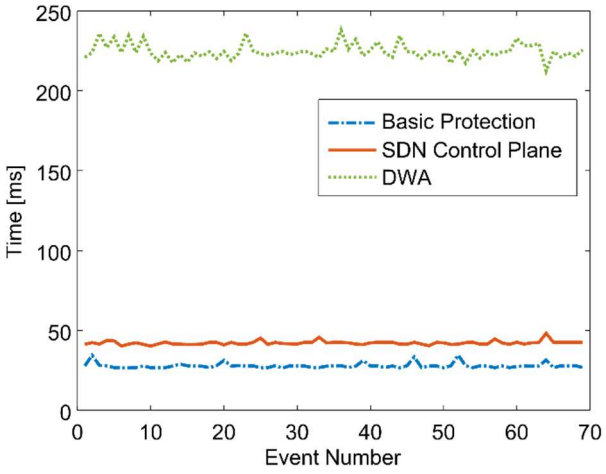


Figure 28 - Service restoration time for the protection mechanism and the DWA through the implemented SDN control plane

In Figure 29, we show the breakdown of the various timings that comprise the 41 milliseconds protection figure. The hardware monitoring at the OLT can detect a failure in the network in about 2.5 milliseconds. A further 1 milliseconds is taken for the alarm packet

to be created and sent to the Core Node switch. The time needed by the protocol to re-establish downstream synchronization is between 2 and 3 milliseconds.

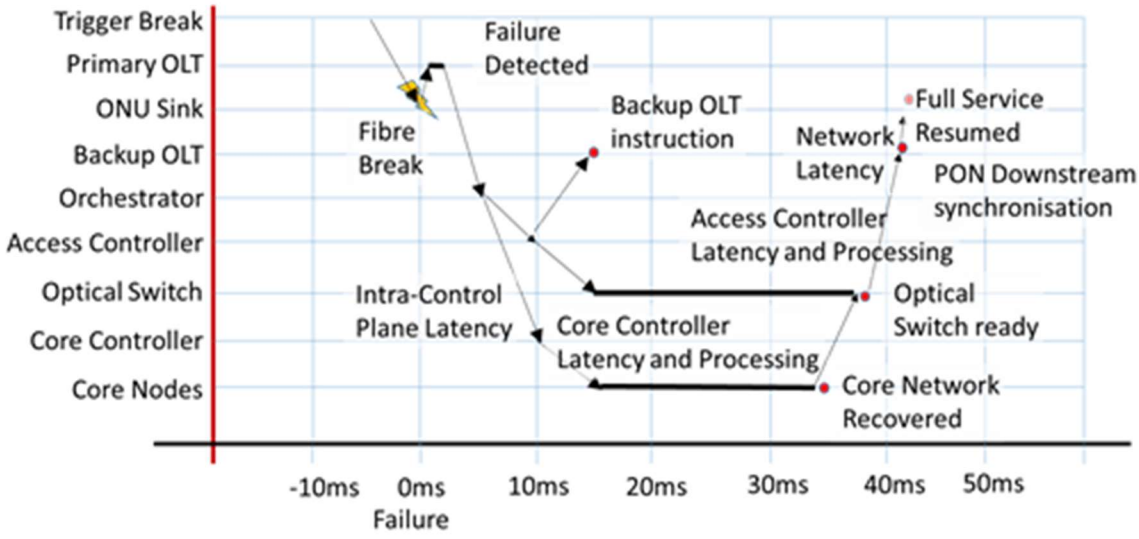


Figure 29 - Timings Trace

We know that some time may be needed to re-range the ONUs in addition to the synchronization time (between 2 and 4 milliseconds), however in this work we assume that ranging to the backup OLT can be done during normal operation of the PON [116]. Intra-control plane communication is done through a dedicated network with typical latencies. The network latencies between both the OLT and the Network Orchestrator and the Network Orchestrator and the Network Controllers are emulated in the test-bed and set at 4 milliseconds each. The latency and the processing times for both the Network Controllers is also emulated as 5 milliseconds each. The core network recovery happens in parallel to the access network recovery time.

Accordingly, within 15 milliseconds of the failure, the optical and electronic switch components and the backup OLT have been instructed to reconfigure their protection paths. Within 33 milliseconds after the failure, the electronic switch components within the core and access are configured, and by 38 milliseconds, the optical switch component is configured. In order to understand the effect of centralising both the Network Orchestrator and the Network Controllers, we compared the above results with the case where orchestrator and controllers are collocated within the Core Network. This was accomplished by setting the emulated intra-control plane latencies at zero. The results are shown in Figure 28 as the basic protection line. On average, basic protection can be accomplished within 27.8 milliseconds.

5.4 Summary

In this chapter, we have been able to demonstrate, progressively more complex scenarios for the protection of converged architecture networks. In our first 1:1 protection experiment

(see section 5.1) across a pan-European network, full recovery took place over an elapsed time period of 124 milliseconds. From analysis, this was composed of 3 individual time periods - a period in which traffic in the access was failed over from the primary path to the secondary path (7.2 milliseconds); a period in which core traffic was being redirected before the service could be restored (25 milliseconds); an intervening period in which the end to end link was in flux (92 milliseconds). We optimised the failure detection mechanism in our first N: 1 experiment (section 5.2). Overall, Restoration time of the data traffic improved and was measured at 81.29 milliseconds.

In our second N: 1 experiment (see section 5.2), we included a PON physical layer with backup OLTs shared among PONs in an N:1 scheme. The average protection time was measured at 64 milliseconds, with variations between 50 and 100 milliseconds attributed to the random delay in the failure detection. In our third N:1 experiment, we optimised the failure detection response again and achieved an average restoration time of 41 milliseconds across 70 measurements.

Chapter 6 Converged Architecture DWA

On a shared transmission pipe, all traffic flows contend for the common bandwidth. Capacity on Demand allows a dedicated portion of bandwidth to be allocated to a flow or group of flows, for a period of time. The capacity may be requested in real-time or semi-real-time by an end user. Typically, a user selects through a portal, the source and destination for the transmission of traffic and the bandwidth required. The portal is a front-end to the network orchestrator. Type traffic types include Video-On-Demand (VOD) and Bandwidth-on-Demand (BOD). In Figure 30(a), we give an example of Capacity that is assured across the Openflow switch, where the useful traffic must contend with best effort background traffic. The useful traffic has a fixed commit Information Rate (CIR) on the egress port of the Openflow switch which cannot be compromised by the Best Effort traffic. Separately, the traffic can peak to a Peak Information Rate which exceeds the Committed Information Rate, however, this additional bandwidth is contended. This solution is adequate where there is sufficient bandwidth across the PON.

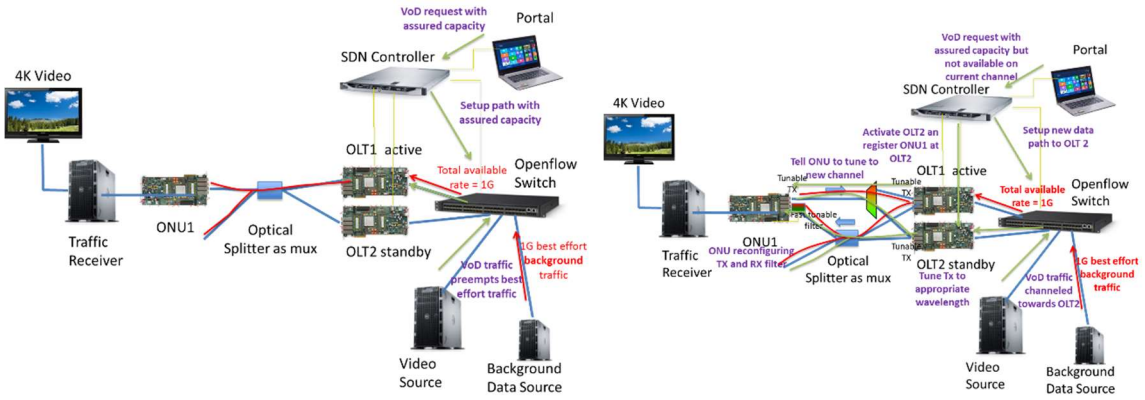


Figure 30 – (a) VOD with Assured Capacity. (b) Assured capacity on new channel

However, there are situations where some level of guaranteed bandwidth can be provisioned across the PON, in addition to the assured bandwidth in the metro core network. In Figure 30(b), capacity is assured across the PON through the allocation of a dedicated wavelength in a real-time. This Dynamic Wavelength Assignment (DWA) use case exemplifies how capacity constraints in one PON channel may be overcome by re-allocating one or more end user ONUs dynamically to a different channel in order to assure quality of service. This could also be used for the opportunistic provision of high bandwidth services (on-demand video and big data transfers), to specific PON users on a dynamic basic. Since the DWA use case is aimed at capacity provision, the wavelength and service reconfiguration times targeted are in the region of a few hundreds of milliseconds. While wavelength assignment is not carried out at the granularity of individual burst transmission, dynamic wavelength assignment is still being invoked since the change in wavelength is dynamically and automatically allocated by the controller as a response to an increase in user capacity, rather than being statically assigned by the network management plane.

6.1 Assured Capacity on a new Channel

In the cases of temporary and dynamic migration of wavelength it is important that the wavelength switching time is minimized to avoid impacting users of other network services. Typically, an end user (customer) elects through a portal frontend to transmit a fixed bandwidth transport between two end points typically for the sending and receipt of video streaming. The Network Orchestrator orchestrates the provision of the path, according to its knowledge of the full End-to-End topology covering the Core and Metro Access Networks. The core Network Controller and the Metro Access Network Controller, are each instructed to provision an explicit path respectively. The SDN Controller (Figure 9) provisions the path through the Metro Access Openflow switch, and the PON. Using a custom implemented PLOAM message, the primary OLT requests the ONU tune to a wavelength provisioned out of the secondary OLT. The DISCUS SDN Network Controller, acknowledges that it has completed the provisioning of the path through the Metro Access portion of the network.

Practically, Dynamic Wavelength Assignment was implemented through the addition of laser and filter control to the LR-PON protocol hardware and control mechanisms. The tuneable laser was controlled across an i2c bus to the Skylane 10G SFP+ tuneable lasers and the tuneable filter was controlled through a UART. To implement DWA in the physical layer, we employed a splitter and filter in the downstream, and an AWG and Polatis Optical Switch in the upstream direction, statically patched to ITU channels 33 and 32 of the primary and secondary OLT respectively (Figure 31). In order to select the OLT and ONU transmission wavelengths, the OLT provided a North Bound interface. Through this Interface, the control plane could tune an OLT's transceivers to a given wavelength. Since the ONU was remote from the control plane, tuning of an ONU's laser and filter was performed also through this interface by the invocation of custom PLOAM message within the LR-PON protocol. The wavelength of the OLT and ONU were selected by writing to control registers in the OLT. Each individual OLT laser wavelength could be set by writing the ITU channel number to its local register. To select the wavelength of transmission for an ONU, the ITU channel number was set by writing the target ITU channel number to register of the OLT to which it is homed. The ONU Id was also specified so as to distinguish an individual from multiple ONU's homed off a single OLT.

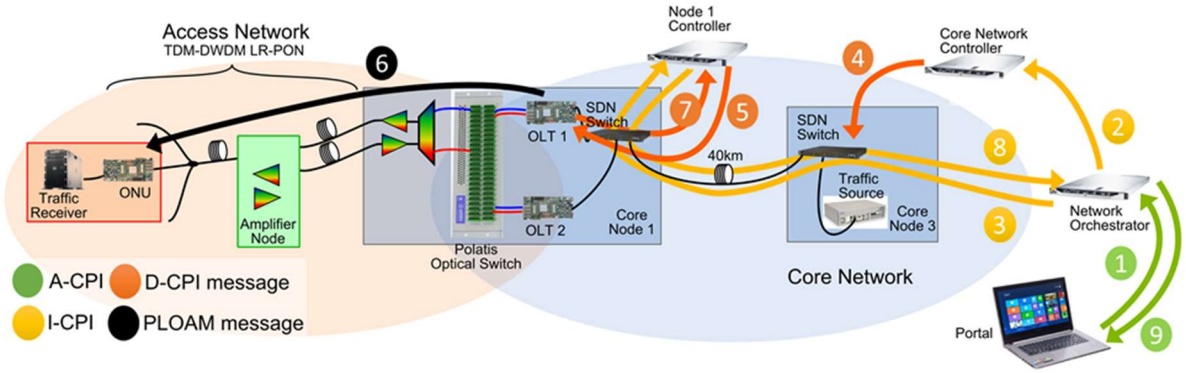


Figure 31 - DWA Scenario

6.2 Results

The DWA results in Table 10 refer to the service provisioning time when, in response to an increase in traffic demand, the Network Orchestrator instructs the core and the access Network Controller (using the control plane messages in Table 7) to provision the new path. The Network Orchestrator uses its knowledge of the full end-to-end network topology to instruct the ONU traffic to move to a different PON channel. In this instance, is from the probing Using a custom implemented Physical Layer Operations, Administration and Maintenance (PLOAM) message, the primary OLT requested the ONU to tune to a wavelength provisioned by the secondary OLT. On inspection, we believe that the measured provisioning time of about 225 milliseconds (from probe event **4.a** to the probe event **6.**a) could be reduced by an optimized design of communication interfaces between the ONU FPGA and the tuneable components.

Timestamp	Type	Source	Destination	Message	Event Number
1458130894.036	A-CPI	Portal	Orch.	Resouerce_Request	1
1458130894.243	I-CPI	Orch.	CNC	Create_Path	2
1458130894.251	I-CPI	Orch.	NC	Create_Path	3
1458130894.265	D-CPI	CNC	SDN_switch	add-flow	4
1458130894.300	**MEAS**	**---**	**---**	**Traffic_Measurement_Break**	**4.a**
1458130894.334	D-CPI	NC	OLT	Set_DS_lambda	5
1458130894.402	PLOAM	OLT	ONU	Set_Wavelength	6
1458130894.521	**MEAS**	**---**	**---**	**Traffic Restored**	**6.a**
1458130894.702	D-CPI	OLT	NC	Status_Report: onu_registration	7
1458130894.704	I-CPI	NC	Orch.	Create_Path_Confirmation	8
1458130895.005	A-CPI	Orch.	Portal	Resource_Confirmation	9

Table 10 - DWA timings

6.3 Interworking between DISCUS and IDEALIST testbeds

This work defined and demonstrated an SDN architecture to support end-to-end service from access to core. The validation work had involved two laboratories that demonstrated that network operators could deploy SDN solutions that covers, not only access or core scenarios, but also end to end.

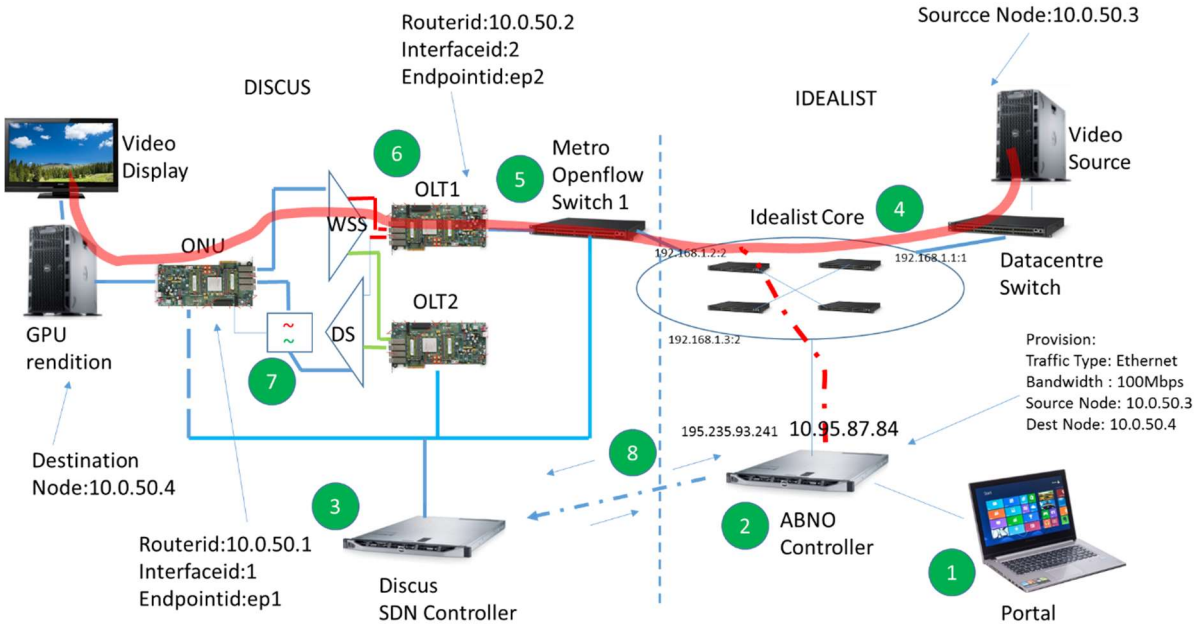


Figure 32 - Experimental Lab Set up

There were two different labs set-up to demonstrate the scenario of this work, one lab in Telefonica premises and another in Trinity College of Dublin. Figure 32 shows the schema of the lab set-up for this experiment. To communicate between them, a VPN was created, so the data plane connection had a low bandwidth. The network orchestrator was located in Telefonica labs and was based on netphony ABNO implementation. The north and south-bound interface of the orchestrator was implemented using the STRAUSS COP. The core controller used the netphony ABNO, which in addition with netphony PCE controlled a GMPLS emulated control plane. The GMPLS nodes used the protocol suite developed by Telefonica I+D and is released in GitHub. This setup was built with 30 virtual machines, which run in a Linux server distribution. Each emulated node implemented a GMPLS stack (including RSVP, OSPFv2 and PCEP) with the extensions to support flexgrid developed in IDEALIST project. The PON backplane connection to the core network contained a 10G Ethernet physical layer and Media Access Control Layer, allowing it to be plugged into any 10G capable network element.

To implement DWA in the physical layer, we employed a splitter and filter in the downstream, and a WSS (Optimum 9x1/1x9 50Ghz Wavelength Selective Switch) in the upstream direction, statically patched to ITU channels 32.5 and 31 of the primary and secondary OLT respectively. In order to select the OLT and ONU transmission

wavelengths, the OLT provided a North Bound interface. Through this Interface, the control plane could tune an OLT's transceivers to a given wavelength. Since the ONU was remote from the control plane, tuning of an ONU's laser and filter could be performed also through this interface by the invocation of custom PLOAM message within the LR-PON protocol. The wavelength of the OLT and ONU could be selected by writing to control registers in the OLT. Each individual OLT laser wavelength could be set by writing the ITU channel number to its local register. To select the wavelength of transmission for an ONU, the ITU channel number was set by writing the target ITU channel number to register of the OLT to which it is homed. The ONU Id was also specified so as to distinguish an individual from multiple ONU's homed off a single OLT. For the experimental LR-PON access network, the configuration was comprised of a Pronto 3780 switch with 48 10G interfaces, running release 2.4 (Openflow v1.4 compatible firmware). The Pronto switch was configured with multiple virtual bridges. A Video Server (VLC) application was co-located with the ABNO controller interface in the Telefonica premises. This transmitted a UDP based video stream across the Tunnel between the two testbeds, traversing the DISCUS PON and was received by the GPU workstation for display by the TV display.

6.4 Results

The latency between the two testbeds was measured over 100 measurements at between 45 and 48 milliseconds. It was not possible to transmit the video stream through the IDEALIST network, as a physical data path was not available. In Step 1, an end user (customer) elected through a portal frontend to the ABNO controller to transmit a fixed bandwidth transport (100Mbps) between two end points aEnd (10.0.50.3) and zEnd (10.0.50.4) typically for the sending and receipt of Video streaming. In Step 2, the ABNO orchestrated the provision of the path, according to its knowledge of the full End to End topology covering the Core and Metro Access Networks. The IDEALIST core Network Controller and the Metro Access (DISCUS) SDN NC, were each instructed to provision an explicit path in steps 3 and 4 respectively. For the Metro Access portion of the network (the path from 10.0.50.2 to 10.0.50.4), the DISCUS SDN Network Controller was instructed to provision the path according to the route 10.0.50.2 to 10.0.50.1 to 10.0.50.4. The DISCUS SDN Controller provisioned the path through the Metro Access (Openflow) switch (step 5), and the PON (primary/secondary OLT and ONU – step 6). Using a custom implemented PLOAM message, the primary OLT requested the ONU tune to a wavelength provisioned out of the secondary OLT (step 7). In step 8, the DISCUS SDN NC, acknowledged that it had completed the provisioning of the path through the Metro Access portion of the network. In step 9, the Video transmission was triggered to start.

First, the portal requested a new video service, which could not be processed within the access area scope. This meant that there was a request from the video platform to provision

an end to end path between the client and the video server. Therefore, the STRAUSS ABNO received a COP calls service set-up to establish the connection. The STRAUSS ABNO carried out a path computation, which crossed different networks, the core (IDEALIST) and access network (DISCUS). Therefore, the STRAUSS ABNO sent a COP calls service set-up to each controller to configure the nodes in their domain. The IDEALIST PCE configured the GMPLS nodes, while the DISCUS controller configured the access elements. The workflow is explained in Figure 33.

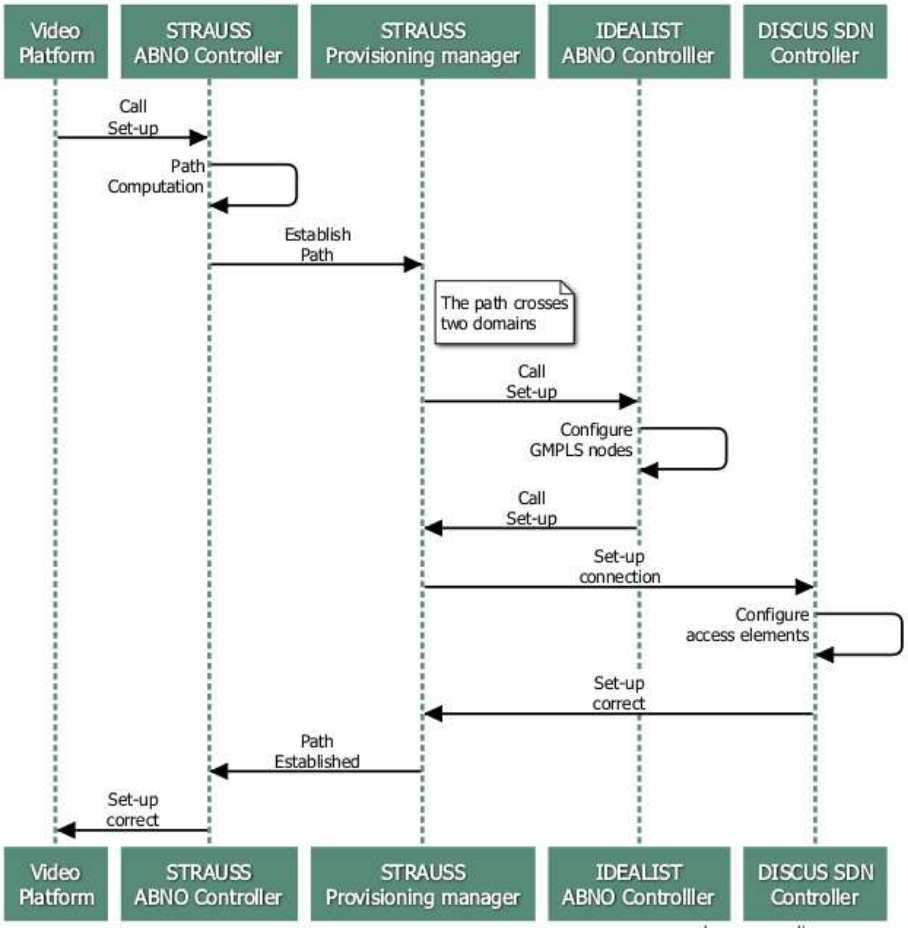


Figure 33 - Workflow Steps

Figure 33 shows the message exchange between the different elements.

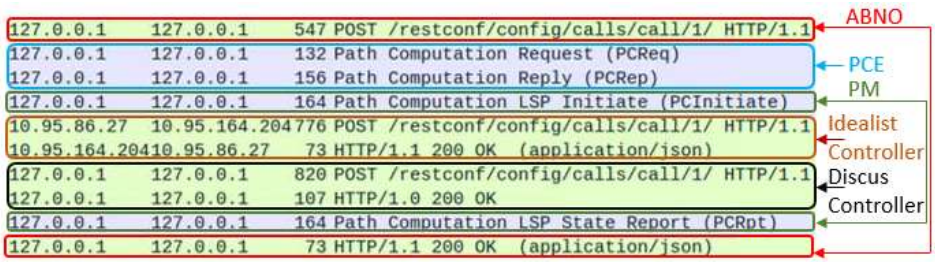


Figure 34 - Whireshark capture

As it is shown in Figure 34, the ABNO received an HTTP POST request with COP syntax. Figure 35 shows the JSON object with the request parameters. The aEnd and zEnd routerIds identifies the client and the video server. The traffic parameters were set to request

a 100Mbps connection and a 100 milliseconds latency. This request was sent as an Ethernet service.

Once the STRAUSS ABNO received the requests, it requested its PCE for a path computation between the two end points. To do so a PCReq-PCResp process was performed. Now, the PCE could calculate the path and response to ABNO with a PCResp, which contained the Explicit Route Object with the path. The ABNO controller with the ERO information call to Provisioning Manager (PM) via a PCInitiate message. The PM split the route in different domains and with a COP message call to each controller to create a path in each domain (IDEALIST and DISCUS). When the path was created each controller sent respective http message with an OK status. With this information PM response to ABNO controller with a PCEReport message and finally ABNO report to video platform with an HTTP response.

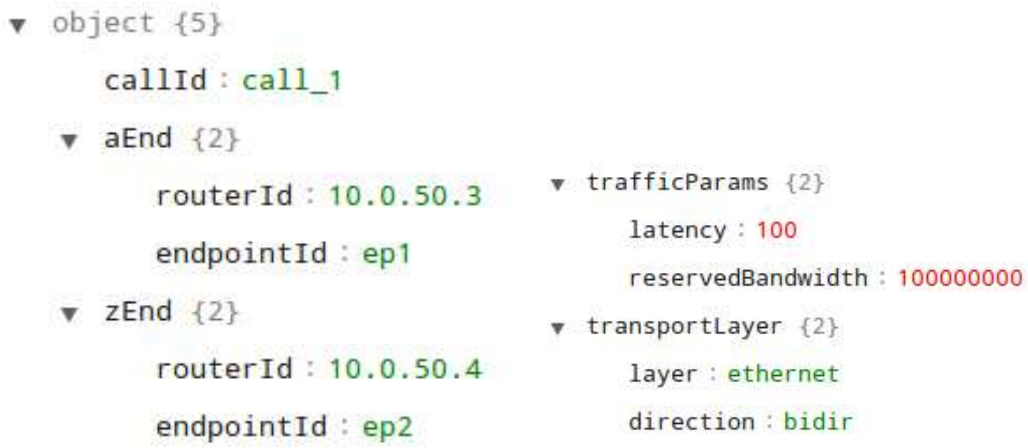


Figure 35 - JSON object for a COP service-call set-up

Across 10 repetitions of the experiment, the total completion time of the workflow was measured at 275 milliseconds, of this, 35 milliseconds (with associated inter-testbed latency) related to the blocking element of the call to the DISCUS SDN Controller. The non-blocking elements of the DISCUS SDN proceed in parallel with the completion of the return calls by the ABNO controller.

6.5 Summary

The Dynamic Wavelength Assignment (DWA) use case exemplifies how capacity constraints in one PON channel may be overcome by re-allocating one or more end user ONUs dynamically to a different channel in order to assure quality of service. We executed this scenario twice. In the first instance, we provisioned DWA over a PON physical layer, and in the second instance, DWA provisioning on the DISCUS testbed was trigger from the IDEALIST ABNO orchestrator. Using a custom implemented physical layer operations, administration and maintenance (PLOAM) message, the primary OLT requested the ONU to tune to a wavelength provisioned by the secondary OLT. In the first scenario, we

0.

measured the total provisioning time as 225 milliseconds. On analysis, we identified much of this time taken up with latency between the interfaces of the FPGA and the controller. We believe that this time could be reduced by an optimised design of communication interfaces between the ONU FPGA and the tuneable components. Across 10 repetitions of the experiment in the second scenario, the total completion time of the workflow was measured at 275 milliseconds, of this, 35 milliseconds (with associated inter-testbed latency) related to the blocking element of the call to the DISCUS SDN Controller. The non-blocking elements of the DISCUS SDN proceed in parallel with the completion of the return calls by the ABNO controller.

Chapter 7 Performance Evaluation - NSIM

NSIM is a distributed, multi-layer network simulator, which has been developed from the ground-up using a semi-interpretive programming language. The intention of nsim is to overcome restrictions of other network simulators (most notably NS3) and to leverage the wider tools and libraries available to commonly used languages such as Java, Python and JavaScript.

The key functional requirements of any Network simulator are the ability to configure network topologies with typical components such as switches, routers and network links, to support for network protocols, to configure and run scenarios at simulation speeds and to produce results such as timings and network traces equivalent to those produced from live experiments.

NS3 is the most common open-source simulator used in particular for network domain problems. It is written entirely in C++ and requires development of models in C++ also. The NS3 simulator is monolithic in that it must run on a single host environment, so that the computing resources such as memory, CPU, disk and I/O available to simulations are limited to those available on a given machine. To run larger simulations requires execution on a host with more resources.

The primary design requirements of NSIM are

- It should be open and extendible simulation framework that is flexible to support any variations of standard and non-standard network stacks and topologies.
- It should have support for standard network protocols such as Ethernet, IP, TCP and UDP but also fractional layer protocols such as PPP, MPLS, Dot1Q, and PPPoE. Support should include the encapsulation as well as the state management aspects of the protocol.
- it should support for network characteristics such as buffering and latency

The secondary design requirements of NSIM are that

- It should support simulations to run across multiple hosts, and thereby leverage the computing resources, such as CPU, memory, disk and I/O of the constituent machines. It should also run adequately on a single machine.
- It should support access to other python computational, and analytical tools such as **twisted, numpy, scapy and octave**. **Octave** is an open source variant of **Matlab**.
- It should allow access to other computational resources such as GPU cores, in-memory and distributed databases

When trying to simulate the FLATLANd architecture in NS3, we identified a number of shortcomings. NS3 has a restricted array of network protocol stacks. While NS3 provides some basic helper functions to create wired networks – **PointToPointHelper**, **CsmaHelper** and **bridgeHelper**, these require full stacks to be installed on each terminating and intermediate nodes. The NS3 Openflow switch module it mandates the use of CSMA

interfaces on each of its ports [119], the effect of this is to impose CSMA limitations and characteristics on the interfaces. There is currently no switched Ethernet models available that simulate Ethernet packet level switching between ports on an Ethernet switch, and which use a non-CSMA physical layer. Albeit there have been a number of attempts to create a basic Ethernet switching model. One uses the half-duplex csma channel type, and nests two csma-net-devices (rx and tx) inside a single ethernet net device. Another is derived from point-to-point links with some of the protocol-specific parts (header processing) from the csma model. The work on advancing the Ethernet Switch module by the Ns-3 community has stalled since 2014[120].

7.1 Generic Functionality

The NSIM scheduler (Figure 36) maintains the simulation clock, defaulted to a granularity of 1 millisecond. The scheduler negotiates locking and unlocking of the simulation clock with the constituent Processes. A Process is any component which must function at simulation speed either at one or multiple clock ticks. The scheduler is, and indeed any Process maybe, a network based function, allowing the simulation to run across multiple machines. Depending on the complexity of the simulation, a simulation clock tick may endure from a fraction of a second up to minutes or hours in real-time. Functionality available to components based on the Process component include waittick() which locks the component until the next clock tick, waitfor(n) which locks the component for n clock ticks, lock()/unlock() which prevents the simulation clock advanced for a period of execution by the component. The simulation clock cannot advance until all components are unlocked. Other functionality includes wait10mstick(), wait100mstick() and waitsectick() which locks the components for 10 milliseconds, 100 milliseconds and 1 second respectively. The waittick() functionality is most used in loops to schedule events at specific intervals, for example, transmitting packets.

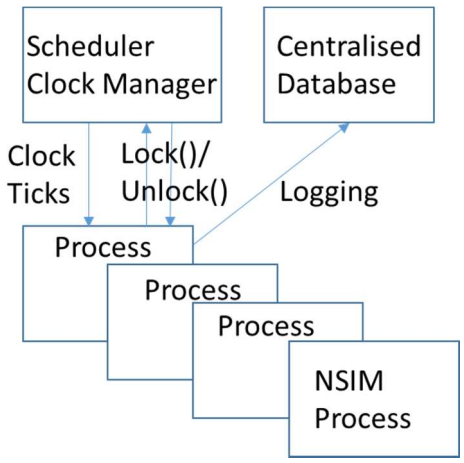


Figure 36 - NSIM Scheduler and Distributed Processes

Fundamental to the simulation of network stacks and network links are Queues, which allows data to be exchanged between components using standard set() and get(). A

maximum size can be set on the size of the queue. NSIM queues are stored in a distributed database, so they are accessible from NSIM components which are located on a different machines. There are three variants of Queues , a standard Queue which can emulate a buffer, a latency Queue which can emulate transmission latency delays with a granularity of 1 milliseconds, and an aged Queue which manages the age of entries in the queue.

A Connector connects Queues which maybe constituents of higher level modules, and because it is a threaded variant of a Process, it operates asynchronously and at simulation speeds. It implements two pieces of functionality key to network simulation – most notably behaviour when Queues (or buffers fill-up) as well as the control of the rate at which data is transferred, that is rate limiting. The Connector allows data that is being transferred between connected queues to be inspected (the inspect() function), which again is key to higher level functionality such as network packet analysis. A Connect functional block cross-connects two interfaces, each of which is implemented using a Connector.

A Duplex is a Process with two interfaces (A and B) which allows bidirectional transfer between the two interfaces, and is a super-class for network layers or network transmission links. Each interface has a transmit and a receive Queue. An interface can be configure to behave in one of two ways when its buffers reach saturation. They either behave in a **lossy** manner in which case packets are dropped, or they can behave **lossless**, in which case the packet is not accepted for delivery. We use the term **back-pressure** to describe the effect on downstream queues. Back-pressure can have a domino effect on a downstream chain of lossless queues. In this instance, the downstream queue that has large buffer space, or is lossy bears the onus for the domino effect.

Two connectors are used to cross-connect the interfaces. Rate limiting, Maximum Queue size, latency and inspection functionality can be specified when the Duplex is configured.

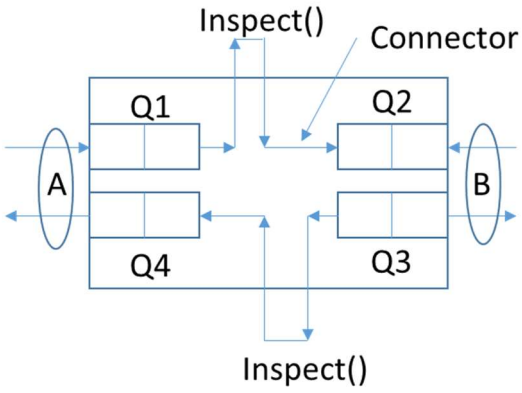


Figure 37 - Duplex Process

Stack and Hub network paradigms can be implemented respectively by interconnecting multiple Duplex blocks using Connect functions, or connecting multiple Duplex blocks using a 1-to-many Connector.

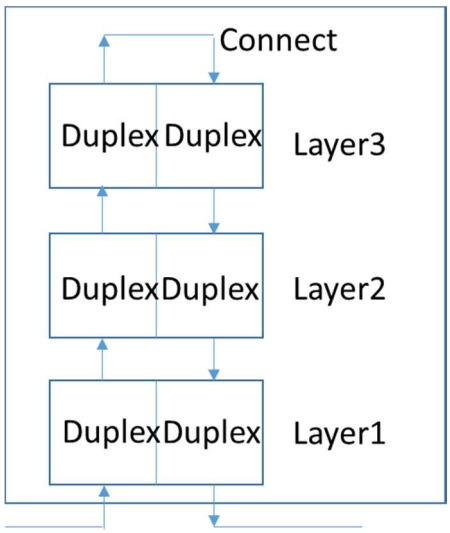


Figure 38 - Stack

A Traffic Generator is a Process which generates traffic with one of three profiles – a single packet, a stream of fixed length packets every n milliseconds or a stream of traffic at a rate of N Mbps. The traffic generator connects to a network host or to another network block and can also receive traffic that is returned from an upstream host. This allows packets returned to be compared with traffic transmitted, and thereby calculate packet loss and packet delay.

Figure 39 uses a simulation clock period of 1 millisecond, and lasts for 10 seconds. The traffic generator trafgen generates 200 byte packets at a rate of 1 every millisecond. To generate 1 Mbyte of traffic, the parameter ms1 is replace with the parameter capacity=1. Term2 is the received node, which accepts and returns traffic which it receives. In sub-scenario 0, the nodes traf and term2 are connected using a simple connect block. Sub-scenario 1 connects the nodes through a third duplex block. In sub-scenario 2, the nodes are connected through 2 x 3-layer stacks (stack1 and stack2). These stacks simply transfer data up and down with modifying the data. The data is not encapsulated with any network protocol. Once the nodes, links and topology are defined, the simulation is started for the specified duration.

```
scenario =0
sched=scheduler(tick=0.001,finish=10)
traf=trafgen('traf',ms1=1)
term2=terminal('term2')
if scenario == 0:
  connect('con1',traf.B,term2.A)
elif scenario == 1:
  node1=duplex('node1')
  connect('con1',traf.B,node1.A)
  connect('con2',term2.A,node1.B)
```

```
elif scenario == 2:
  stack1 = stack('stack1')
  stack2 = stack('stack2')
  connect('linkA',traf.B,stack1.A)
  connect('linkB',stack1.B,stack2.A)
  connect('linkC',stack2.B,term2.A)
sched.process()
```

Figure 39 - NSIM example scenario A

In Figure 40, A Flow Generator creates a set of parallel running flows, with number flowcount. There is an initial start time and a flow duration. An interval specifies the gap between subsequent flows.

Typically, a flow is used for background traffic which begins and ends. The staggered delay in starting the flows allows this background traffic to ramp up and fall off. Background flows should traverse the network to a specific network termination point. The flow-generator is bound to host3. Destination traffic will have a destination of mdst='00:00:00:00:00:00', which is then dropped by the termination host host2, which drops any traffic with mac address mdrop='00:00:00:00:00:00'

```
sched=scheduler(tick=0.001,finish=10)

host1=host('host1',stack='udp')
host2=host('host2',stack='udp',mdrop='00:00:00:00:00:00')
host3=host('host3',stack='udp',mdst='00:00:00:00:00:00')
#traf=trafgen('traf1',ms1=1)
traf=trafgen('traf1')
term2=terminal('term2')
flowgen=flowgen('flowgen',start=0.002,stop=2.5,ival=0.300,flowcount=5)
# sw=datalink('node1',capacity=1,MaxSize=10000,latency=40)
connect('hostcon1',host1.B,traf.B)
connect('con1',host1.A,sw.A)
connect('con2',sw.B,host2.A)
connect('hostcon2',host2.B,term2.A)
connect('flow',flowgen.B, host3.B)
connect('con3',host3.A, sw.A)

sched.process()
```

Figure 40 - NSIM example scenario B

7.2 Network Functionality

Network functionality is implemented using the fundamental generic functionality described such as Duplex, Connectors and Connects. The **scapy** python module is used for packet crafting.

A host creates a Process based network stack composed of an Ethernet layer, IP layer and either a TCP or UDP layer. For data that is being sent down the stack, data has to be encapsulated with the relevant parameters of each layer (such as Ethernet addressing and IP addressing) in turn. For data that is being sent up the stack, from the network interface, data must be de-encapsulated each layer in turn.

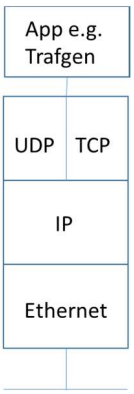


Figure 41 - Host Stack

A Datalink is a subclass of a Duplex block, through which the link capacity in Mbps and Bit Error Rate (BER) of Transmission may be specified. The link capacity translate to the rate limit parameter of the underlying Connector blocks. BER is implemented by inspecting the traffic (using the underlying Connectors inspect() function) then decoding the line traffic at an Ethernet link level. Given the size of the packet and the bit error rate, a packet_drop() function determines True or False to drop the packet at the Ethernet interface level (Equation 1). A packet drop hit is recorded in the database for that interface.

$$\text{Probability of Packet Drop} = 1 - (1 - \mathrm{p})^{n}$$

Equation 1 - Packet drop probability

Where n is the number of bits, and p is the probability of an error.

Separately, the inspect() function writes the Ethernet packet out to a .pcap file if tracing is switched on this datalink. Inspection happens separately for data in the forward and reverse date transmission directions, since there are two separate Connector blocks in use.

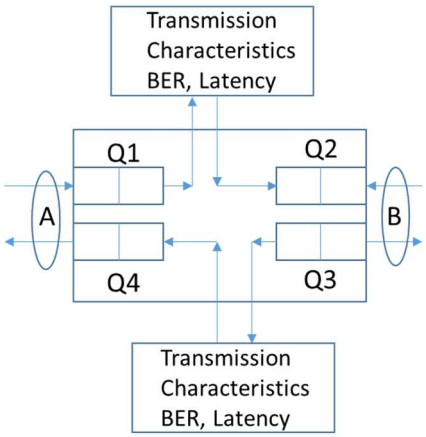


Figure 42 - Datalink - based on Duplex block

An Ethernet switch is a single layer stack, containing two Duplex blocks. Two blocks are required for the forward and reverse data transmission directions. The Ethernet switch has two network interfaces A and B through which data is switched as well as two application interfaces. A Connect block cross connects the application interfaces. Ethernet packets are simply inspected and switched between the A and B interfaces without modifying any characteristics including the Ethernet addressing. A Router However, it routes traffic between the two interfaces and thus modifies the Ethernet packet addressing. A vswitch inserts and drops MPLS, Dot1Q, PPPoE and PPP sublayers. It is similar to the Ethernet switch and contains two Duplex blocks.

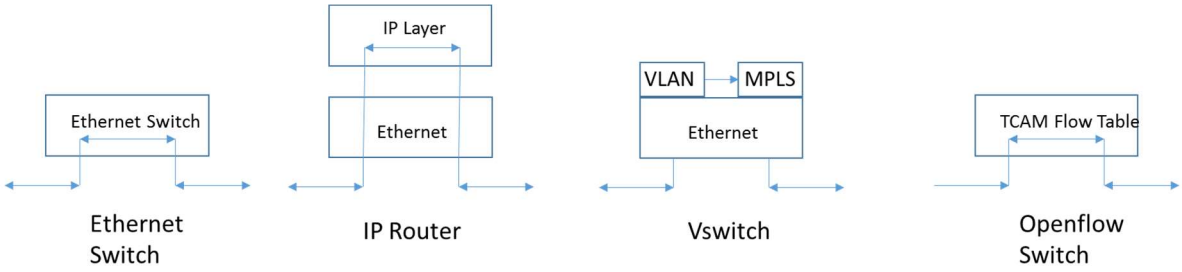


Figure 43 - NSIM switch types

As an example, in Figure 44, two hosts are created, each with a UDP stack (which includes Ethernet and IP layers.) Traffic Generator and Termination points are created. A datalink (dl) is created with a data transfer rate of 1000 bps, and QueueSize of 1000 bytes. Connects are made between terminating points and the stacks, and between the stacks.

```
sched=scheduler(tick=0.001,finish=10)
host1=host('host1',stack='udp')
host2=host('host2',stack='udp')

traf=trafgen('traf1')
term2=terminal('term2')
```

```
 dl=datalink('node1',ratelimit=1000,MaxSize=1000)
connect('hostcon1',host1.B,traf.B)
connect('con1',host1.A,dl.A)
connect('con2',dl.B,host2.A)
connect('hostcon2',host2.B,term2.A)
sched.process()
```

Figure 44 - NSIM example scenario C

In Figure 45, the datalink is configured with a latency of 50 milliseconds and a Bit Error Rate (BER) of 10^-9. PCAP tracing is enabled on this link (to file link.pcap), which may then be read by wireshark or tcpdump.

```
link=datalink('link',latency=50,trace=True,debug=True,ber=-9)
connect('hostcon1',host1.B,traf.B)
dataconnect('con1',host1.A,link.A)
dataconnect('con2',link.B,host2.A)
connect('hostcon2',host2.B,term2.A)
```

Figure 45 - NSIM example scenario D

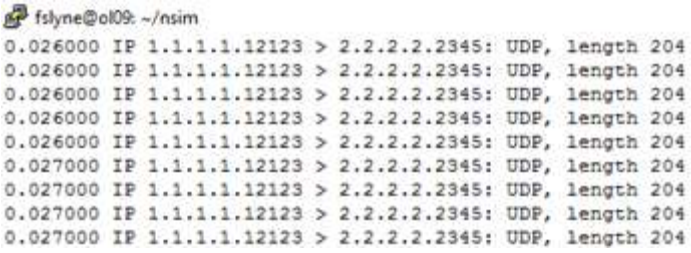


Figure 46 - TCPDump of link.pcap

In Figure 47, the hosts are connected by two vswitches, which uses MPLS as a sub-layer between the switches. In the forward data direction, vsw1 pushes the MPLS label and sw2 pops the label. In the reverse direction, vsw2 pushes the MPLS label and sw1 pops the label

```
sw1=vswitch('sw1',"","MPLS(label=250)")
sw2=vswitch('sw2',"MPLS(label=250)","")
connect('hostcon1',host1.B,traf.B)
connect('con3',host1.A,sw1.A)
connect('con3',sw1.B,sw2.A)
connect('con4',sw2.B,host2.A)
connect('hostcon2',host2.B,term2.A)
```

Figure 47 - NSIM example scenario E

Typically, hardware devices are lossless and trigger back pressure when their fixed buffers become saturated. Hardware devices such as switches, routers and hosts can be configured specifically to have a fixed length buffers and then drop packets when their buffer gets saturated. Since datalinks do not have buffers other than the latency used for the data in transit, they are configured as lossy by default.

7.3 TCP protocol

Implementing the TCP transport layer requires additional end to end functionality, not evident in the packet level transfer for UDP. This end-to-end functionality assures data that is transferred between the end-points, so that the connection is created between the correct end-points but also all data that is transfer is transferred intact, and in an efficient manner possible. TCP is implemented in NSIM in two parts – the TCP network stack and the TCP protocol. The TCP network stack performs the (TCP/IP/Ethernet) encapsulation and de-encapsulation of data for transmission to the network and reception by a remote host. The stack creates a TCP listener and an application level socket through which data may be sent to /from an application, as well as a network (that is, an ethernet) interface to the NSIM (virtual) physical layer.

self.listener = TCPListener(self.A.get,self.A.put,'1.1.1.1') self.conn=TCPSocket(self.listener) self.conn.connect('2.2.2.2',80)	Client end of TCP connection
self.listener = TCPListener(self.B.get,self.B.put,'2.2.2.2' self.conn=TCPSocket(self.listener) self.conn.bind('2.2.2.2',80)	Server end of TCP connection

Figure 48 – NSIM example Scenario F

This process is identical for both local and remote nodes. However, a Server binds to its local socket, while a Client connects to its local socket, which allows the client to make the connection request.

The TCP protocol manages transition between states of the TCP connections (CLOSED,LISTEN, SYN-SENT,SYN-RECEIVED,ESTABLISHED,CLOSE_WAIT, LAST-ACK and FIN-WAIT) as well as the transition between sub-states within the operation of protocol, in particular, related to Congestion management.

In Figure 49, a TCP client (tcpxmit) and a TCP server (tcprecv) are created. They are connect through three switches vsw1, vsw2 and vsw3 by two datalinks link1 and links2 with latencies of 10 milliseconds and 50 milliseconds respectively. VLAN encapsulation is run between sw1 and sw2, and MPLS is run between sw2 and sw3. Sw2 performs de-encapsulation of VLAN and encapsulation of MPLS in the forward direction.

```
sw1=vswitch('sw1',"","Dot1Q(vlan=3)")
sw2=vswitch('sw2',"Dot1Q(vlan=3)","MPLS(label=101)")
sw3=vswitch('sw3',"MPLS(label=101)","")
link1=datalink('link1',latency=10,trace=True)
link2=datalink('link2',latency=50,trace=True)
connect('con3',tcpxmit.B,sw1.A)
connect('con3',sw1.B,link1.A)
connect('con3',link1.B,sw2.A)
connect('con3',sw2.B,link2.A)
connect('con3',link2.B,sw3.A)
connect('con4',sw3.B,tcprecv.A)
```

Figure 49 - NSIM Example Scenario G

In the reference TCP/IP protocol stack, TCP encapsulates the application PDU (protocol data unit) with a TCP header to be transmitted from the upper layer application. The application PDU must be segmented into TCP segments. In turn, the lower IP layer encapsulates the TCP PDU. On the receiving (peer) side, the process of de-encapsulating and interpreting the data happens in reverse. The TCP header holds the meta data such as source and destination ports (16 bits each), sequence and acknowledgement numbers, protocol flags and checksum. The Sequence number is a 32 bits number that represents either the initial sequence number, if the SYN bit is set, or the sequence number of the current packet if the SYN bit is not set. Here, the sequence number of the first data byte will then be one plus the initial sequence number. Similarly, the Acknowledgment number is a 32 bit number that represents the sequence number of the next expected byte to be received from the sender if the ACK bit is set or the acknowledgment of the remote end's initial sequence number itself, if the ACK bit is not set.

A TCP session progresses through three phases. Connections from a client to a server are established through a 3 step handshake process [121] . Once the connection is established, Data is transferred bi-directionally. Connections may be terminated by either client or server through a three or four steps. TCP manages the myriad of states and sub-states which each TCP connection can progress through, using a complex state machine. Within the Connection phase a connection may be in a state of LISTENING, SYN-RECEIVED, SYN-SENT. Within the Data Transfer phase, a connection may be in a state of ESTABLISHED. Within a Termination Phase, a connection may be in a state of CLOSED, FIN-WAIT-1, FIN_WAIT-2, CLOSE-WAIT, TIME-WAIT, and TIME-WAIT. During the establishment of a connection, a server starts off in a LISTENING state. A client generates a TCP SYN packet, with the segment sequence number set to a random value X. The server responds with a

SYN-ACK packet (both bits set). The ACK number is to one more than the received sequence number (X+1). Also the server chooses a sequence number Y for packets which are being sent in the opposite direction. The client responds to the SYN-ACK with an ACK. The sequence number in the client to server direction is increased by one (X+1). When either the client or server wishes to terminate a connection, they use a four step sequence FIN,ACK,FIN,ACK or a three step sequence FIN, FIN+ACK, ACK. After which, the terminating side waits for a timeout before finally closing the connection.

Upon connection establishment, TCP uses a slow start mechanism to increase the congestion window, from an initial value of twice the Maximum Segment Size (MSS) [122]. With every packet acknowledgment, the congestion window increases by one Maximum Segment Size so that the congestion window doubles for every Round Trip Time (RTT). Karn's algorithm was used to better estimate the RTT [123]. When the congestion window exceeds the Slow Start Threshold (ssthresh), the algorithm enters the congestion avoidance state. Where the Initial value of ssthresh is large, the initial slow start usually completes in a packet loss. The Slow Start Threshold is updated at the end of each slow start. In the congestion avoidance state, the congestion window is increased by one Maximum Segment Size every Round Trip Time, as long as there are no duplicate ACK received. The probability of receiving a duplicate ACKs is high, when a packet is lost. For Triplicate ACKs, TCP Tahoe [122] performs a "fast retransmit", resetting to the slow-start state and reduces the congestion window to a single Maximum Segment Size. TCP Tahoe was the first TCP congestion control strategy. For each connection, TCP Tahoe maintains a congestion window that limits the total number of unacknowledged packets that may be in end-to-end transit. The congestion window is a derivation of the TCP sliding window for flow control.

Chapter 8 FLATLANd Architecture

8.1 Outline of FLATLANd Architecture

FLATLANd uses the Portland architecture [102] for Data Centres to facilitate an efficient hierarchy of layer-2 switches and distributed Openflow tables (across ONU/OLT, electrical and optical switches). A translation is performed between the real (physical) address of the end device and the internal structured (pseudo) addressing used within the network. In the case of LR-PON, this translation is performed at the ONU GEM port. The mechanism partitions the internal 48-bit address space of an Ethernet layer into a number of arbitrary subfields, each routed to a different part of the network. The correlation between the real and pseudo addressing is performed dynamically by the SDN controller. For the LR-PON scenario we have identified a possible addressing scheme based on the following allocation: '**mm-tt-nn-cc-gg-dd'**. Following the structure in Figure 50, '**mm**' identifies up to 4096 different metro-core nodes (12 bits), each with up to '**tt**' up to 4096 OLT ports (12 bits). Within an OLT port,'nn' identifies up to 4096 ONUs (12 bits), each with 16 '**cc**' T-CONTs (4 bits). A T-CONT is a group of logical connections that carries traffic within an ONU. Each T-CONT is identified by a unique Allocation Identifier (Alloc_ID) carrying traffic associated to one bandwidth type (i.e., QoS characteristic). The final 8 bits are split between GEM ports '**gg**' (4 bits or 16 GEM ports per T-CONT) and devices '**dd**' (4 bits or 16 devices per GEM ports). A GEM Port is a virtual port that encapsulates frames transmitted between the OLT and the ONU. Each traffic-class is assigned a different GEM Port. This would allow for example different users on the same ONU to acquire services from different providers concurrently. It should be noticed that we consider a classless address structure, where each block can have an arbitrary number of bits (up to a maximum sum of 48 bits, defined by the Ethernet address space limit).

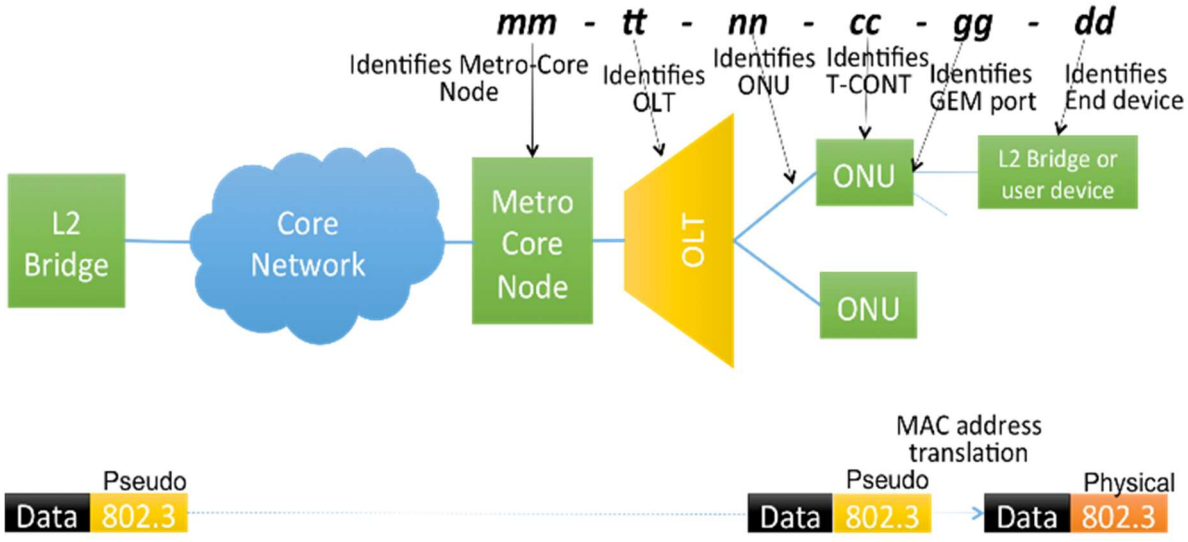


Figure 50 - FLATLANd FTTH function-level diagram

There are two main distinguishing differences between the scenarios of the data centre networks and the Passive Optical networks. Firstly, traffic is predominantly *consumed* by the PON access termination nodes (customer devices) whereas in the Data Centre, traffic is predominantly *generated* by the access termination nodes (Data Centre machines). The traffic ratios are essentially reversed, the significance of which is indeterminate at this stage. Secondly, and most crucially, access termination nodes are not under the control of the Infrastructure provider. This strictly precludes schemes that use secondary MAC addresses (*NANDO*) and IP-in-IP encapsulation (*VL2*) because the operator plays no part in the operation of the device configuration. While there is a need for customised components in *SPB, Trill and Portland*, the scale of *Trill and SPB* is limited by PBB encapsulation. In the case of *Portland*, the range of pseudo MAC addresses is relatively unbounded (2^48 or 281 Trillion addresses). There is some merit in the approach adopted by Portland due to its scale and flexible Layer 2 addressing scheme.

FLATLANd uses the Openflow broad and flexible definition of a flow. This definition has expanded from the basic 5-tuple to included other attributes such as MPLS labels, VLAN tags and IP TOS fields. All devices are granted access to the network but subsequently may be dynamically or statically bound to the profile of a target service provider. Indeed the flexibility of the addressing scheme favours multi-tenancy, as parts of the address can be used for packet routing purposes and other parts for QoS and SP differentiation. Distinct flow rules are created for the metering of each class of traffic at each Metro-Core node, OLT and ONUs. These are separate from the rules necessary or forwarding flows. Table 11 compare the characteristics of FLATLANd against those of other flow-based networks which were highlighted in section 2.4

Architecture	Pros	Cons
Flatland	Layer 2 mechanism that supplements other Flow-based approaches. Stateless Core No Signalling Inherent Admission Control	Rigid Service definition, required at planning stage
Integrated Services	Real Guarantees	Low Scalability
Connectionless Approach	No Signalling	Implicit differentiation, user misbehaviour Vulnerability
Dynamic Packet State	Stateless Core	Complex data handling
Feedback and Distribution	Simple core operations	Per-flow signalling, weak service differentiation
Flow-based differentiated Services	DiffServ scalability	Fixed Classes
Flow-Aware networking	No Signalling	Weak service differentiation
Flow-State Aware transport	Diverse service differentiation options	Complex signalling
Flow-Aggregated-Based Services	Diverse service differentiation	Complex signalling

Table 11 - Comparison of Flow based Networks

FLATLANd is distinguished from IP-layer QOS framework such as IntServ, DiffServ, by providing a layer 2 QoS guarantees at layer 2, and does not generally require signalling to provided service differentiation. The fact that QoS characteristics are determined in advance, to which flows bind, brings a certain level of rigidity to how applications use the network. The FLATLANd architecture facilitates the determination of QoS parameters on a per-application, per-user, per-ONU, per OLT per Service Provider level.

FLATLANd shares a common infrastructure amongst all Service Providers where bandwidth apportionment is done throughout the network. Figure 51 shows the contiguous 48-bit pseudo MAC address range. 36 bits of the address relate to the routing of traffic across the core and metro networks to an ONU. This is composed of Metro Core, OLT and ONU address portions. 12 bits of the address relate to the identification of Service Provider and Service Type. Bandwidth apportionment may be performed at the root of the network, which has visibility of all traffic flows in the network, however, that would require a continuous flow table which is unfeasibly large (with potentially 2^{48} of 48 entries).

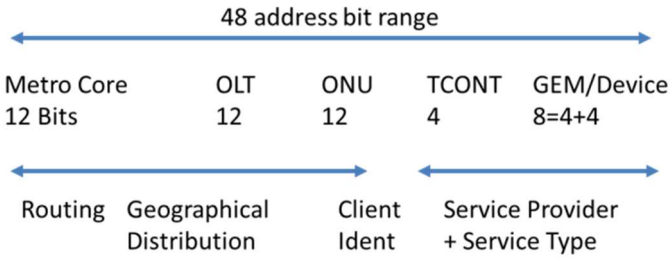


Figure 51 - 48-bit Address Range

The FLATLANd architecture allows two main approaches for distributed bandwidth apportionment: Geographical and per-Class. Geographical bandwidth apportionment applies control to the flows traversing each network element. For example, in order to apportion bandwidth according to a per-OLT basis, rules need to be applied at the upstream Metro Core network. In order to apportion bandwidth on a Service Type or Service Provider basis in the Geographical model, rules need to be applied to the upstream TCONT and GEM ports. The existing flow rules can be modified with the meter tags on the output action. Per-Class applies control to the flows traversing each network element. The key difference with the Geographical model is that distinct flow rules are created for the metering of each class of traffic at each Metro-Core, OLT and ONUs. These are separate from the rules necessary for forwarding flows. The advantage of per-Class bandwidth apportionment is that there is greater control over each Class of service across the network, whereas with Geographical, there is probably more efficient use of bandwidth.

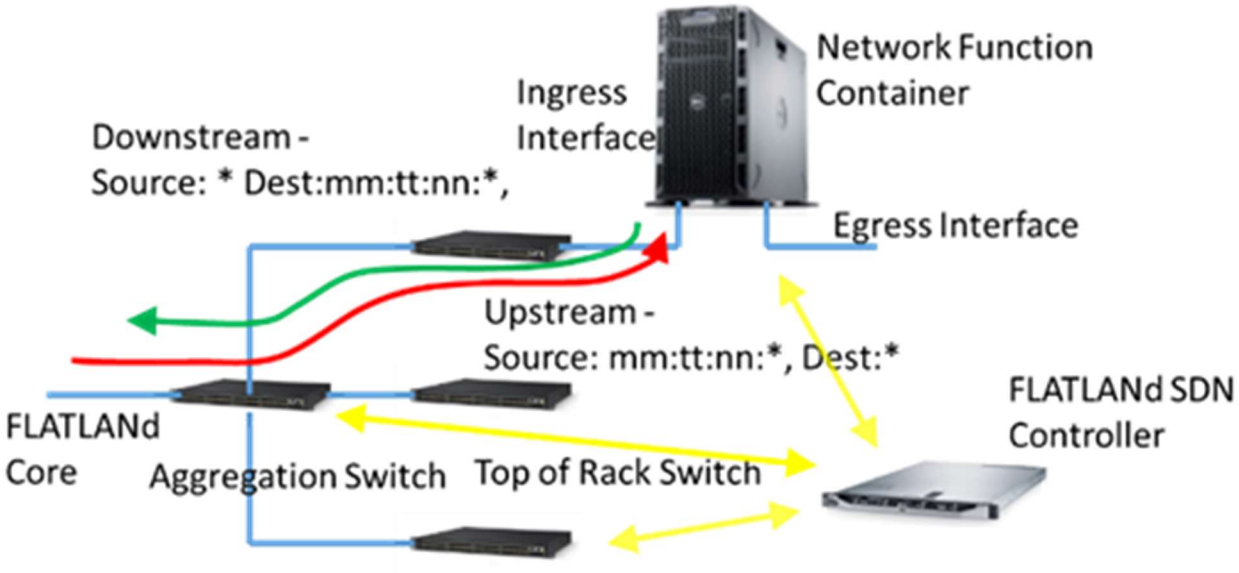


Figure 52 - FLATLANd Network Function Container

In the FLATLANd architecture, Network Functions are classified as either service-control or in-line. Examples of service-control are service authorisation and service binding.

Examples of in-line services are services that impact or touch the actual traffic being generated by an end-user, examples of which are firewalling and traffic monitoring and control and the ETSI vCPE. Figure 5 shows how the in-line services are provided in the Data Center, with dedicated processing allocated to each network terminating device such as an ONU. A multi-tiered Openflow based switch network connects Aggregation and Top of the Rack (ToR) layers. The centralised FLATLANd SDN Control function that directs the Core, Metro and Access network, also directs the Data Center NFV functions. Where there are 4 million ONU's then an equal number of Virtual Machine's (VM) would be allocated within the datacentre. The working ratio of VM's to physical machines is 20:1, however this ratio may be altered upwards or downwards on an individual basis. FLATLANd provides the equivalent of a virtual network between the customer terminating network and the VM in the Data Center. Each VM has two virtual interfaces, one of which faces into the FLATLANd core, the other faces into the public or provider network. In its simplest form, these virtual interfaces may be bridged in order to connect the terminating network with the public or provider network, or may form the ingress and egress interface of a single in-line network function such as a firewall or a chain of network functions. The virtual interfaces are opened in raw socket mode, so the Ethernet encapsulation (pseudo-mac) addresses and thereby preserving the identity (source MAC address) of the remote devices. Flows within the Data Center rely on the hierarchy of addressing. Upstream traffic flows that match the wild-carded **source** pseudo-mac address **mm:tt::nn:*** are directed through the switched network according to Openflow rules injected into the aggregation and TOR layer switches by the SDN controller. Similarly, downstream traffic match wild-carded **destination** pseudo-mac address **mm:tt::nn:***

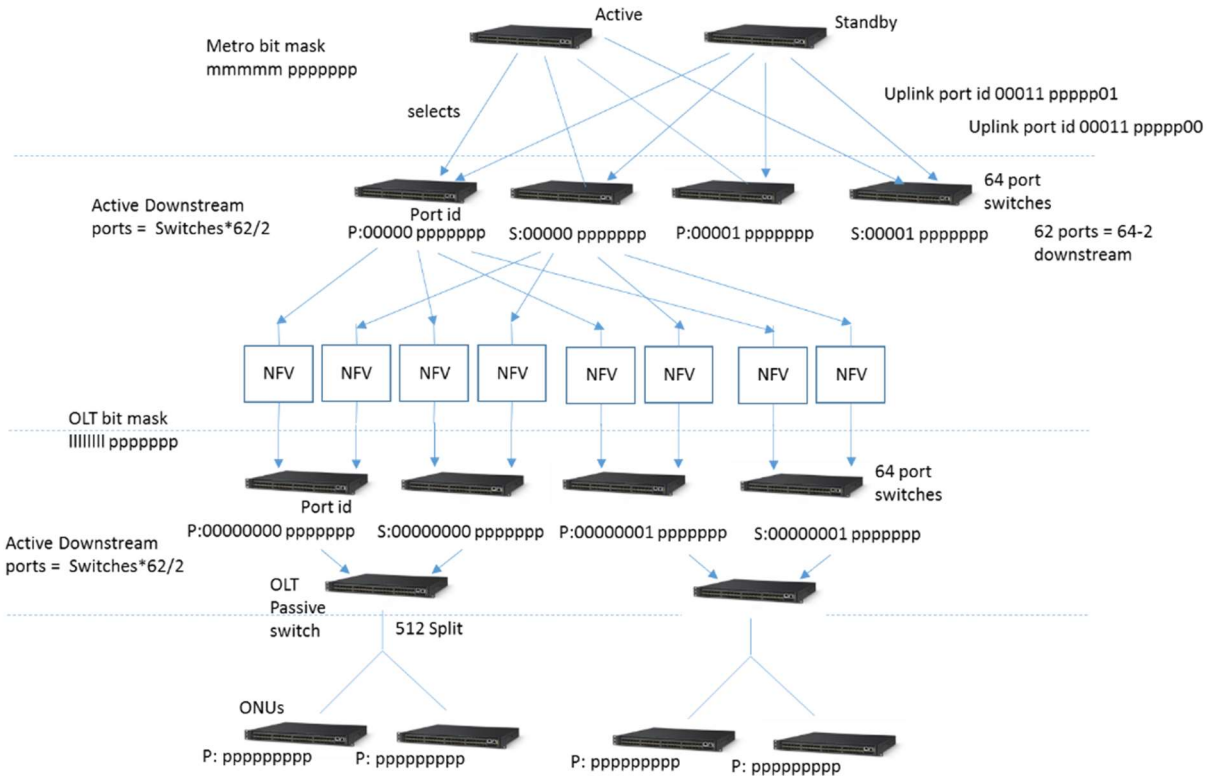


Figure 53 - Address Scheme

8.2 Architectural Patterns

An architectural pattern may be seen as a general, reusable solution to a commonly occurring problem, analogous to a software design pattern. To give examples of FLATLANd architectural design patterns, we use the FLATLANd addressing convention node1:node2:node3:node4. In FLATLANd the FLATLANd address 7:7:7:1:1 designates the first device off the first service provider off the 7th ONU off the 7th OLT off the 7th Metro Switch. This convention allows us to aggregate bits within the 48 bit pseudo Mac address range into groupings that are appropriate to their layers, in a manner akin to how 32 bit binary IP addresses are grouped into dotted decimal for person-friendly use. The number of bits from the total pseudo 48 bits assigned to a particular layer can be kept flexible.

The FLATLANd multi-service / Open Access pattern in Figure 54 gives the example of three separate devices off the one PON ONU which received service from three separate service providers.

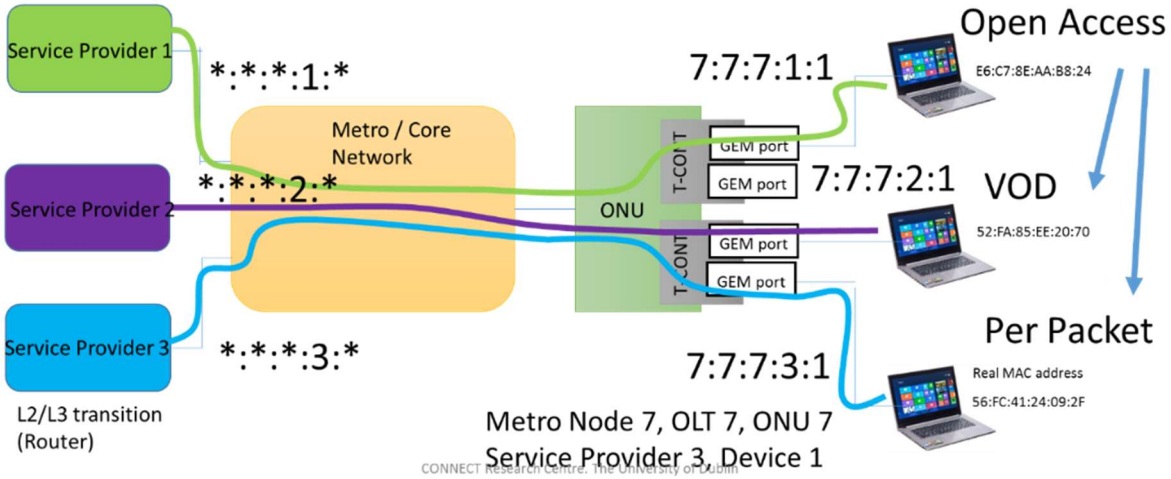


Figure 54 - FLATLANd Multi-service / Open Access Pattern

Device E6:C7:8E:AA:B8:24 with FLATLANd address 7:7:7:1:1 is bound to Service Provider 1 which has a profile of *:*:*:1:*. This has a traffic profile for Open Access. Device 52:FA:85:EE:20:70 with FLATLANd address 7:7:7:2:1 is bound to Service Provider 2 which has a profile of *:*:*:2:*. This has a traffic profile of Video on Demand. Device 56:FC:41:24:09:2F with FLATLANd address 7:7:7:3:1 is bound to Service Provider which has a profile of *:*:*:3:*, which has a low bit rate, all-ways on per packet traffic profile.

The traffic regulation pattern in Figure 55 shows how traffic regulation is applied at the various layers in the FLATLANd architecture.

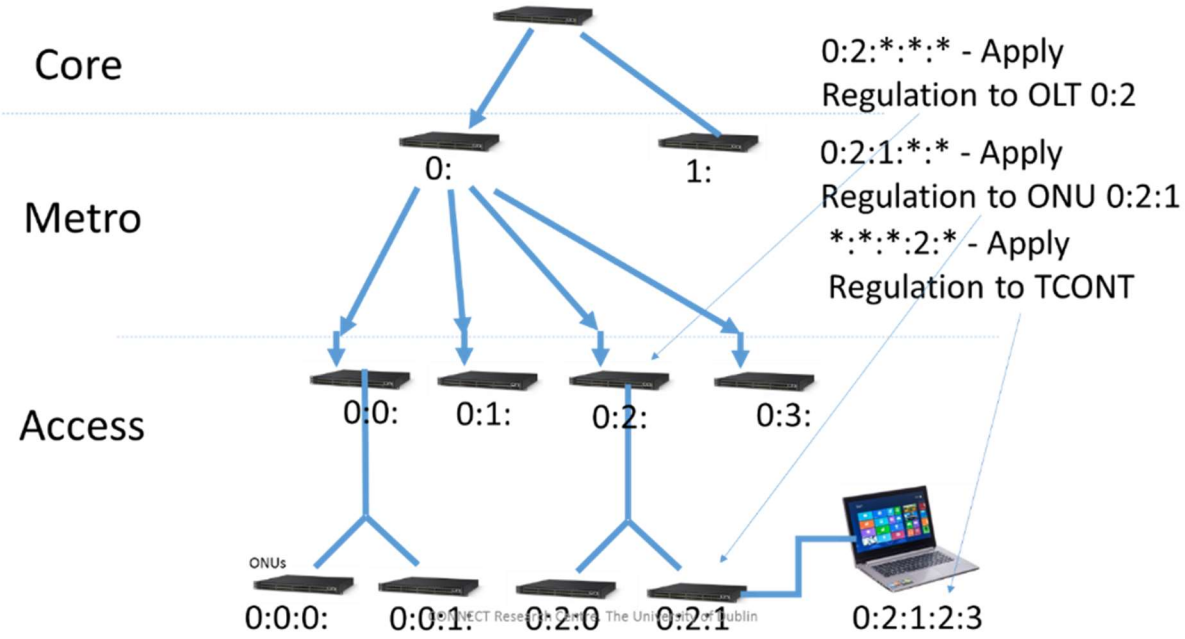


Figure 55 - FLATLANd Traffic Regulation Pattern

In order to regulate traffic for device 0:2:1:2:3, three meters are applied. At OLT 0:2, an Openflow meter is applied to network mask 0:2:*:*:*. At OLT 0:2:1, an Openflow meter is applied to network mask 0:2:1:*:*. Lastly, a meter is applied to TCONT *:*:*:2:*

The Protection Pattern in Figure 56 provides a primary and secondary route to ONU 0:0:0 along the path 0: to 0:0: to 0:0:0: A standby device picks up the identifier of the failed device.

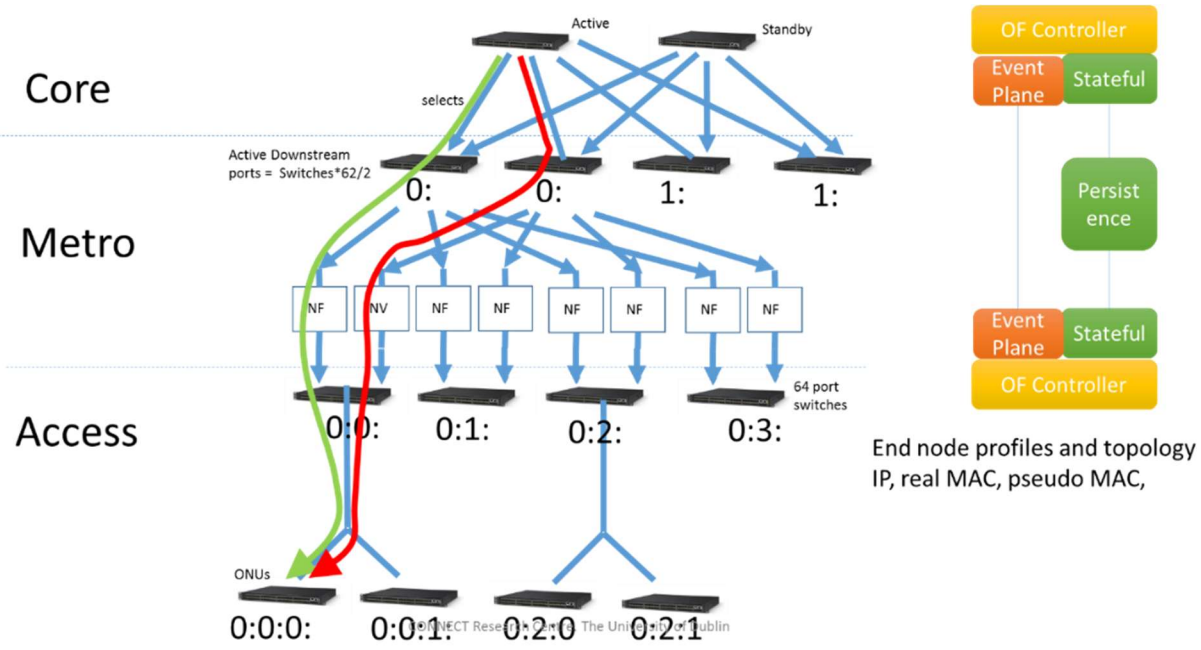


Figure 56 - FLATLANd Protection Pattern

Figure 57 show the FLATLANd NFV architectural pattern. A single layer datacenter switch is instantiated with small number of rules to direct the upstream and downstream traffic flows to and from the Network Function Virtual Machine.

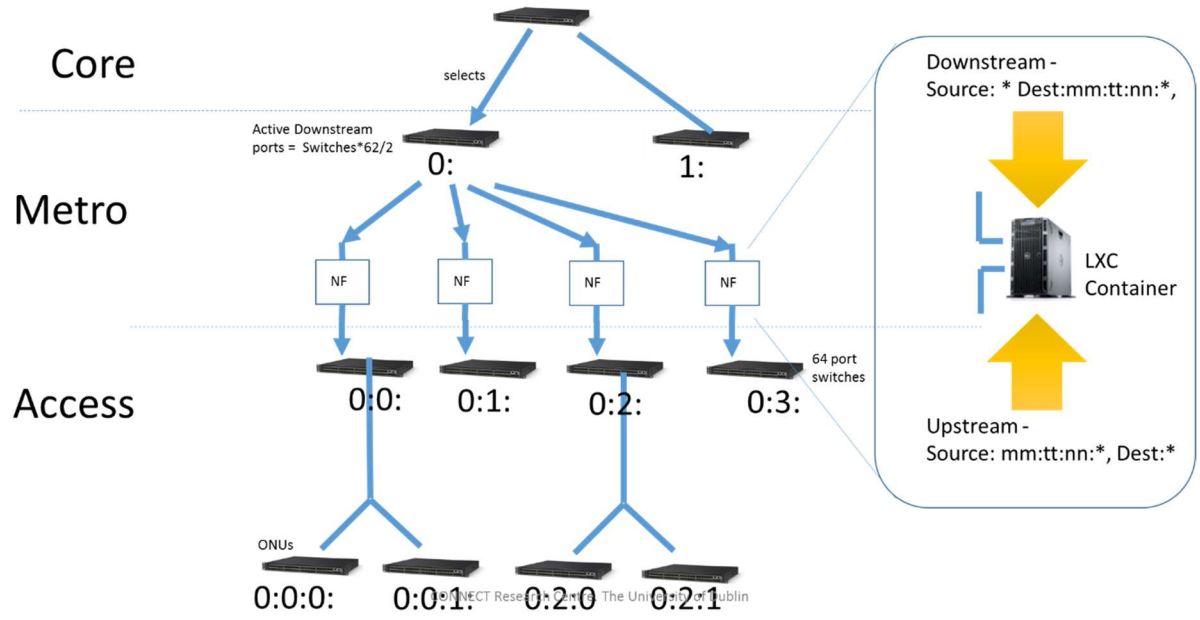


Figure 57 - Network Function Pattern

Linux LXC Container technology is used for the virtualisation, with the advantages that a low storage and processing overhead is imposed on the host environment. A basic traffic application was deployed on the Network Function Container, which both transited traffic between the ingress and egress interfaces as well as inspected and logged packet headers and payload.

8.3 Functional Validation

In implementing the flat Layer 2 network, the MAC address organisation is moving from a state of randomness to one of order and structure. In theory, the network should be compatible with all current applications and services. We validate the functionality of a number of client-server, peer-to-peer and network services such as DHCP, DNS and AAA (Authentication, Authorisation and Accounting) on the architecture, from the perspective of performance and resilience. The architecture is fundamentally a Layer 2 fat tree topology. The correlation between the MAC addresses and IP addresses of client devices is usually done through ARP (Address Resolution Protocol). Because the MAC address of the terminating devices are translated, these devices will not natively be able to respond to ARP requests from upstream devices. The issue that arises is how to implement ARP proxing or ARP translation. This function will require interaction with the service that leases IP addresses or with the device that is aware of the correlation between the real and pseudo MAC addresses. While there is considerable emphasis on the layer 2 routing and MAC translation in the downstream direction, a similar regime needs to be implemented in the upstream direction. This regime does not have to be as complicated since there are less constraints on bandwidth, however it does have to be robust. A technique employed in switched datacentre networks such DSR (Direct Server Return) could be both robust and simple. At the ONU termination, there needs to be optimal discovery of real MAC addresses and correlation with the pseudo MAC addresses through the MAC address translation algorithm. For test purposes, this functionality is not critical since correlation can initially be statically coded. However, in a (near-) production environment, any MAC address learning or translation or correlation algorithm needs to be scalable and fast.

We validate the operations of the SDN controller in the FLATLANd architecture in the stages required to register an ONU device with a given Service Provider: first pseudo address allocation and then layer-3 authorization. We validated the FLATLANd address partitioning and mapping scheme by replicating the 6-tier network hierarchy shown in Figure 4, and test key NFV functionality such as service registration. Service registration is the process that allows each network element to obtain a pseudo-MAC address unequivocally associated to its physical MAC address, thus enabling its association to the FLATLANd network, and Layer-3 authentication. To validate the functionality on a virtual environment implemented

on the Mininet platform [124]. A custom Openflow controller was developed, derived from the base POX implementation, and extended with memory-based Redis database. The Redis database is low-latency and can be (geographically) distributed across many physical machines, with some implementations handling millions of queries per second. Redis preserves transactionality between nodes. While there is a single master read/write node, changes in this database can be instantaneously mirrored across many read only nodes. For the current purposes, the database maintains the mappings between all real-mac, pseudo-mac addresses, IP addresses, and f lows both in the network in the Data Center for the virtualization of Network Functions.

Figure 58 shows the emulated network architecture, inclusive of emulated latency times between the network elements (the values used are only indicative of the LR-PON case study considered), and the client binding and registration process. The test initiates with the Layer-2 Bind Phase, where the client device at the GEM port of the ONU registers its interface on the network. This interface is configured to obtain its IP address from a DHCP server, situated centrally and upstream from the device.

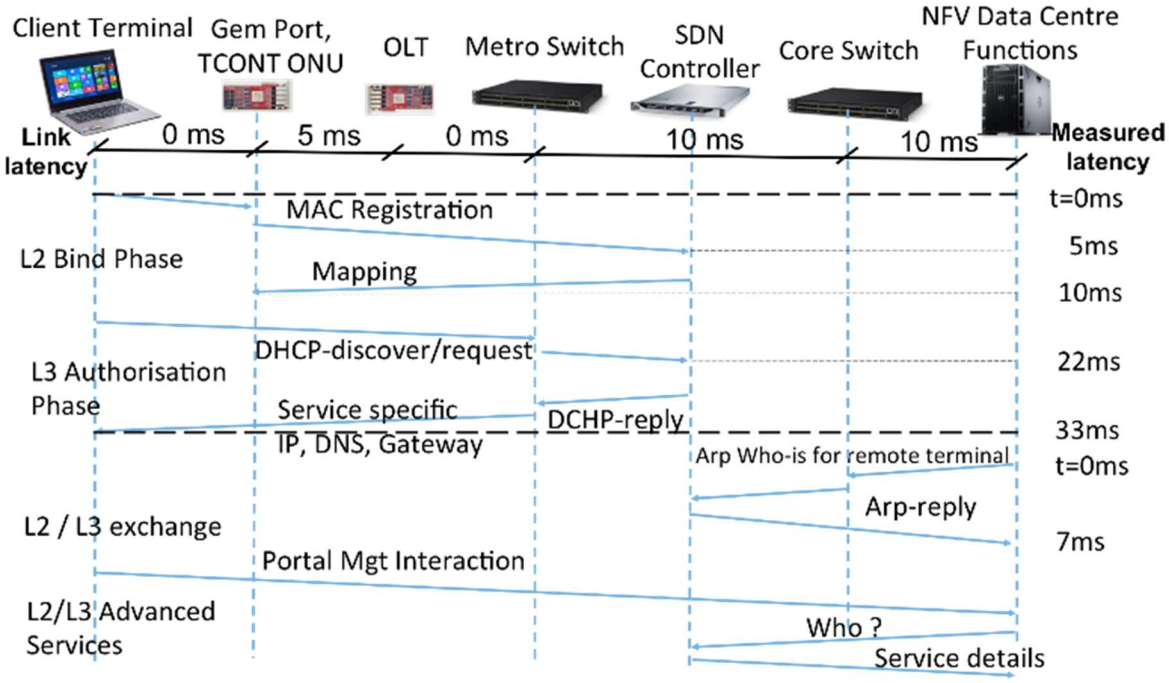


Figure 58 - Service Registration carried out on SDN/NFV testbed

On sensing of a DHCP-discover/request packet, the layer-2 of the customer Openflow-enabled ONU, sends the DHCP packet to the centralised Openflow Controller [125]. At the ONU the Openflow switching is operated by the ONU GEM port switch. The Openflow Controller then performs three actions. Since the Openflow Controller knows the ONU from which packets are received, the controller formulates a pseudo-MAC address appropriate to that ONU. The Openflow Controller database creates a forward and reverse mapping between the real- and pseudo-MAC addresses to allow fast database lookups. The mapping

is then sent to the ONU as an Openflow rule. The layer-3 authorization phase is required for the ONU client to receive appropriate network layer facilities such as IP address, DNS settings and Gateway addressing. The system operates by the Openflow controller intercepting a DHCP-discover/request either as part of the layer-2 bind phase or as a retransmission of this request. The Openflow Controller constructs a DHCP-reply packet with the appropriate settings, for transmission through the ONU switch, to the client. The Openflow Controller also constructs per-Service Provider IP addresses and DNS settings. In the ARP Exchange Phase, the end-points exchange IP addresses and MAC address pairings. Where the client device sends an IP packet to a Data Centre, the ARP who-is request is broadcast upstream and the upstream device responds with an ARP response. The GEM Port switch performs a swap of real- and pseudo-MAC addresses for the client device. The metro switch intercepts the ARP who-is request destined for the pseudo-MAC of the client device. Finally, the controller performs a proxy-ARP functionality based on the pseudo-MAC address of the client device.

To demonstrate the NFV functionality, a single layer Data Center switch was instantiated with small number of rules to direct the upstream and downstream traffic flows to and from the Network Function Virtual Machine. Linux LXC Container technology was used for the virtualisation, with the advantages that a low storage and processing overhead is imposed on the host environment. A basic traffic application was deployed on the Network Function Container, which both transited traffic between the ingress and egress interfaces as well as inspected and logged packet headers and payload.

The testbed results show that registration times of around 30 milliseconds can be achieved for the LR-PON based scenario shown in Figure 58. While such operations are generally not time-critical, these results demonstrate the type of benefits that a simplified SDN-driven flat architecture can bring about. Once registration was complete, we successfully transmitted traffic between the client and Data Center end-points. The traffic included both typical HTTP web traffic, but also less conventional Ethernet frames more suited to the transit of IoT device traffic.

8.4 Performance Scenarios

To benchmark the performance of the proposed Flatland scheme against the state of the art Classical QoS Frameworks, we simulated the models in NSIM and applied similar traffic profiles to the models.

The classical architecture (Figure 59) is characterised by 5 distinct domains – Customer Premises, the Access Network, Metro Network, Core Network and the Data Centre.

Within the Customer premises, there is typically an ONU which physically terminates the network and acts as a line protocol DTE, and which presents a higher layer connection, most typically Ethernet, to internal customer premises equipment. Typically, the CPE is a

multilayer device which acts as a Layer 2 switch, a PPPOE termination device and a NAT firewall.

The network line protocol extends across the Access Network between the customer premises as well as the network providers local exchange. The function of the Access Network is to provide the physical distribution of the cabling to the customer terminations, as well as to physically aggregate the cabling for termination upstream on the Metro Node. The function of the Metro Node is to physically terminate tunnel connections from the Access Network, aggregate traffic and to create tunnel connections across the core. Tunnelling of connections allows traffic to follow a predetermined route with an optional alternative or diverse path that can be invoked for protection or the provision of additional capacity. Such routes can have a level of assurance over quality metrics, and are typically assigned to tenant service providers according to network wholesale model. A Metro Network provides a distribution of these nodes, such that a large geographical area is covered, such as the provinces in a country or districts in a city.

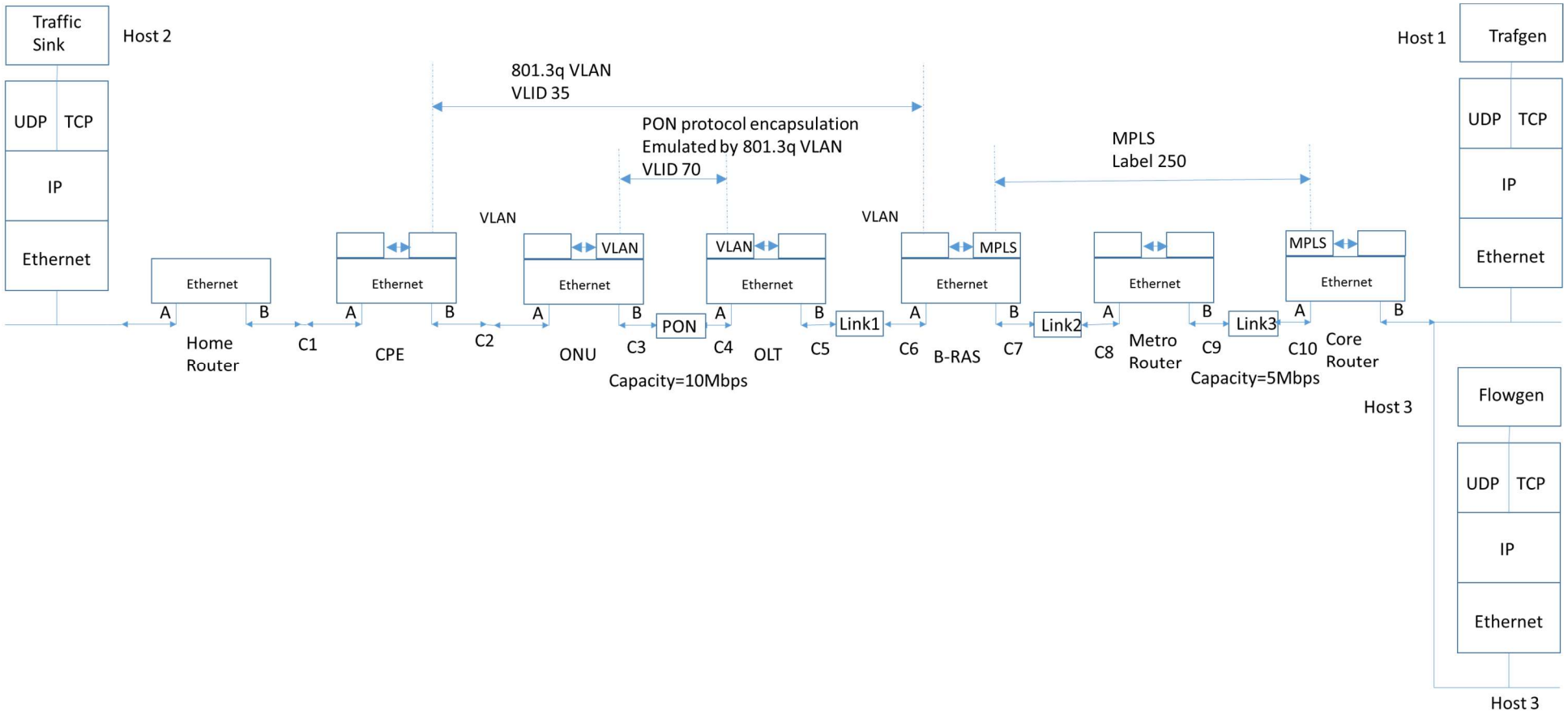


Figure 59 - Classic Model simulation

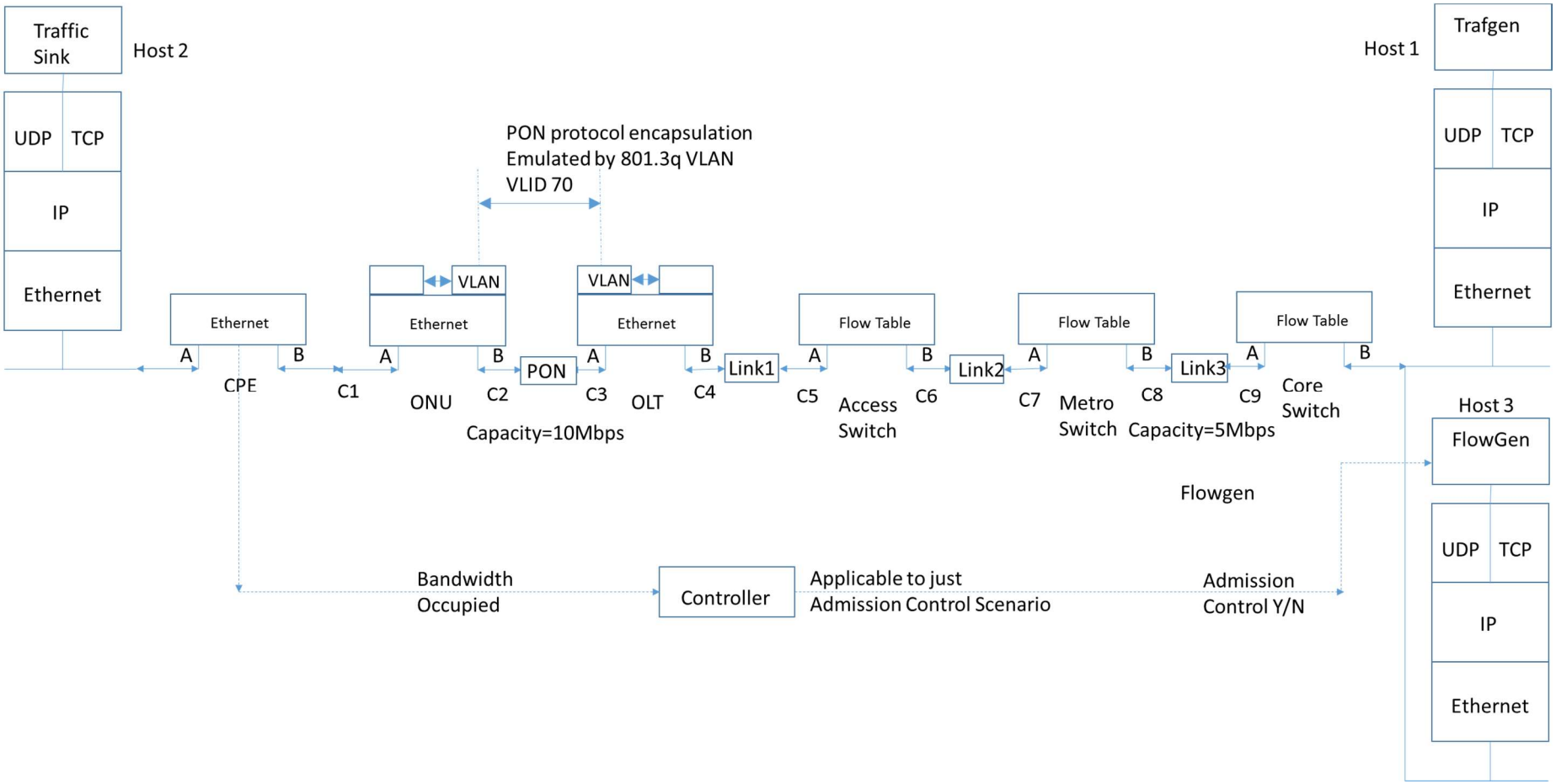


Figure 60- FLATLANd Model Simulation

The Core network aggregates traffic even further while transporting it back to a small number of nodes, operated by the host network provider, or by tenant service providers. The core nodes as well as the network connections to and from the core nodes require a very high level of performance and resilience. The core nodes terminate the tunnelled connections from Metro Nodes. The Data Centre hosts network functions that are necessary for authenticating / authorising clients on the network, providing routing information and support services such as Domain Name Services. The FLATLANd architecture (Figure 60) is characterised by 3 distinct domains – Customer Premises, the hybrid Access / Metro Network and the Core Network. The Data centre functionality is co-locate with the Access / Metro Network.

The distinction between the Classical and Flatland schemes is that, in the Classical case, the network nodes have full stacks and buffers of a typical size, while in the Flatland case, these nodes have been replaced by switches with minimal buffers. The performance metrics we measure and compare are the Congestion Windows (CWND) and Round Trip Time (RTT) specifically for TCP, and the packet Loss, throughput and jitter for both UDP and TCP transport protocols. The six scenarios are as follows:

Scenario	Topology	Enhancement
0	Point to Point link between Hosts	N/A
2	Classic Architecture	Standard Queues
3	Classic Architecture	Aged Queues
4	FLATLANd Architecture	Standard Queues
5	FLATLANd Architecture	Small Queues
6	FLATLANd Architecture	Flow regulation

Table 12 - Simulation scenarios

All scenarios shared the same NSIM header and footer configuration (Figure 61). This defines the duration of the scenario and the simulation clock ticks. The clock ticks determine the granularity with which the scheduler controls and reports on events.

```
sched=scheduler(tick=0.001,finish=5)

host1=host('host1',stack='udp') # Good traffic generator
host2=host('host2',stack='udp',mdrop='00:00:00:00:00:00') # terminal, dropping fake traffic
host3=host('host3',stack='udp',mdst='00:00:00:00:00:00') # fake traffic generator

traf=trafgen('traf1',ms1=1)
term2=terminal('term2')
```

```
standardbuffers=64000
smallbuffers=8000

#### Scenarios go here <<<<<<<<

sched.process()
```

Figure 61 - Shared Configuration

A host (host1) application generates useful traffic, which is direction to a traffic sink (host2). In this example, all hosts are built with a UDP stack. For TCP scenarios, the hosts are built with a TCP stack. Host2 is directed to drop traffic with a target Ethernet address of '00:00:00:00:00:00', which is generated by the load traffic generator attached to host3. The ms1=1 parameter configures the good traffic application (trafgen) to emit application packets at a rate of 1 every millisecond. The standard size of each packet is 200 Bytes, which is then encapsulated in UDP or TCP and the other lower levels in the protocol stack. We define small and standard buffer size constants of 8000 Bytes and 64000 Bytes respectively. The scenario is run using the sched.process() command at the end of the configuration script.

Scenario 0 (Figure 62) is a baseline scenario with a simple topology. The hosts are connected using an (Ethernet) datalink with a capacity of 10 Mbps and an end to end latency of 10 milliseconds. The sending hosts host1 (good traffic) and host 3 (load traffic) are placed at one end of the datalink. The receiving host2 is placed at the other end.

```
flowgen=flowgen('flowgen',start=0.002,stop=1.0,ival=0.05,flowcount=5)
sw=datalink('pon',capacity=10, latency=10)
connect('hostcon1',host1.B,traf.B)
connect('con1',host1.A,sw.B)
connect('con2',sw.A,host2.A)
connect('hostcon2',host2.B,term2.A)
connect('flow',flowgen.B, host3.B)
connect('con3',host3.A, sw.B)
```

Figure 62 - Scenario 0

The load traffic generator flowgen triggers 5 flows starting at t=0.002 seconds, with an interval of 50 milliseconds between them. flowgen terminates all flows at t=1.000 seconds. Scenario 2 (Figure 63) is the scenario for the classic architecture. All components such as datalinks, Ethernet switches and routers are configured with standard buffer sizes.

```
flowgen=flowgen('flowgen',start=0.002,stop=1.0,ival=0.05,flowcount=5)
pon=datalink('pon',latency=2,capacity=10,ber=-12,MaxSize=standardbuffers)
link1=datalink('link1',latency=2,ber=-12,MaxSize=standardbuffers)
link2=datalink('link2',latency=2,ber=-12,capacity=5,MaxSize=standardbuffers)
link3=datalink('link3',latency=2,ber=-12,MaxSize=standardbuffers)
onu=vswitch('onu',"","Dot1Q(vlan=22)",MaxSize=standardbuffers)
olt=vswitch('olt',"Dot1Q(vlan=22)","",MaxSize=standardbuffers)
cpe=vswitch('cpe',"","Dot1Q(vlan=35)",MaxSize=standardbuffers)
bras=vswitch('bras',"Dot1Q(vlan=35)","",MaxSize=standardbuffers)
homerouter=eth_switch('hr',MaxSize=standardbuffers,profile=True)
metrorouter=vswitch('mr',"","MPLS(label=250)",MaxSize=standardbuffers)
corerouter=vswitch('cr',"MPLS(label=250)","",MaxSize=standardbuffers)
connect('hostcon1',host1.B,traf.B)
connect('hostcon2',host2.B,term2.A)
connect('c1',homerouter.B,cpe.A)
connect('c2',cpe.B,onu.A)
connect('c3',onu.B,pon.A)
connect('c4',pon.B,olt.A)
connect('c5',olt.B,bras.A)
connect('c6',bras.B,link1.A)
connect('c7',link1.B,metrorouter.A)
connect('c8',metrorouter.B,link2.A)
connect('c9',link2.B,corerouter.A)
connect('c10',corerouter.B,link3.A)
connect('c8',host2.A,homerouter.A)
connect('c9',host1.A,link3.B)
#
connect('flow',flowgen.B, host3.B)
connect('con3',host3.A, corerouter.B)
```

Figure 63 - Scenario 2

All datalinks (pon, link1, link2 and link3) are configured with a Bit Error Rate (BER) of one bit error in 10^-12 bits transmitted. link1, link2 and link3 are not rate limited. Traffic on the PON link are constrained by a rate limit of 10 Mbps. The traffic on link2 between the Core Router and the Metro Router is deliberately rate limited to 5 Mbps. Traffic between the ONU and the OLT is encapsulated in a 801.3q tunnel Traffic traversing the ONU from the A to the B interface tags traffic with VLAN ID 22. The VLAN tag on traffic going in the opposite direction (from interface B to interface A) is dropped. Traffic between the CPE and the BRAS

is encapsulated in an 801.3q tunnel with Vlan ID 35. VLAN encapsulation is used to emulate PPPoE encapsulation. Traffic traversing the CPE from the A to the B interface tags traffic with VLAN ID 35. The VLAN tag on traffic going in the opposite direction (from interface B to interface A) is dropped. Traffic between the Core Router and the Metro Router is encapsulated in an MPLS tunnel with label 250.

Scenario 3 (Figure 64) is similar to Scenario 2, but with one difference. Both routers connected through link1 with the traffic restriction of 5 Mbps, are configured with queues that use an aged queue discipline. Any traffic data older than 10 milliseconds that is dropped from the internal buffers of the core router and metro router.

```
flowgen=flowgen('flowgen',start=0.002,stop=1.0,ival=0.05,flowcount=5)
pon=datalink('pon',latency=2,capacity=10,ber=-12,MaxSize=standardbuffers)
link1=datalink('link1',latency=2,ber=-12,MaxSize=standardbuffers)
link2=datalink('link2',latency=2,ber=-12,capacity=5,MaxSize=standardbuffers)
link3=datalink('link3',latency=2,ber=-12,MaxSize=standardbuffers)
onu=vswitch('onu',"","Dot1Q(vlan=22)",MaxSize=standardbuffers)
olt=vswitch('olt',"Dot1Q(vlan=22)","",MaxSize=standardbuffers)
cpe=vswitch('cpe',"","Dot1Q(vlan=35)",MaxSize=standardbuffers)
bras=vswitch('bras',"Dot1Q(vlan=35)","",MaxSize=standardbuffers)
homerouter=eth_switch('hr',MaxSize=standardbuffers,profile=True)
metrorouter=vswitch('mr',"","MPLS(label=250)",MaxSize=standardbuffers,age=10)
corerouter=vswitch('cr',"MPLS(label=250)","",MaxSize=standardbuffers,age=10)
connect('hostcon1',host1.B,traf.B)
connect('hostcon2',host2.B,term2.A)
connect('c1',homerouter.B,cpe.A)
connect('c2',cpe.B,onu.A)
connect('c3',onu.B,pon.A)
connect('c4',pon.B,olt.A)
connect('c5',olt.B,bras.A)
connect('c6',bras.B,link1.A)
connect('c7',link1.B,metrorouter.A)
connect('c8',metrorouter.B,link2.A)
connect('c9',link2.B,corerouter.A)
connect('c10',corerouter.B,link3.A)
connect('c8',host2.A,homerouter.A)
connect('c9',host1.A,link3.B)
#
connect('flow',flowgen.B, host3.B)
```

```
connect('con3',host3.A, corerouter.B)
```

Figure 64 - Scenario 3

Scenario 4 (Figure 65) shows the configuration for the FLATLANd architecture. All components such as datalinks and switches are configured with standard buffer sizes. All datalinks (pon, link1, link2 and link3) are configured with a Bit Error Rate (BER) of one bit error in 10^-12 bits transmitted. link1, link2 and link3 are not rate limited. Traffic on the PON link are constrained by a rate limit of 10 Mbps. The traffic on link2 between the Core Router and the Metro Router is deliberately rate limited to 5 Mbps. Traffic between the ONU and the OLT is encapsulated in a 801.3q tunnel Traffic traversing the ONU from the A to the B interface tags traffic with VLAN ID 70.

```
flowgen=flowgen('flowgen',start=0.002,stop=1.0,ival=0.05,flowcount=5)
pon=datalink('pon',latency=2,ber=-12,capacity=10,MaxSize=standardbuffers)
link1=datalink('link1',latency=2,ber=-12,MaxSize=standardbuffers)
link2=datalink('link2',latency=2,ber=-12,MaxSize=standardbuffers)
link3=datalink('link3',latency=2,ber=-12,MaxSize=standardbuffers,capacity=5)
onu=vswitch('onu',"","Dot1Q(vlan=70)",MaxSize=standardbuffers)
olt=vswitch('olt',"Dot1Q(vlan=70)","",MaxSize=standardbuffers)
cpe=eth_switch('cpe',MaxSize=standardbuffers,profile=True)
accessswitch=eth_switch('as',MaxSize=standardbuffers,profile=True)
metroswitch=eth_switch('ms',MaxSize=standardbuffers,profile=True)
coreswitch=eth_switch('cs',profile=True)
connect('c1',cpe.B,onu.A)
connect('c2',onu.B,pon.A)
connect('c3',pon.B,olt.A)
connect('c4',olt.B,link1.A)
connect('c5',link1.B,accessswitch.A)
connect('c6',accessswitch.B,link2.A)
connect('c7',link2.B,metroswitch.A)
connect('c8',metroswitch.B,link3.A)
connect('c9',link3.B,coreswitch.A)
connect('c10',host1.A,coreswitch.B)
connect('c11',host2.A,cpe.A)
connect('hostcon1',host1.B,traf.B)
connect('hostcon2',host2.B,term2.A)
#
connect('flow',flowgen.B, host3.B)
```

```
connect('con3',host3.A, coreswitch.B)
```

Figure 65 - Scenario 4

The VLAN tag on traffic going in the opposite direction (from interface B to interface A) is dropped.

We investigate the use of small buffers with the FLATLANd architecture in Scenario 5 (Figure 66). All components such as datalinks and switches are configured with standard buffer sizes.

```
flowgen=flowgen('flowgen',start=0.002,stop=1.0,ival=0.05,flowcount=5)
pon=datalink('pon',latency=2,ber=-12,capacity=10,MaxSize=smallbuffers)
link1=datalink('link1',latency=2,ber=-12,MaxSize=smallbuffers)
link2=datalink('link2',latency=2,ber=-12,MaxSize=smallbuffers)
link3=datalink('link3',latency=2,ber=-12,MaxSize=smallbuffers,capacity=5)
onu=vswitch('onu',"","Dot1Q(vlan=70)",MaxSize=smallbuffers)
olt=vswitch('olt',"Dot1Q(vlan=70)","",MaxSize=smallbuffers)
cpe=eth_switch('cpe',MaxSize=smallbuffers)
accessswitch=eth_switch('as',MaxSize=smallbuffers)
metroswitch=eth_switch('ms',MaxSize=smallbuffers)
coreswitch=eth_switch('cs')
connect('c1',cpe.B,onu.A)
connect('c2',onu.B,pon.A)
connect('c3',pon.B,olt.A)
connect('c4',olt.B,link1.A)
connect('c5',link1.B,accessswitch.A)
connect('c6',accessswitch.B,link2.A)
connect('c7',link2.B,metroswitch.A)
connect('c8',metroswitch.B,link3.A)
connect('c9',link3.B,coreswitch.A)
connect('c10',host1.A,coreswitch.B)
connect('c11',host2.A,cpe.A)
connect('hostcon1',host1.B,traf.B)
connect('hostcon2',host2.B,term2.A)
#
connect('flow',flowgen.B, host3.B)
connect('con3',host3.A, coreswitch.B)
```

Figure 66 - Scenario 5

In scenario 6 (Figure 67), we apply admission control to the FLATLANd architecture for new flows being created in the down stream direction. The addition of the fblimit parameter to the flowgen initialisation only allows flowgen flows on to the network when the the data rate at the CPE is less than the value of fblimit measure in Mbps.

```
flowgen=flowgen('flowgen',start=0.002,stop=1.0,ival=0.05,flowcount=5,fblimit=3)
pon=datalink('pon',latency=2,ber=-12,capacity=10,MaxSize=smallbuffers)
link1=datalink('link1',latency=2,ber=-12,MaxSize=smallbuffers)
link2=datalink('link2',latency=2,ber=-12,MaxSize=smallbuffers)
link3=datalink('link3',latency=2,ber=-12,MaxSize=smallbuffers,capacity=5)
onu=vswitch('onu',"","Dot1Q(vlan=70)",MaxSize=smallbuffers)
olt=vswitch('olt',"Dot1Q(vlan=70)","",MaxSize=smallbuffers)
cpe=eth_switch('cpe',MaxSize=smallbuffers)
accessswitch=eth_switch('as',MaxSize=smallbuffers)
metroswitch=eth_switch('ms',MaxSize=smallbuffers)
coreswitch=eth_switch('cs')
connect('c1',cpe.B,onu.A)
connect('c2',onu.B,pon.A)
connect('c3',pon.B,olt.A)
connect('c4',olt.B,link1.A)
connect('c5',link1.B,accessswitch.A)
connect('c6',accessswitch.B,link2.A)
connect('c7',link2.B,metroswitch.A)
connect('c8',metroswitch.B,link3.A)
connect('c9',link3.B,coreswitch.A)
connect('c10',host1.A,coreswitch.B)
connect('c11',host2.A,cpe.A)
connect('hostcon1',host1.B,traf.B)
connect('hostcon2',host2.B,term2.A)
#
connect('flow',flowgen.B, host3.B)
connect('con3',host3.A, coreswitch.B)
```

Figure 67 - Scenario 6

.

8.5 Performance Results

Figure 68 shows the traffic flows which are applied to the NSIM simulation topologies. The simulation lasts for 5 seconds in total. For the duration of the simulation, a constant traffic stream is generated by the trafgen traffic generator attached to host1 and to the traffic sink attached to host 2. This traffic is termed goodput or the effective traffic being generated and received at the application layer. For the simulations, this is set at 1.6Mbps. This is to be

distinguished from the actual throughput at the data link layer which includes the application data encapsulated with TCP/IP layer headers and trailers.

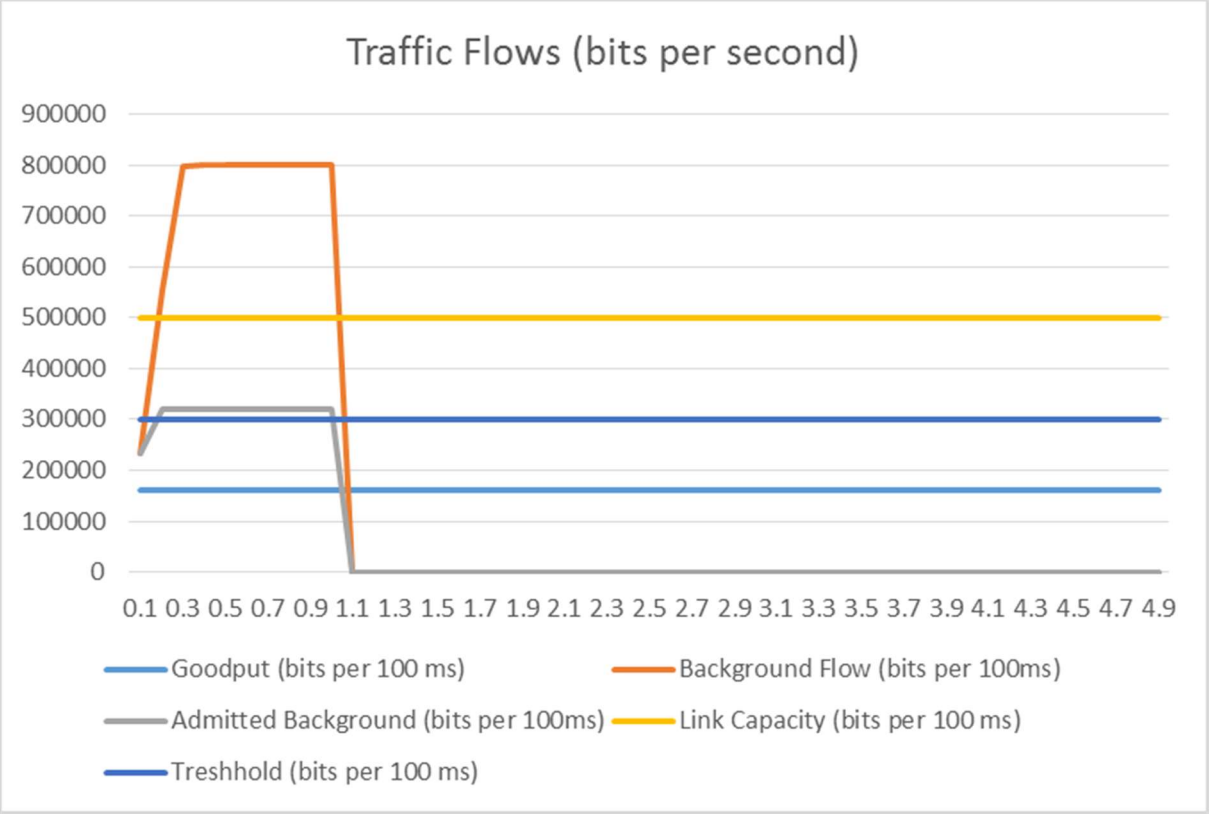


Figure 68 - Traffic Flows

At t=0.002 seconds, the background traffic load starts and ramps up quickly so as to swamp both or either of the limited bandwidth in the network topology or the buffers and queues in the network device. The load traffic lasts until t=1.000 seconds, when it stops abruptly. Shown on the graph also is the restricted bandwidth capacity also present in some of the traffic models. This is set at 5 Mbps. Also shown is the admission control limit of 3 Mbps. Again, this is present only some of the simulation models.

The scenario numbers are given in table Table 12. Figure 69 shows the results of the UDP performance tests for the six scenarios, each graph showing a different performance metric. It should be noted that scenario 0 is a reference scenario that uses a basic network between transmitting and receiving hosts. It has a latency of 10 milliseconds and is included so as to provide a benchmark for the other scenarios. Scenarui 0 does not include any load traffic congestion. Scenario 3 has a number of distinct characteristics in comparison to the other scenarios. Scenario 3 uses a queuing discipline at the metro and core network that disposes of packets with an age of 10 milliseconds or older. We see a high level of packet loss (477), with the cause seen in the Aged Queue graph.

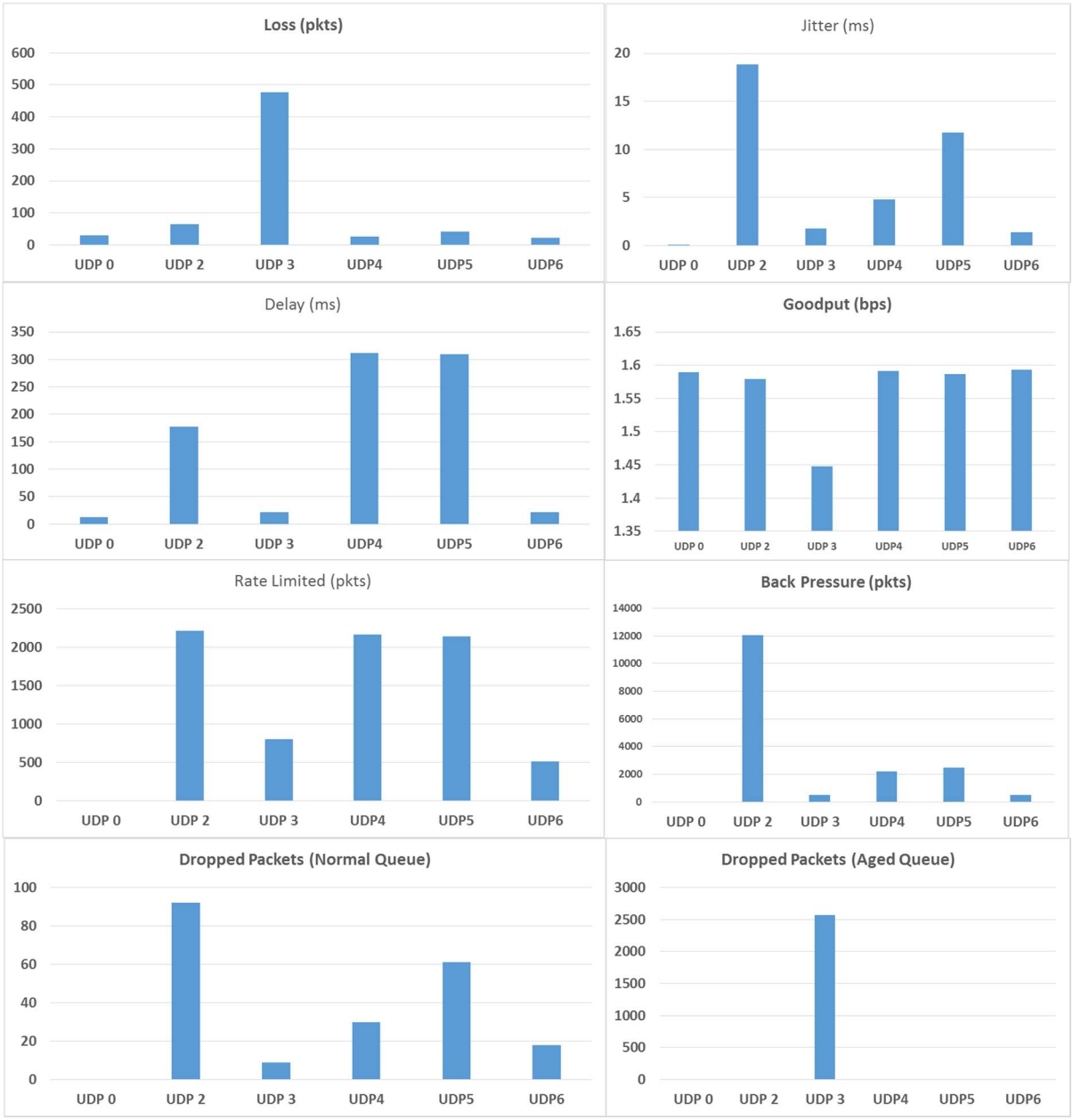


Figure 69 - UDP performance metrics

The aged queue graph shows a much higher level of packet loss (2574) because this includes good traffic as well as the load traffic. Much of the packet loss happens during the period of congestion (from time t=0.002 to t=1.000 in the simulation). Packet loss due to Normal Queue overrun (6), Rate Limiting Events (802) and Back pressure events (505) for scenario 3 are quiet low compared to the other six scenarios. While scenario 3 has a very good average jitter value of 1.719 milliseconds and average delay of 21 milliseconds, the overall throughput is poorest at 1.4476 Mbps. Unlike TCP, UDP is not a guaranteed delivery protocol, so once the packets are lost, it is the responsibility of the higher application layer to recover the lost packets. The classic architecture (scenario 2) has the poorest UPD average jitter (18.83 milliseconds) over the simulation. While packet loss due to buffer overrun is high (92), this is very low compared to the dropped packet strategy in scenario 3

(2574). When the link between the metro and core router gets congested, there are 2216 rate limiting events, which results in a very high level of back pressure events (12057). Thus Scenario 2 deals with congestion by filling and emptying buffers repeatedly, giving rise to the high metric for jitter. Scenario 4 is the FLATLANd scenario with standard buffer sizes. Average packet delay is high (312 milliseconds) and average jitter is low (4.765 milliseconds). Congestion causes a high level rate limit events (2164), which causes the buffers at the core router to drop packets (30). The effective loss of good packets is 25. Scenario 5 reduces the buffer size of scenario 4. There is a marginal improvement in delay (dropping from 312 to 309 milliseconds) but a significant degradation in jitter, rising from 4.765 milliseconds to 11.749 milliseconds. Because the buffers have been reduced, the effective number of packet losses rises from 25 to 41. Scenario 6 uses admission control to prevent flows from joining the network, thereby preserving back width for existing flows. As expected, average delay is as good as the aged queue scenario (21.663 milliseconds), and jitter is good (1.381 milliseconds), this is because all three underlying causes for poor performance are low. There are just 21 lost packets, 509 rate limiting events and 504 back pressure events.

The equivalent TCP performance metrics are given in Figure 70. These metrics reflect additional protocol overhead that TCP uses to resend lost packets, as well as congestion control mechanisms to optimise the throughput of traffic given varying network conditions. The average Jitter values for the standard and small buffers versions of FLATLANd have evened out to 0.259 milliseconds and 0.137 milliseconds respectively. Because the congestion event (Figure 72) happens within one Retransmission Time Out (RTO), initially set to 1 second, and because the Round Trip Time causes delayed acknowledgment, the sender resends packets. Much of the bandwidth for scenarios 4 and 5 after the congestion event (time t=1.000 seconds to t=2.000) is taken up with packet retransmissions. This causes a significant increase in Round Trip Time (1.2 seconds)

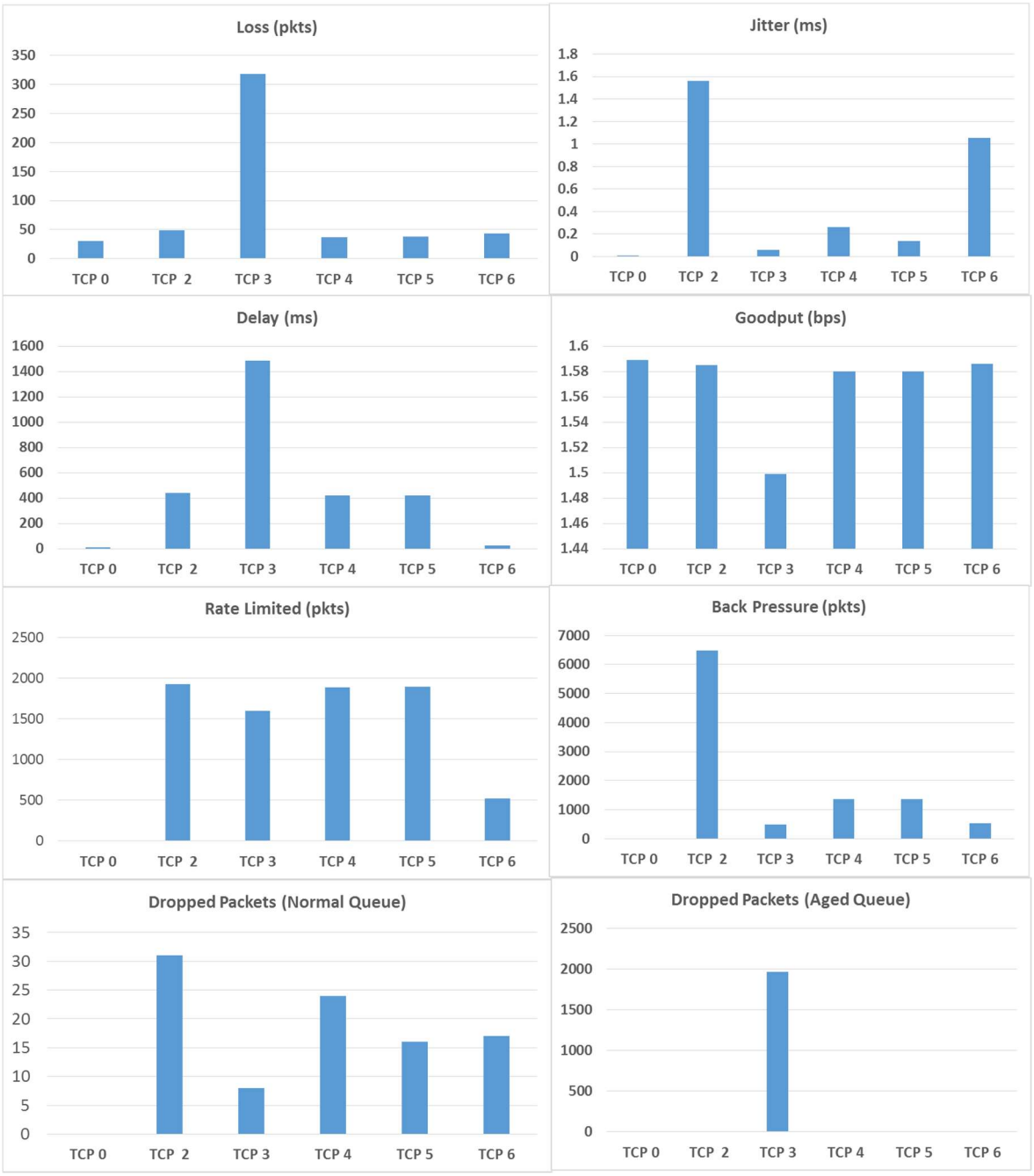


Figure 70 - TCP performance metrics

The strict aged queue policy has a significant effect on all packets buffered in the core and metro routers. Not only are first time transmit packets which exceed 10 milliseconds disposed of, but also retransmitted traffic is affected. The effect is to flatten the Congestion

Window (Figure 71) for up to 2 seconds after the removal of the congestion event. Like the UDP scenario 3, the throughput for TCP scenario 3, poor (1.5 Mbps)

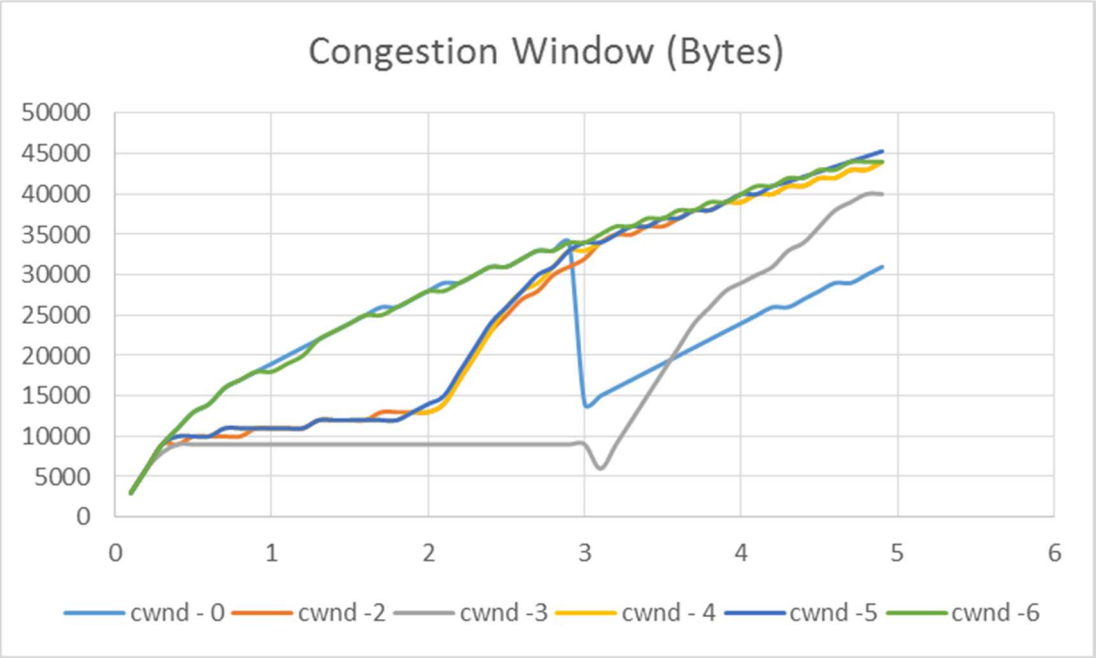


Figure 71 - TCP Congestion Window

The TCP Round Trip Times are quite predictable with the TCP scenario 3 having a flat response due to the fixed packet discard threshold. Classic scenario 2 has an adequate RTT response, given that it does not discard incoming flows nor discards aged packets. Flatland scenarios 4 and 5 follow the same graph.

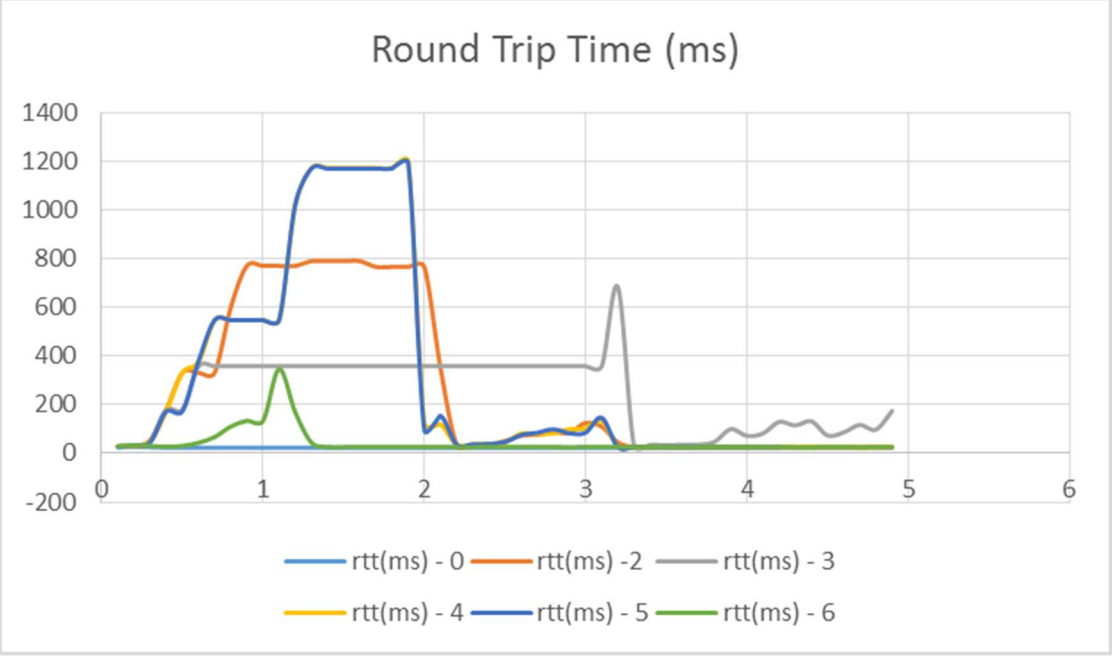


Figure 72 – TCP Round Trip Time (RTT)

8.6 Protocol Efficiency

NSIM captures the proportion of bandwidth used by each layer in the protocol stack, at specific points in the network. For these experiments, this specific point at which the data is captured is at the PON link between the ONU and OLT. We can then compare the proportion of protocol overhead used in the classical (Table 13) and FLATLANd architectures (Table 14). We see that each tunnelling layer (MPLS or 802.1Q) adds an additional 2.4% of overhead onto the overall data transferred at line level. In the FLATLANs case, the PON encapsulation protocol is emulated using 801.1Q. The protocol used both cases is UDP without background traffic.

Protocol	Bytes	Protocol Overhead	% Protocol Overhead	% Overhead wrt total
Ethernet(0)	1710596	138486	8.81%	8.10%
802.1Q(1)	1572110	41271	2.70%	2.41%
802.1Q(2)	1530839	40048	2.69%	2.34%
IP(3)	1490791	197384	15.26%	11.54%
UDP(4)	1293407	81296	6.71%	4.75%
Raw(5)	1212111	1212111		70.86%

Table 13 - Protocol Efficiency, Classic Architecture

Protocol	Bytes	Protocol Overhead	% Protocol Overhead	% Overhead wrt total
Ethernet(0)	1670850	138680	9.05%	8.30%
802.1Q(1)	1532170	42685	2.87%	2.55%
IP(2)	1489485	196667	15.21%	11.77%
UDP(3)	1292818	78955	6.50%	4.73%
Raw(4)	1213863	1213863		72.65%

Table 14 - Protocol Efficiency, FLATLANd Architecture

The useful application payload occupies between 70% and 72% of total traffic. In both cases, a significant amount of overhead (about 11.5%) is taken up by the IP protocol header. Thus FLATLANd does not add a substantial improvement to the protocol efficiency. Where instead it can contribute to improving network efficiency is in reducing the network operations required to switch traffic. We analysed this by using NSIM also profiles the router and switch operations performed during the simulation. This allows us to compare the switch performance in the classical architecture (Table 15) and the FLATLANd architecture

(Table 16). This shows the number of Ethernet forwarding operations, and MPLS/VLAN label switching/routing operations.

	Home router	ONU	CPE	Core Router	Metro Router	OLT	B-RAS	Total	%
eth fwd	4986							4986	8%
label pop		4986	4986	4995	4994	4989	4989	29939	46%
label push		4986	4986	4974	4980	4984	4984	29894	46%
								64819	100 %

Table 15 - Network operations - Classic Architecture

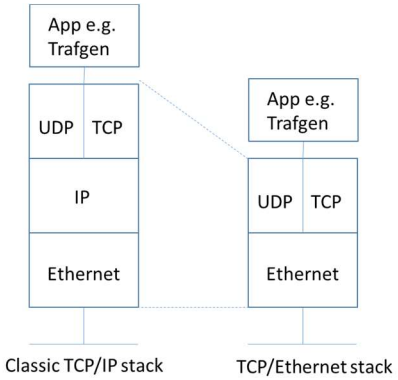
	CPE	ONU	OLT	Access Switch	Metro Switch	Core Switch	Total	%
eth_fwd	4978			4976	4973	4968	19895	50%
label_pop		4984	4986				9970	25%
label_push		4982	4980				9962	25%
							39827	100 %

Table 16 - Network operations - FLATLANd Architecture

The overall amount of network operations executed in the Classic Architecture is 40% higher than in FLATLANd. This is due to the additional VLAN tunnelling across the access network and the MPLS tunnelling across the metro-core network. The types of operations being executed also are more complex, with MPLS switching and routing, and VLAN pushing and popping being more complex to execute and more expensive in terms of calculation and processing (Table 1). The classic architecture must execute 59’833 complex (label pushing and label popping) operations as opposed to 19’932 complex operations by FLATLANd. The processing overhead for PPPoE is not accounted for.

8.7 New Protocols

Because it is possible to transmit packets over a wide area, using the FLATLANd model, the source and destination hosts can identify each other using the Ethernet addressing of their respective Network Interface Cards. For the purposes of routing traffic over FLATLANd, the IP address of each host end becomes redundant. We can envisage a collapsed network protocol stack where the IP layer of the TCP/IP stack is removed, and the TCP or the UDP layer communicates directly with the Ethernet Layer. TCP and UDP continue to provide the interface to the application layer for the purposes of end to end transport layer communication.



Because the IP header in the packet is not used for routing by any device within FLATLANd, the removal of the IP packet encapsulation does not affect the functioning of the FLATLANd network architecture. The advantages of removing a layer in the communications stack layer can have advantages. There is less packet processing required for encapsulation and de-encapsulation of packets. From our simulation measurements, the IP layer accounts for 11% of the total data exchanged at a line level. By removing the IP layer, Ethernet frames are shorter so less bandwidth is used to transfer data. In transferring the same application payload, switch and host buffers are less utilised, potentially leading to less network congestion. There is no requirement for Address resolution so as to determine which IP address is bound to which Ethernet Address, since the IP address layer would not exist any longer.

8.7.1 TCP over Ethernet (TCPoE)

Because, TCP over Ethernet (TCPoE) does not have a standard Ethertype, we need to create a new Ethertype (0x9998) and new binding between Ethernet and TCP in NSIM. Figure 73 shows the resulting TCPoE packet trace. There are no functional complications in NSIM with TCP communicating to the VLAN (802.1Q) layer as opposed to the Ethernet layer directly.

```
0000   99 88 77 66 55 44 11 22  33 44 55 66 81 00 00 46   ..wfUD."3DUf...F
0010   99 98 11 D4 00 50 2D 47  B8 4C 04 4B 7B CD 50 02   .....P-G.L.K{.P.
0020   20 00 00 00 00 00 41 42  43 44 45 46 47 48 49 4A    .....ABCDEFGHIJ
0030   4B 4C 4D 4E 4F 50 51 52  53 54 55 56 57 58 59 5A   KLMNOPQRSTUVWXYZ
0040   41 42 43 44 45 46 47 48  49 4A 4B 4C 4D 4E 4F 50   ABCDEFGHIJKLMNOP
0050   51 52 53 54 55 56 57 58  59 5A 41 42 43 44 45 46   QRSTUVWXYZABCDEF
0060   47 48 49 4A 4B 4C 4D 4E  4F 50 51 52 53 54 55 56   GHIJKLMNOPQRSTUV
0070   57 58 59 5A 41 42 43 44  45 46 47 48 49 4A 4B 4C   WXYZABCDEFGHIJKL
0080   4D 4E 4F 50 51 52 53 54  55 56 57 58 59 5A 41 42   MNOPQRSTUVWXYZAB
0090   43 44 45 46 47 48 49 4A  4B 4C 4D 4E 4F 50 51 52   CDEFGHIJKLMNOPQR
00a0   53 54 55 56 57 58 59 5A                           STUVWXYZ
###[ Ethernet ]###
  dst       = 99:88:77:66:55:44
  src       = 11:22:33:44:55:66
  type      = 0x8100
###[ 802.1Q ]###
     prio      = 0
     id        = 0
     vlan      = 70
     type      = 0x9998
###[ TCP ]###
        sport     = 4564
        dport     = http
        seq       = 759674956
        ack       = 72055757
        dataofs   = None
        reserved  = 0
        flags     = S
        window    = 8192
        chksum    = None
        urgptr    = 0
        options   = {}
###[ Raw ]###
          load      = 'ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXY
```

Figure 73 - TCP over Ethernet packet trace

8.7.2 UDP over Ethernet (UDPoE)

Similarly, because, UDP over Ethernet (UDPoE) does not have a standard Ethertype, we need to create a new Ethertype (0x9999) and new binding between Ethernet and TCP in NSIM. Figure 74 shows the resulting UDPoE packet trace.

```
0000   99 88 77 66 55 44 11 22  33 44 55 66 81 00 00 46   ..wfUD."3DUf...F
0010   99 98 14 E0 00 50 00 8A  00 00 41 42 43 44 45 46   .....P....ABCDEF
0020   47 48 49 4A 4B 4C 4D 4E  4F 50 51 52 53 54 55 56   GHIJKLMNOPQRSTUV
0030   57 58 59 5A 41 42 43 44  45 46 47 48 49 4A 4B 4C   WXYZABCDEFGHIJKL
0040   4D 4E 4F 50 51 52 53 54  55 56 57 58 59 5A 41 42   MNOPQRSTUVWXYZAB
0050   43 44 45 46 47 48 49 4A  4B 4C 4D 4E 4F 50 51 52   CDEFGHIJKLMNOPQR
0060   53 54 55 56 57 58 59 5A  41 42 43 44 45 46 47 48   STUVWXYZABCDEFGH
0070   49 4A 4B 4C 4D 4E 4F 50  51 52 53 54 55 56 57 58   IJKLMNOPQRSTUVWX
0080   59 5A 41 42 43 44 45 46  47 48 49 4A 4B 4C 4D 4E   YZABCDEFGHIJKLMN
0090   4F 50 51 52 53 54 55 56  57 58 59 5A               OPQRSTUVWXYZ
###[ Ethernet ]###
  dst       = 99:88:77:66:55:44
  src       = 11:22:33:44:55:66
  type      = 0x8100
###[ 802.1Q ]###
     prio      = 0
     id        = 0
     vlan      = 70
     type      = 0x9998
###[ UDP ]###
        sport     = 5344
        dport     = http
        len       = None
        chksum    = None
###[ Raw ]###
          load      = 'ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWX
```

Figure 74 - UDP over Ethernet packet trace

We can compare the performance differences between using the UDPoE approach and the previous UDP over IP approach, in executing the FLATLANd scenario 4. The traffic

generation sinks and sources in scenario 3 are configured to use the UDPoE stack (Figure 75)

```
host1=host_udp('host1',stack='udpoe') # Good traffic generator
host2=host_udp('host2',stack='udpoe',mdrop='00:00:00:00:00:00') # terminal, dropping fake traffic
host3=host_udp('host3',stack='udpoe',mdst='00:00:00:00:00:00') # fake traffic generator
```

Figure 75 - UDPoE host configuration

Figure 76 shows the overhead of each protocol layer when scenario 4 is run used UDPoE. Because the IP layer is no longer in the protocol stack. The payload now accounts for 85% of the data transmitted at an Ethernet line level. This is an increase from 70% in the original scenario 4.

Protocol Layer	Bytes	protocol Overhead (Bytes)	% protocol Overhead	Over as % of total frame
Ethernet(0)	2507490	199374	9.33%	7.95%
802.1Q(1)	2308116	59566	2.79%	2.38%
UDP(2)	2248550	111052	5.20%	4.43%
Raw(3)	2137498	2137498		85.24%

Figure 76 - UDPoE protocol stack utilisation

From Figure 77, we less effects from packet rate limiting (down 17%) and back pressure (down 16%) compared to the FLATLANd UDP over IP scenario 4. This results in a 25% improvement in delay and in 18% improvement in jitter.

Parameter	Value	Units
Send	5000	packets
Loss (pkts)	22	packets
Jitter (ms)	3.878	millisecond
Delay (ms)	234.762	millisecond
Goodput (bps)	1.59264	Bits per Second
Rate Limited (pkts)	1787	Packets
Back Pressure (pkts)	1853	Packets
Dropped Packets (Normal Queue)	28	Packets
Dropped Packets (Aged Queue)	0	Packets

Figure 77 - UDPoE performance for scenario 4

Chapter 9 Discussion

While the prospect of removing layers of legacy functionality can be attractive, the risk is that the resulting architecture can become simplistic. Buffers in network equipment are essential, particularly at the egress to long fat networks to accommodate Bandwidth Delay Product which is essential for the operating of TCP based application protocols. Likewise, the necessity for inter-layer co-ordination. While buffer size should be adequate, it should not be excessively large. Due to the availability of cheap RAM, large buffers can be configured needlessly at many interfaces in the network regardless of whether they are required or not. With the domino effect of back pressure, a temporary spike in traffic at the junction of high-speed and low-speed networks can rapidly fill successive upstream buffers. This can lead to unexpected sluggish response within a network which has more than adequate built-in capacity. There have been various queue discipline attempts at dealing with bloated buffers. The predominant solution is the CoDel queue discipline that drops queued packets older than a set age. In our tests on UDP and TCP streams, the FLATLANd architecture compared favourable against aged queues.

TCP is a problematic protocol in a network which has shared bandwidth. Not only is it adversely affected by bandwidth hogging by other (predominantly) UDP protocols, it depends on TCP intra-flow co-operation. However, there are different implementations of TCP, some of which operate in a bandwidth selfish manner. This is why it is essential in any network, or portion of a network where there is not unlimited bandwidth, to have Quality of Service or Traffic Management. This is traditional done at the IP network layer or the data plane.

Flow based QoS frameworks can apportion bandwidth in an equitable manner between flows, once flows can be defined. However, while flows can be easily identified and managed at the edge, it becomes a much harder issue to manage them as they become aggregated and concentrated in the core of the network. FLATLANd is unique in that it shares characteristics of the IP-layer flow-based QoS frameworks but it operates at a data plane level. The FLATLANd architecture is a hybrid between a flow-based network and a tree network, with a mix of the advantages and disadvantages of both. In our UDP and TCP comparative tests, we implemented Admission Control characteristic of the Flow Aware Networks.

QoS frameworks may be categorised by whether or not they allow applications to engage in the negotiation of a QoS characteristics. It can be a pointless exercise allowing applications to choose their QoS characteristics, since most end user applications do not expect, nor are they given, explicit QoS guarantees. This is despite basic mechanisms for traffic differentiation such IntServ and DiffServ existing in most modern routers. As a result,

many networks and applications continue to avail of 'best effort services' [91]. The absence of ubiquitous QoS profile implementation and simple and standardised interface/protocol, independent of network and geography has meant that developers typically do not implement QoS support into their applications. This is because requiring an application to run exclusively in an IntServ or DiffServ environment significantly reduces the target number of users. Xiao [126] shows that it is commercially difficult to introduce QoS into a network which works satisfactorily mainly due to over-provisioning. Even highly demanding applications can achieve sufficiently good QoS, providing that the access networks are not congested. Many major networks operators claim that their core network suffer from congestion. The pattern is to continually upgrade capacity of the basic services. While there are certain applications with well-known QoS profiles (such as VoIP) which are understood between developers and network providers, there is a significant hurdle to getting new and emerging applications such as multi-player games accommodated by network providers since the QoS requirements are often quite difficult to state explicitly. Most network operators do not support mechanisms for the dynamic provisioning of QoS for more recent applications. It is difficult to convince users to buy extra services while the standard service works adequately. This does not put pressure on telecom operators to introduce any differentiation mechanisms.

The impact and challenges in concentrating the ARP functions for an entire network in a small number of locations, should not be underestimated. In classic architectures, the function of ARP address resolution is distributed to each Layer 2 broadcast domain, in particular at the terminating LAN and WiFi networks. In total, the number of hosts generating ARP queries and seeking ARP responses for an entire network could run into the hundreds of thousands or millions per second. However, centralisation of ARP is an important network control network, already implemented in large Data Centres and can be quantified. In the Portland model [102], it is assumed that each ARP requires 25 microseconds execution time with an ARP timeout of 60 seconds and each ARP packet is 28 bytes long. Using the model proposed, for a Flatland network with 4 million terminating ONU's, each generating 1 Arp request per second, would require a 100 Core processors, which may be parallelized and distributed to 4 or 5 geographical areas in the network. In total, Arp queries and responses generates 896 Mbps of traffic. In this chapter, we have identified one solution to the issue of handling large volumes of ARP traffic. Through the elimination of the IP layer, the requirement for Address Resolution to map the IP addresses to MAC addresses is obviated. Table 17 is a brief synopsis of the features and benefits of the FLATLANd architecture. Many of these features and benefits are applicable to generic SDN based architectures.

The debate around the re-architecture of the Internet was initially split between two camps. There were those that wanted a big plan for a New Architecture for the Internet, which

would, at a specific point in the future, rectify all the issues with the Internet. Core to the future internet would be the regaining of the original experimental nature of the Internet which had been lost through ossification of technology and processes. The other camp saw a gradual and phased migration to the new Internet.

Core to the resolution of the debate was what type of discipline, such as engineering or computer science, should be used to address the design issues of the future Internet. There were misgivings that computer science was not suitable discipline even though many of the contributors to Internet standardisation bodies were computer scientists. Some question whether computer science is a bona fide science in the first place [30].

Prior to 1990, in the age of mainframe computers, the range of Computing related disciplines was narrow - focusing on Electronic Engineering (Hardware), Computer Science (Software) and Information Systems (Business). With the scope and scale of computing increasing in the 1990's, it was only natural that there would be additional categories added. The previous three areas have now been supplemented in the Computing Curricula [127] by Computer Engineering, Software Engineering and Information Technology, bridging the gap between technology and the business of end users (Figure 78).

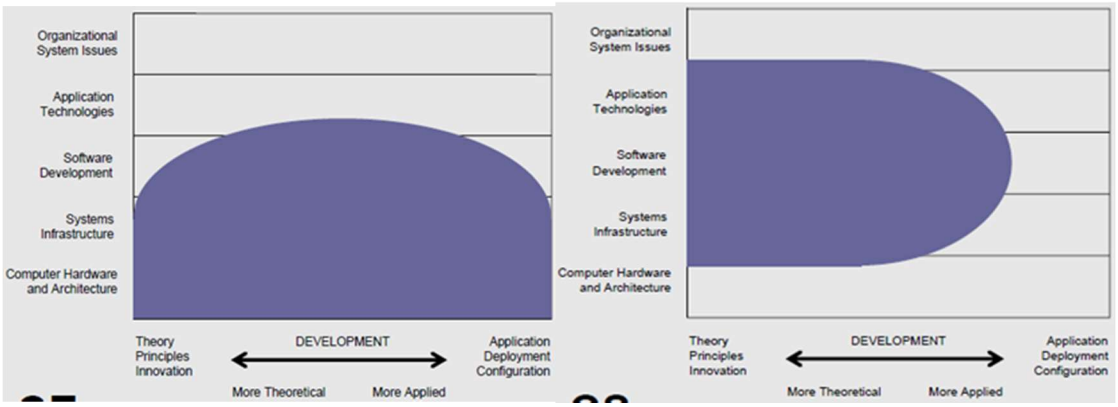


Figure 78 - (a) Computer Engineering Curriculum. (b) Computer Science Curriculum

Of most pertinence to the Internet New Architecture are the fields of Computer Engineering (Design and construction of computer based systems, digital hardware/software systems, embedded systems and integration of hardware/software) and Computer Science (Effective ways of solving computing problems, design and implementation of software, new ways to use computers). What distinguishes Computer Engineering from Computer Science is the former spans all theoretical and applied aspects of lower level technology, while computer science concentrates on more theoretical aspects of application, software development and infrastructure.

In theory, Internet Architecture should concern itself with the IP network and upwards of the TCP/IP stack, however it finds itself distracted with the issues such as bloated embedding of functionality within routers and switches. The approach being pursued to resolve the issues should be a combination of computer engineering and computer science. Once the

hardware and physical layer issues are resolved, then there is a reasonable prospect the Internet Architecture agenda can concern itself with Computer Science issues methods.

Over the course of 10 to 15 years since the New Architecture was first discussed, the dominant view has been that of those favouring a phased migration. While the structure of the Internet, in terms of processes, standards and architecture is consistent with the past, there has been steady adoption of SDN paradigms in technology islands such as data centres. The SDN approach to separation of the data and control plane has allowed styles and pace of innovation to be split also, and has, rightly, facilitated the use of computer engineering disciplines for the data plane and computer science techniques for the control plane.

If control plane design and development falls under the remit of the computer science discipline, the focus moves to the level of Software Development Lifecycle rigour that is applied to this development. In the past, networks have had some level of resilience to faulty design due to the autonomous nature of switches and routers. An upgrade of a network node, might cause affect a node or a domain, however, the rest of the network might still function. The hurdles to making changes in the network were very often physical, often requiring updates to remote central offices by on-site technicians. With SDN, the entire control of the network is centralised with the functionality defined by, for instance, YANG, COP and Openflow configurations residing in code repositories such as git. SDN upgrades to an entire can be effected with a push of a button. Alternatively, upgrade or changes can be rolled back, or the state of the network may be changed to a configuration at a specific point in the past. This may be judiciously or erroneously.

If there is some consolation, it is that there has been a similar revolution in other areas of computer science and Information technology, from which equivalence can be identified and approached learnt. There has been deprecation of mainframe technologies (equivalent of switches and routers), and the virtualisation of computing and storage (equivalent to virtualisation of networks and the adoption of SDN). The current best practice for the management of system functionality is the Continuous Delivery lifecycle (Figure 79) that is used to manage highly complex functionality such as the Linux Kernel. Both Linux and Git were initially developed and are currently maintained by Linus Torvalds.

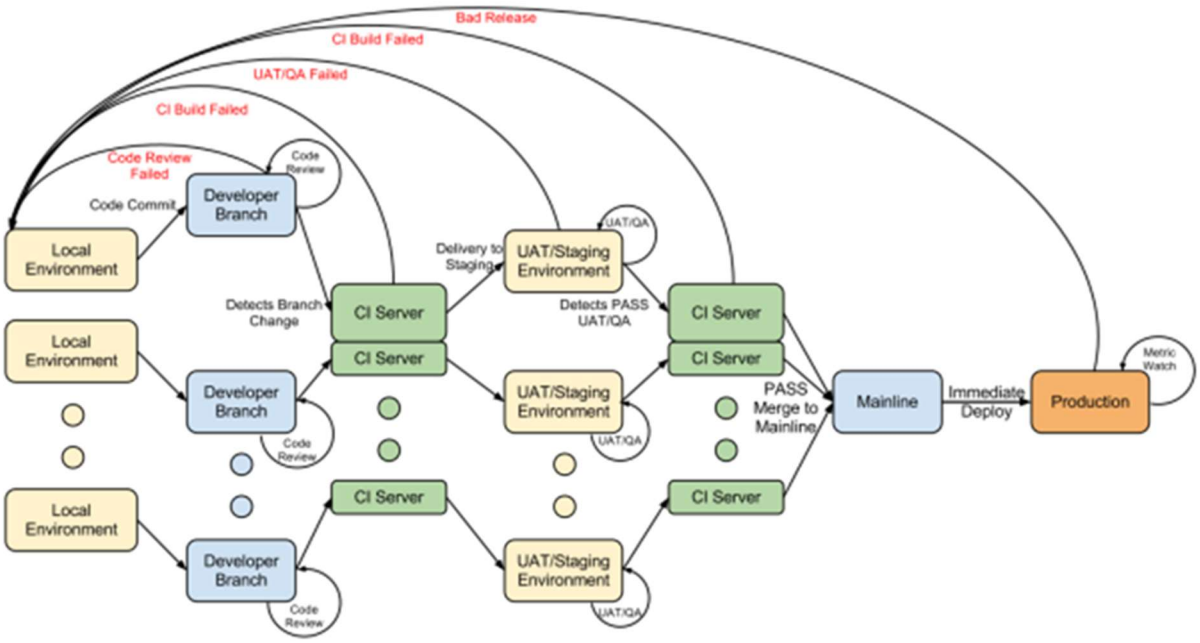


Figure 79 - Continuous Delivery life cycle [128]

For this regime to be applied to SDN based development, there needs to be multiple environments for Staging, User Acceptance Testing and Development that are identical to the production environment. These environments may be physical, however, preferably they should be virtualised so they can be created on demand, and in any quantity required by developers. The SDN code should be in a single git repository with separate branches for development, integration testing and production. With SDN functionality progressing from left to right in the diagram, there are gates through which the development must go through, the success of which demands on the execution of tests that test both existing functionality as well as new functionality. The rigour with which new functionality is introduced and bugs and faults are prevented is down to how detailed tests are defined. Preferably, tests should be automated and managed along with the SDN functionality residing in the git repository. There needs to be unit tests that test atomic functions and procedures. There needs to be system tests for aggregate system functionality, integration tests for validating interoperability between systems and User Acceptance tests for testing the fulfilment of business level requirements. Typically in a Test Driven Development (TDD) approach associated with Continuously Delivery, for every line of SDN code, there would typically be 3 lines of test code. The combination of CD, TDD and the easy availability of virtualised environments has led to the Dev Ops (Development Operations) that uses SDLC rigour to IT and Networks Operations Management. Given the complexity and the volume of changes that may be progressing through an SDN lifecycle, fortunately, there are Continuously Delivery, such as Jenkins, that can managed the process. In a CIO-envisaged environment [14], there would not be a distinction between SDN system delivery and delivery of functionality for Telecommunications OSS or BSS except that they related to different domains.

Area	Feature	Benefit	Beneficiary	Current situation
Network efficiency	Large Addressing Scheme	2^48, Much larger than is required for Internet of Things, forecasted to be 20 billion by 2020.	Network provider such as Telefonica, BT, France Telecom	this address space is much larger than current IPV4 Address space (2^32)
Network Security	End devices are identified directly at a lower level in the network i.e. at Layer 2	Secure. Services bind to devices rather than other way around.	Network providers. End Customers	Customer IP addresses are currently assigned by service providers. Identification of malicious activity is currently hidden through various layers of obfuscation
Service Provision	Binding between real and pseudo MAC addresses is controlled by the infrastructure, or a delegated party.	Speed. This binding is unique can be done quickly. Also, it can be removed/changed quickly.	Service providers Service provider – video platform, IOT platform	This is equivalent to setting up a subscription to a broadband service provider, which can take days/weeks/months to put in place.
Network Security	Binding between real and pseudo	Secure. This binding is unique and prevents	Service providers	Currently, black hat hackers can hijack identities and IP address sessions, through man-in-the middle attacks.

	MAC addresses is controlled by the infrastructure, or a delegated party.	duplication or take-over of mapping by third parties		
Network Convergence	MAC address translation facility is operated by infrastructure provider. As a common broker between higher level network providers and/or service providers	Utilisation of lower layer infrastructure resources is efficient and economic. Tenant network providers leases capacity required. This leads to much sought after Open Access.	Service providers	Network providers compete for the provision of infrastructure, leading to replication of infrastructure, and islands of low infrastructure in some areas, and over provision in others.
Network convergence	MAC address translation at the last hop in the network (GEM port) allows binding of services to be changed quickly.	Service characteristics can be carried with a device, if they move between locations or between termination nodes (ONU's). This allows services such as tablet or phone moving from a broadband line to a wifi or Cable modem, or another	Service providers	Currently there is no co-ordination between service providers. A customer has to have a separate contract, and authentication profile if they are in a different building or using a different access type (mobile, WiFi etc.)

		building, and carry service characteristics with them (e.g. speed, authenticated services, ip address etc.)		
Network convergence	MAC address translation can be used on any network that uses Ethernet as a layer 2 carrier.	Ethernet is ubiquitous and is found on all LAN's, WAN's, PON sub-layers, mobile LTE, DOCSIS cable TV networks, WiFi. The same principles as described for Optical networks can be applied to these networks. This supports principle of open access and efficient use of infrastructure.	Network providers. Service providers.	Current, network provider types (cable TV, Mobile, Wi-Fi___33, Broadband) must maintain their own infrastructure, leading to issue replication of infrastructure, and islands of low infrastructure in some areas, and over provision in others.
Internet of Things	A ubiquitous packet based Layer 2 network	Basic layer 2 service can be provided to the granularity of individual devices, without necessity of subscription to a service provider	Service providers. CPE manufacturers	Households or businesses must subscribe to a service provider before they access any services. This can be a lengthy process.
Network efficiency (Energy)	Service binding can be completed on a per-device	Subscription to services does not need to done at the level of an entire household or business premises, nor	Network providers. Service providers	Households or businesses must subscribe to a service provider before they access any services. Generally, there is only service provider, with a subscription which is long-term in the case of broadband.

	level or even a sub-device level.	does it need to be time-based. One device can be accessing a service (e.g. Video-On-Demand) from one service provider, at one quality of service. A separate device can be accessing a different service provider at a separate quality of service.		
Future proofness	Proposed structure addressing scheme is a sub-set of the (current) unstructured addressing scheme	Service binding at layer 2, is backwardly and forwardly compatible with existing layer 3 services	Network providers	
New Services	Structured addressing	Traffic flows can be aggregated according to customer, device, locality, city, town, service provider, service type. Flows can be metered and controlled in real time.	Network providers. Service providers	It is difficult for service providers to identify traffic according to customer, device, locality, city, town, service provider, service type. This can lead to over and under-provisioning of network resources in different parts of the network. Where service providers do control bandwidth, it is difficult to alter i.e. time-of-day or customer profile based.

		Networking traffic engineering and planning can be done efficiently and economically.		
Network efficiency. New Services	Structured addressing and binding	The customer network environment can be extended to include a centralised data centre portion. This allows services (such as firewalling, parental access control) to be hosted in a virtual manner, efficiently and securely by the service provider	Network provider	Current Customer Premises equipment are low specification and only adequate for purpose. They can become out of date quickly, and difficult to maintain, usually be customers themselves who don't have the skillset.
Service Quality	Layer 2 network	Low latency, less buffering of traffic, lower jitter, Higher quality of high speed transmission.	End Customers	Every device in current networks that operates at layer 2, 3 etc. has buffers to reduce packet loss and assist with flow control. These devices include broadband modems, router cards/ports, DSLAM's, B-RAS's etc. This leads to the phenomenon known as Bufferbloat.
Network efficiency	Layer 2 network	Much more cost efficient Ethernet ports. 48x 10Gb ports Openflow switch = €8k.	Network provider	Nexus 7000 - €540k

Table 17 - Features and Benefits of SDN FLATLANd architecture

Chapter 10 Conclusions and future work

We have shown the principles of Software Defined Networking can be applied at different points and levels within a Telecommunications Networks, almost mutually exclusively of each other. We have demonstrated quite different applications of SDN in the physical (optical) layer, as well in the higher (level 2 and beyond) layers. This undoubtedly benefits the phased introduction of SDN into legacy architectures. However, as a concept, for the full benefits of SDN, there should be complete separation of control plane and data plane for all components at all layers in the network architecture. We initially look at how successful are the application of SDN at each of the two distinct layers.

We devised and tested an end-to-end 1:1 protection scheme for a combined LRPON access and core network, using a multi-tier Control Plane over a Pan-European network. We achieved fast recovery within 7.2ms with subsequent core traffic redirection in 117 milliseconds across the metro core network. We predicted that using loosely coupled multi-tier controllers with dedicated links could reduce total link outage to 41 milliseconds.

In our first 1:1 protection experiment (see section 5.1), full recovery took place over an elapsed time period of 124 milliseconds which was composed of 3 individual time periods - a period in which traffic in the access was failed over from the primary path to the secondary path (7.2 milliseconds); a period in which core traffic was being redirected before the service could be restored (25 milliseconds); an intervening period in which the end to end link was in flux (92 milliseconds). The bulk of the 92 milliseconds was caused by two factors: link latency and the synchronous sequential update of the Openflow rules in the four Openflow switches along the backup path in the core.

The time lag between the access control plane sensing the failure and the controller in the core receiving the trigger over the Internet connection was measured at 70 milliseconds. Much of the elapsed time was taken up by the time to transmit the instruction between the controller and node, and an acknowledgment to be received. We the elapsed time of 67 milliseconds for the controller to update the flow rules across the secondary path, could be reduced significantly if the instructions could be issued asynchronously or in parallel by multi-threaded dispatcher. This would become a function of the longest node update time between the controller and a node (in this instance, 18 milliseconds). For typical sized countries using dedicated links between the access and the core, we felt that the total elapsed time for recovery could be reduced to 41 milliseconds.

We optimised the failure detection mechanism in our first N:1 experiment so that restoration time of the data traffic was occurred on average 81.29 milliseconds across a Pan European network. In our second N:1 experiment (see section 5.2), we included a PON physical layer, backup OLTs were shared among PONs in an N:1 scheme. The average protection time

was measured at 64 milliseconds, with variations between 50 and 100 milliseconds attributed to the random delay in the failure detection. In our third N:1 experiment, we optimised the failure detection response and achieved an average restoration time of 41 milliseconds across 70 measurements. Wwithin 15 milliseconds of the failure, the optical and electronic switch components and the backup OLT have been instructed to reconfigure their protection paths. Within 33 milliseconds after the failure, the electronic switch components within the core and access are configured, and by 38 milliseconds, the optical switch component is configured. In order to understand the effect of centralising both the Network Orchestrators and the Network Controllers, we compared the above results with the case where orchestrator and controllers are collocated within the core network. This was accomplished by setting the emulated intra-control plane latencies at zero. The results are shown in Figure 28 as the basic protection line. On average, basic protection can be accomplished within 27.8 milliseconds.

Overall, the experiments were successfully in demonstrating that SDN based path protection can be achieved well below the target switch over time of 50 milliseconds which are common for leased line traffic or 100 milliseconds for realistic internet scenarios [17]. Unlike Multiprotocol label switching (MPLS) which executes protection through an alternative Label Switched Path (LSP) at each switch along a path in the core network, our experiments demonstrated a co-ordinated control plane approach that can be centrally defined. Open shortest path first (OSPF) can take considerably longer to route, up to 1 or 2 seconds to route through a shortest path. The number of comparative case studies of path restoration in metro access networks are sparse. In 2008 an experiment was carried out using commercial GPON hardware and the restoration time was found to be in the order of 30s [19]. The authors of this experiment believed this could be reduced to approximately 500 milliseconds if they could optimise the switching, ranging and registration mechanisms of the GPON system. The same operator published in 2013 an updated protection mechanism using VLAN switching with an automated restoration solution, achieving protection times in the order of 4.5 s (with maximum values of 9.5 s) [20].

We presented a flat layer 2 architecture for telecommunications networks that allows removing many components traditionally active in telecommunications architectures, while still retain much of the functionality for access and the delivery of service. The benefits of the flat layer-2 approach are exemplified by contrasting today's (Figure 1) and the proposed (Figure 50) architectures. There is a flattening of layers within the access and metro portions of the network, with some functions, such as B-RAS and PPPoE terminating modems being made redundant, and other network functions such as AAA (Authentication, Authorization and Accounting) services being virtualised at the periphery of the network, following a Network Function Virtualisation approach. With the elimination of functions in some instances, and the virtualisation of functions within a property run datacentre, there is

potential for significant Capex and Opex improvements through reduced Operations and Maintenance. This is facilitated through the adoption of white-boxes Openflow-based switches controlled by a unified SDN control plane.

The FLATLANd architecture can function entirely at a layer 2 network and is inherently Open Access in that the roles of infrastructure provider, network provider and service provider can be clearly demarcated. All terminating devices can be granted access to the network and at any time be dynamically or statically bound to the profile of a target service provider. Indeed the flexibility of the addressing scheme favours multi-tenancy, since parts of the address can be used for packet routing purposes and other parts for QoS and SP differentiation. Distinct flow rules are created for the metering of each class of traffic at each Metro-Core node, OLT and ONUs. These are separate from the rules necessary or forwarding flows. We demonstrate that the functionality of registration and binding devices to the FLATLANd service are successful. A registration time of around 30 milliseconds was achieved for the LR-PON based scenario shown in Figure 58. While such operations are generally not time-critical, these results demonstrate the type of benefits that a simplified SDN-driven flat architecture can bring about. Once registration was complete, we successfully transmitted traffic between the client and Data Centre end-points. The traffic included both typical HTTP web traffic, but also less conventional Ethernet frames more suited to the transit of IoT device traffic.

Since a number of layers (such as PPPOE tunnelling) and component stacks (such as Broadband Access Services) are removed, there is less requirement for authentication and authorisation across junctions between these layers. This has potential for much savings in Opex and Capex through reduced equipment plant in the metro and access networks. With less layers (such as PPPoE tunnelling) and component stacks (such as Broadband Access Services), the requirement for cross-layer authentication and authorization is greatly reduced. In addition, the FLATLANd architecture provides a separation between the provision of infrastructure, network services and Internet services by distinct entities, potentially enhancing efficiency of use of resources.

We conclude that SDN has facilitated the separation of Telecommunications networks into a domain constrained by physics (data plane) and a domain liberated by software engineering (control plane). Together, they have enabled fresh approaches to the provision of network services. Whilst the initial results for SDN base Telecommunications networks are positive, the bridge between proof of concept and production solutions will require rigour of defining adequate use cases for regression and future network services. The level of adequacy will prevent over-simplification which could lead to poor solutions.

10.1 Recommendations for future work

The following are selected recommendations for future work.

NSIM Enhancements.

- NSIM has been developed and employed to test specific scenarios outlined in this paper. However, NSIM is a generic simulator which can be enhanced in a number of directions. NSIM has a functional TCP protocol implementation which is a rare example of an implementation written as part of a Python based simulator. While the implementation follows the standards, it can be enhanced to include other features, such as Window Scaling, that appear in the Linux kernel. This would make NSIM of interest to wider community users.
- The base Duplex block has been sub-classed to implement network blocks such as switches and routers, as well as physical layer components such as transmission lines. NSIM can be enhanced to include other physical (optical) layer characteristics within DWA's, EDFA's and Optical switches. This would allow NSIM to simulate realistic, full stack networks.

Network convergence

Within this thesis, we evaluated the effects of SDN, separately, on the two distinct layers - converged network and flat layer two. It was not possible to fully evaluate both SDN controlled layers working together. However, the combination of converged network, flat layer two and IP-less network could provide ubiquitous high speed resilient network (converged network), open access granular services (FLATLANd) and protocol efficiency (TCPoE). A common SDN controller for these layers could provide more insight into customer demand and service quality. A study would compare the performances of existing networks and further converged architectures.

Chapter 12 Appendix

Glossary

ABNO Application Based Network Operations

ACK Acknowledgment (packet)

API Application Programme Interface

ARP Address Resolution Protocol

ATM Asynchronous Transfer Mode

BGP Border Gateway Protocol

B-RAS Broadband Access Server

CAC Call Admission Control

CD Continuous Delivery

CBR Constant Bit Rate

DAE Digital Agenda Europe

DARPA Defence Advanced Research Projects Agency

DISCUS Distributed Core for unlimited bandwidth supply for all users and services

DSCP DiffServ Code Point

DWA Dynamic Wavelength Assignment

DWDM Dense Wave Division Multiplexing

EU European Union

FPGA Field Programmable Gate Array

FTP File Transfer Protocol

FTTH Fibre to the Home

GENI Global Environment for Network Innovations

HTTP Hypertext Transfer Protocol

IEEE Institute of Electronic and Electrical Engineers

IETF Internet Engineering Task Force

IETF Internet Engineering Task Force

IP Internet Protocol

JSON JavaScript Object Notation

LR-PON Long Reach Passive Optical Network

MPLS Multiprotocol Label Switching

MPLS Multiprotocol Label Switching

MTU Message Transfer Unit

NAT Network Address Translation

NFV Network Function Virtualisaion

NNI Network to Network Interface

NSF National Science Foundation

OLT Optical Network Termination

ONF Open Network Foundation

ONU Optical Network Unit

OSI Open Standards Institute

OSPF Open Shortest Path First

PCE Path Computation Element

PCEP PCE protocol

POTS Plain Old Telephone Service

QoS Quality of Service

RAM Random Access Memory

RFC Request For Comment

RFC Request For Comment

RTT Round Trip Time

SDN Software Defined Networks

SDN Software Defined Networks

SFP Small Form Factor

SMTP Simple Mail Transfer Protocol

SQL Structured Query Language

SSH Secure Shell

TCAM Ternary Content Addressable Memory

TCP Transmission Control Protocol

TDD Test Driven Development

TDM Time Division Multiplexing

TED Transmission Database

UBR Unspecified Bit Rate

UDP User Datagram Protocol

UNI User Network Interface

VBR Variable Bit Rate

VLAN Virtual Local Area Network

VOIP Voice over IP

VPN Virtual private network

VPN Virtual private network

XFP Extended Form Factor

XML Extended Mark-up language

Chapter 1.

FGPA 1

TCAM

The following is a description of TCAM is programmed on the FGPA. The upstream and downstream systems work in a similar way but are both programmed separately. Both are programmed via the OLT.

Both downstream and upstream appear to be working from my tests but until you get a full scenario working I won't know for sure how you want to use it. A very simple pseudo code version of what happens is the following

Downstream: Data_in is packet from backplane Data_out is packet heading to PON. ONU_out is the ONU on PON to address data to. (Ref)

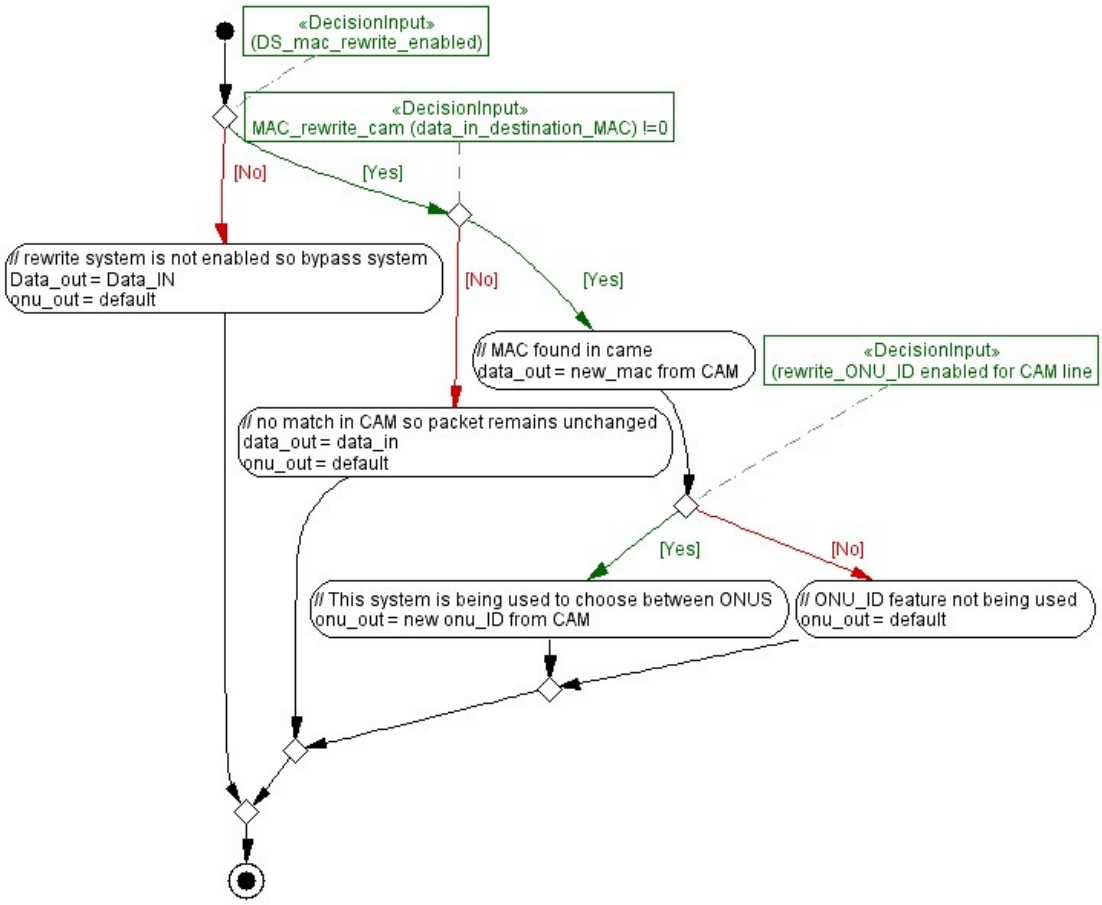


Figure 80 - Downstream TCAM

As can be seen, any packets that don't match a CAM rule are passed through unchanged. However if a CAM rule is found to match the MAC address is rewritten. If multiple CAM rules are found to match the last one is used. I.e. if rule 0 and rule 15 match then rule 15 will apply.

Upstream US is very similar to downstream except that source XGEM can now be used as a matching criteria together with destination MAC (Ref)

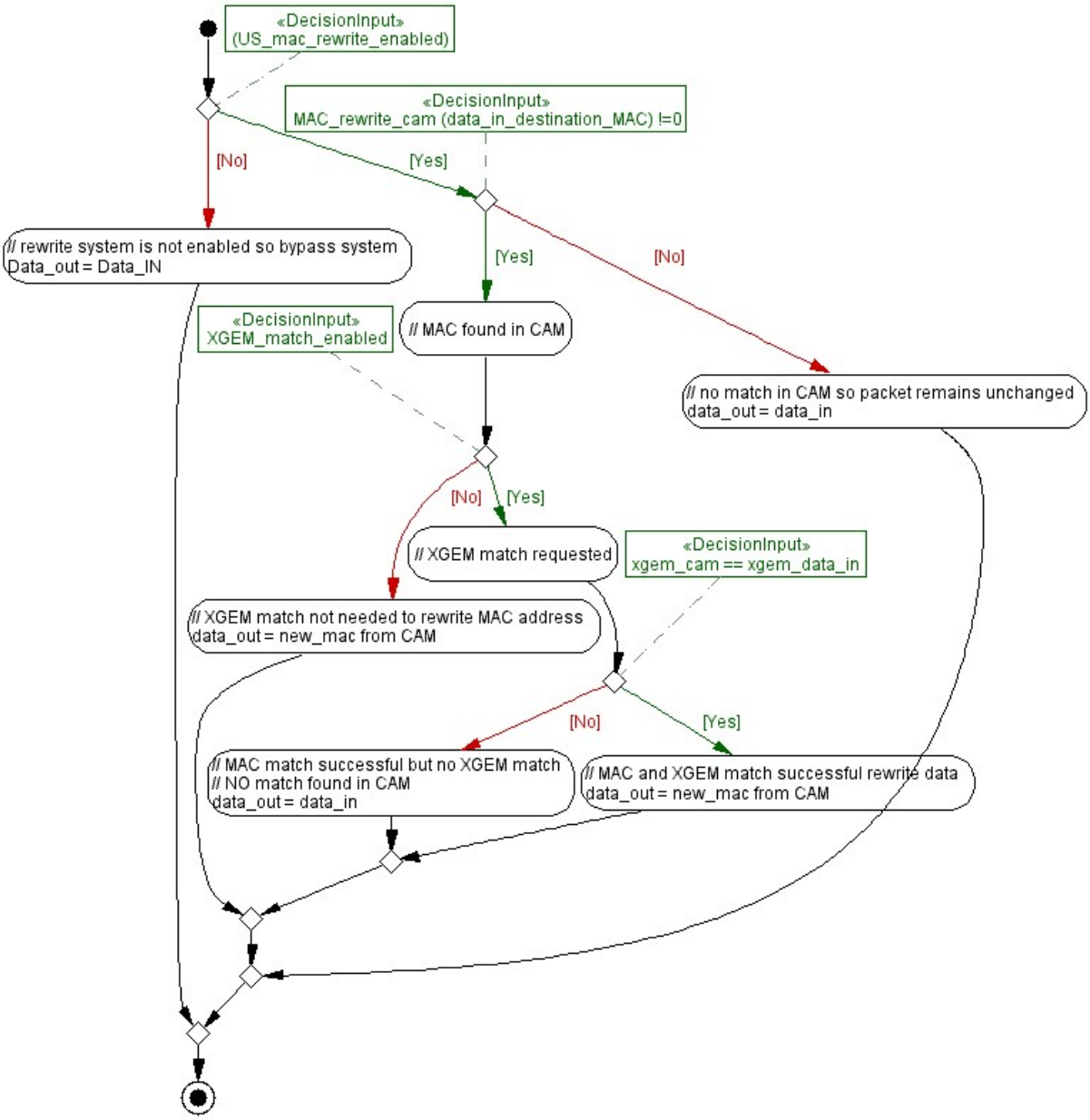


Figure 81 - Upstream TCAM

Controls

0xc020 = enable mac rewrite
0xc021 = LSB MAC to be replaced
0xc022 = MSB MAC to be replaced
0xc023 = LSB new MAC
0xc024 = MSB new MAC
0xc025 = corresponding xgem label (can be applied DS -since we have one ONU it is not necessary however it will be used US to map real to pseudo)
0xc026 = command 16bit <4 bit unused> <4 bit CAM address><6bit unused><1 bit line active><1 bit replace xgem DS >

When enabled any incoming packet with matching MAC will be replaced with new MAC address and if replace xgem option is set the Xgem address is also changed. Since in the test, there is only one ONU this isn't important. Any packets that do not match any of the 16 rules will pass through unaffected

Where a frame has a destination mac address of, for instance, 11:22:33:44:55:66 and we want to be mapped to aa:bb:cc:dd:ee:ff and xgem/alloc_id 0004. The following commands are issued :

wc021:33445566
wc022:1122
wc023:ccddeeff
wc024:aabb
wc025:0004
wc026:0003 // address 0, enable line of cam(1bit), enable xgem rewrite (1 bit),

// The above will do nothing until the whole system is enabled
wc020:1 // enable the mac switch feature

Now if a second line is added to the cam to rewrite 12:34:56:78:90:ab to 10:20:30:40:50:60 and no modification to xgem. Since CAM line zero is full we put this in cam line 1

wc021:567890ab
wc022:1234
wc023:30405060
wc024:1020
wc025:0006 // value here will not be used
wc026:0102 // address 0, disable xgem rewrite, enable line 1 of cam

Finally to delete any line of the cam simply write to c026 with the address of cam line and command 0

so to delete line 1
wc026:0100

The US registers are identical only the address is at 20 - 26 instead of c020-c026. So US enable for example would be w20:1.

Context for TCAM + tests

FPGA 1

```
if(DS_mac_rewrite_enabled) then
   if MAC_rewrite_cam (data_in_destination_MAC) !=0 then
     // MAC found in came
     data_out = new_mac from CAM
```

```
      if(rewrite_ONU_ID enabled for CAM line then
        // This system is being used to choose between ONUS
        onu_out = new onu_ID from CAM
      else
        // ONU_ID feature not being used
        onu_out = default
    end if
    else
      // no match in CAM so packet remains unchanged
     data_out = data_in
     onu_out = default
   end if
 else
  // rewrite system is not enabled so bypass system
  Data_out = Data_IN
  onu_out = default
end if
```

FPGA 2

```
 if(US_mac_rewrite_enabled) then
    if MAC_rewrite_cam (data_in_destination_MAC) !=0 then
      // MAC found in CAM
      if XGEM_match_enabled then
        // XGEM match requested
        if xgem_cam == xgem_data_in then
          // MAC and XGEM match successful rewrite data
          data_out = new_mac from CAM
        else
          // MAC match successful but no XGEM match
          // NO match found in CAM
          data_out = data_in
        end if
      else
        // XGEM match not needed to rewrite MAC address
        data_out = new_mac from CAM
     end if
    else
      // no match in CAM so packet remains unchanged
     data_out = data_in
    end if
 else
  // rewrite system is not enabled so bypass system
  Data_out = Data_IN
end if
```

Internet Statistics

	UK (2015)	UK (2020)
Internet users	56	62
Devices per person	5.7 devices	
Average fixed broadband (Mbps)	24.7	51.3
Average Wi-Fi speeds (Mbps)	17.4	35
Internet Traffic per month per user (GB)	40	93.9
IP Traffic per annum per user (GB)	45	113
Internet Traffic per annum per user (GB)	35	100
Internet Traffic per month per House (GB)	84.5	202.1
average FTTx Internet household per month (GB)		608.5 (203.2% more than other broadband households)
Devices (M)	368.3	660.3
Consumer IP VOD traffic, per month	514 Petabytes per month (18% of Internet traffic, 21% of consumer IP traffic)	745 Petabytes per month (10% of Internet traffic, 12% of consumer IP traffic)
Consumer Internet Video, per month (ExaBytes)	1.3	4.7
Consumer Fixed Internet per Month (ExaBytes)	1.9 (77% of Consumer IP traffic, 65% of total IP traffic)	5.2 (81% of Consumer IP traffic, 68% of total IP traffic)
Consumer IP traffic per Month (ExaBytes)	2.5 (85% of total IP traffic)	6.4 (85% of total IP traffic)
Fixed/Wired IP traffic per month (Exabytes)	1.4 (49% of IP traffic, 37% of total internet traffic)	2.5 (33% of IP traffic, 25% of Internet traffic)

Fixed/Wifi IP traffic per month (Exabytes)	1.4 (48% of IP traffic, 59% of Internet traffic)	4.5 (59% of IP traffic, 66% of Internet traffic)
IP traffic per month (Exabytes)	2.9	7.6
Average Internet traffic	5 Tbps	20Tbps (2.9 fold increase)
Busy Hour Internet traffic	21 Tbps	117 Tbps (4.5 fold increase)
Internet Video per month		121 Bn Minutes
Internet Video traffic per month (Exabytes)	1.5	5.4
Gaming Traffic per month (Exabytes)	1.5	5.4

Table 18 - Internet Statistics

Definitions

Transponders

Transponders are devices that provide bidirectional conversion from one optical wavelength to another, typically from/to a grey (1300 nm) optical signal to a DWDM-band (1500 nm) specific wavelength optical signal. Transponders can be considered as two back-to-back transceivers. The (grey) client side interface typically has limited reach (e.g. up to 2km, 40km, or 80 km), whereas the line side interface typically has longer reach (e.g. 200km, 500km or 2000 km) given the appropriate amplification

Transceivers provide full-duplex conversion from/to an electrical signal to/from an optical signal. They are typically commercially available in standardized enclosures such as SFP (1G) and XFP (10G), XENPAK (10G), CFP (100G)

TCP

TCP 793 . TCP (Transmission Control Protocol) is a network communications transport protocol that provides a number of services for application and higher layers in the TCP/IP network architecture stack . It is one of the dominant transport protocols that use the IP network layer to provide a logical end to end transport service for application data. The alternative to TCP is UDP (User Datagram Protocol). While UDP is a lightweight connectionless protocol that does not preserve state nor sequence between protocol packets, TCP provides a guarantee that a stream of bytes sent from the sender application on one host is delivered reliably and in the same order to the receiver application on the other host to the application layer. The key features that set TCP apart from UDP include Retransmission of lost packets, Ordered data transfer, Error-free data transfer, Congestion control, Flow control. Examples of applications that rely on the robust transmission characteristics of TCP include SMTP, HTTP, SSH and FTP. Examples of applications that avail of the alternative more lightweight characteristics of UDP include VoIP (Voice over IP) and Video Streaming. UDP based applications either do not require guaranteed delivery of packets and thereby survive packet-loss, or they provide their own equivalent of a stateful transmission protocol within the application layer. The implication being that there are characteristics of TCP which have undesirable performance overhead or latency, since TCP waits for retransmissions of lost messages or reorder out-of –order messages. TCP is a bidirectional protocol which allows two hosts to transmit data in packets asynchronously to each other. For each packet is sent from one host, an acknowledgement (ACK) packet must be received. The ACK packet indicates the next sequence number that the remote host is expecting. Congestion episode occurs if there are three duplicate acknowledgments or after a timeout.

Quality of Service

QoS (Quality of Service) refers to the quality of transmission through a Network System such as a Metro Node. Typical QoS parameters are level of throughput, packet loss, packet

delay, jitter and amount of errors induced [79]. Each of these is an indicator of underlying issues related to design, configuration or presence of faults. Bandwidth and throughput indicate the number of packets that flow through a network every second. The type of media guide that carries the traffic as well as configuration (for example, clock speed) of a terminating port interface have significant bearing on the throughput. Faulty terminations, reflections or a mismatch between packet MTU sizes between terminating interfaces increases packet discards, errors (for example, the Bit Error Rate) and as a consequence the Packet loss ratio. Packet Delay or latency is the average or maximum delay in sending packets in a single direction, or round-trip. Jitter is the level of lack of stability in the packet delay, which can have a detrimental effect on real-time applications such as VOIP (voice-over-ip).

Class of Service

CoS (Class of Service) is the mechanism by which upper layer protocols and applications direct how the lower network layers should carry or handle traffic. In contrast to QoS which is a finely grained traffic control mechanism. CoS is a coarsely-grained traffic control which scales better as a network grows in size and complexity. CoS aggregates a group of flows which are similar in characteristics such as bulk data transfer, video streaming or sporadic email traffic, and assigns a set of class-specific rules to each traffic type. The delivery time and bandwidth assigned to a traffic type is not guaranteed and is offered on a best-effort basis. Class of Service is alternatively called Type of Service. There are a number of examples of how Telco grade services and IETF RFC related services define Class of Service [129]. The parameters used by ATM (Asynchronous Transfer Mode) to distinguish different classes of service include VBR (Variable Bit Rate), CBR (Constant Bit Rate), Available Bit Rate, Guaranteed Bit Rate and UBR (Unspecific Bit Rate). UBR is alternatively called Best Effort [130]. IEEE proposed the 802.1p Layer 2 Tagging which uses a 3-bit field called the Priority Code Point (PCP) within an Ethernet frame header when using VLAN tagged frames as defined by IEEE 802.1Q. It specifies a priority value of between 0 and 7 inclusive that can be used by QoS disciplines to differentiate traffic. The IETF Type of Service field (ToS) field is a six-bit Differentiated Services Code Point (DSCP) field and a two-bit Explicit Congestion Notification (ECN) field in the IPv4 header While DiffServ is somewhat backwards compatible with ToS, ECN is not. The ToS field can be used to specify a datagram's priority and request a route for low-delay, high-throughput, or highly-reliable service. Based on the ToS values, a packet would be placed in a prioritized outgoing queue or take a route with appropriate latency, throughput, or reliability. In general, the type of service (ToS) field has been defined in different ways RFCs and in practice, the it has not been widely beyond experimental networks.

Comreg

Comreg is the regulator for electronic communication (telecommunications, radio communications and broadcasting) and postal sectors in ireland. Further to Regulation No. 2887/2000 of 18 December 2001 of the European Parliament and of the Council on unbundled access to the local loop, Comreg was set up through the 2002 Communications Regulation Act, replacing the previous Office of the Director of Telecommunications Regulation (ODTR). Section 12 of the Act details Comreg's objectives with regards to electronic communications, that is, to promote competition, to contribute to the development of the internal market, and to promote the interests of users within the Community.

Bibliography

[1] F. C. Europe, "The Business Case of Incumbent Telco Fiber Networks," *Heavy Reading*, 2009].

[2] V. N. I. Cisco, "Global mobile data traffic forecast," Feb, 2013.

[3] E. Commission, "Digital Agenda For Europe," 2010].

[4] J. Roberts, "The clean-slate approach to future Internet design: a survey of research initiatives," *Annals of telecommunications,* vol. 64, no. 5, pp. 271-276, 2009.

[5] B. Briscoe, "Flow rate fairness: Dismantling a religion," *ACM SIGCOMM Computer Communication Review,* vol. 37, no. 2, pp. 63-74, 2007.

[6] D. D. Clark, S. Shenker, and A. Falk, "GENI research plan," *GENI Design Document. Research Coordination Working Group*, 2007.

[7] R. Braden, D. Clark, S. Shenker, and J. Wroclawski, "Developing a next-generation Internet architecture," *White paper, DARPA*, 2000.

[8] G. Parulkar, "Reinventing the internet – Platforms for innovation. ," Stanford University, 2009.

[9] D. Clark, R. Braden, K. Sollins, J. Wroclawski, and D. Katabi, *New Arch: future generation internet architecture*, DTIC Document, 2004.

[10] T. Anderson, L. Peterson, S. Shenker, and J. Turner, *Report of NSF workshop on overcoming barriers to disruptive innovation in networking*: National Science Foundation, 2005.

[11] NSF. "Future Internet FIND project," http://www.nets-find.net/.

[12] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future internet research and experimentation: the FIRE initiative," *ACM SIGCOMM Computer Communication Review,* vol. 37, no. 3, pp. 89-92, 2007.

[13] NICT. "JGN+ AKARI," http://akari-project.nict.go.jp/eng/concept-design/AKARI_fulltext_e_translated_version_1_1.pdf.

[14] G. Parulkar, "Case for rethinking the Internet Architecture : some promising approaches," Stanford University, 2010.

[15] M. Ruffini, "Discus: An End-to-End Solution for Ubiquitous Broadband Optical Access," *IEEE Communications Magazine*, 2014.

[16] M. Gupta, and S. Singh, "Greening of the Internet," *Proceedings of the 2003 conference on Applications, Technologies, Architectures and Protocols for Computer Communications - SIGCOMM'03*, pp. 19-26, 2003.

[17] M. Ruffini, "Deployment Strategies for Protected Long Reach PON," *JOCN,* vol. 4, 2012.

[18] N. Kitsuwan, S. McGettrick, F. Slyne, D. B. Payne, and M. Ruffini, "An Independent Transient Plane Design for Protection in OpenFlow-based Networks," *Journal Optical Communications and Networks,* vol. 3, no. 2, 2014.

[19] J. Kang, "Restoration of Ethernet Service over a Dual Homed GPON System," *OFC*, 2009.

[20] A. Rafel, "Automatic Restoration over a Type B Dual Parented PON using VLAN Switching."

[21] M. Ruffini, "Protection Strategies for Long Reach PON."

[22] J. Gettys, "Bufferbloat: Dark Buffers in the Internet," *IEEE Internet Computing,* vol. 15, no. 3, pp. 96,95, May/June 2011, 2011.

[23] V. Cerf, Y. Dalal, and C. Sunshine, *Specification of internet transmission control program*, INWG General Note, 1974.

[24] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. S. Wolff, "The past and future history of the Internet," *Communications of the ACM,* vol. 40, no. 2, pp. 102-108, 1997.

[25] F. Slyne, and M. Ruffini, "FLATLANd: A Novel SDN-Based Telecoms Network Architecture Enabling NFV and Metro-Access Convergence. ," in ONDM, 2016.

[26] P. D. Leedy, and J. E. Ormrod, *Practical Research: Planning and Design*: Pearson education international, 2001.

[27] H. J. Holz, A. Applin, B. Haberman, D. Joyce, H. Purchase, and C. Reed, "Research Methods in Computing: What are they, and how should we teach them?." pp. 96-114.
[28] I. Vessey, V. Ramesh, and R. L. Glass, “A unified classification system for research in the computing disciplines,” *Information and Software Technology,* vol. 47, no. 4, pp. 245-255, 3/15/, 2005.
[29] A. Ginige, “Research Methods in Computing,” 2008.
[30] R. L. Glass, V. Ramesh, and I. Vessey, “An analysis of research in computing disciplines,” *Communications of the ACM,* vol. 47, no. 6, pp. 89-94, 2004.
[31] J. Matias, E. Jacob, N. Toledo, and J. Astorga, “Towards Neutrality in Access Networks: A NANDO Deployment with OpenFlow,” 2011.
[32] O. N. Foundation, 2016.
[33] D. King, J. Fernandez-Palacios, O. D. Dios, and V. Lopez, “using the path computation element to enhance sdn for elastic optical networks (eon),” 2013.
[34] EU-FP7. "CaON positioning paper "; http://caon.i2cat.net/wp.../CaON_positioning_paper_final_v0.1.docx.
[35] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review,* vol. 38, no. 2, pp. 69-74, 2008.
[36] J. Rexford, M. Caesar, N. Feamster, and D. Caldwell, “design and implementation of a routing control platform,” 2004.
[37] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, “Ethane: taking control of the enterprise,” *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, ser. SIGCOMM’07 ACM*, pp. 1-12, 2007.
[38] A. Bianco, L. Giraudo, and R. Birke, “Openflow switching: Data plane performance,” 2010.
[39] A. Farrel, J. P. Vasseur, and J. Ash, “RFC 4655 - A Path Computation Element (PCE)-Based Architecture,,” 2016.
[40] IETF, "I2RS PCEP Extension for Stateful PCE. Draft-ietf-pce-stateful-pce-08-Work-in-progress," 2015.
[41] L. Velasco, A. Castro, D. King, O. Gerstal, R. Casellas, and V. Lopez, “In Operation Network Planning,” *IEEE Communications Magazine*, 2014.
[42] IETF, "I2RS Interface to the Routing Systems (i2rs) - Charter for Working Group - charter-ietf-i2rs-01," 2014.
[43] IETF, "RFC6020 - YANG," 2014.
[44] C. Filsfils, and C. Martin, “"Segment Routing"- Presentation to Ripe 66. .”
[45] P. Willis, "Network Functions Virtualization."
[46] R. Bifulco, "Rethinking Access Networks with High Performance Virtual Software BRASes,."
[47] A. Shafi, J. Faiz, M. Farooq, and S. Shah, “an architectural evaluation of sdn controllers,” 2013.
[48] J. Stribling, M. Casado, N. Gude, and T. Koponen, “Onix: A Distributed Control Platform for Large-scale Production Networks,” 2010.
[49] Fujitsu, *Technical Report Carrier Software Defined Networking (SDN)*, Fujitsu Telecommunications Europe Limited, 2014.
[50] O. D. Light, "Open Day Light Service Provider Edition," 2015.
[51] R. Vilalta, A. Mayoral, R. Munoz, and R. Casellas, “integrated it and network orchestration using openstack, opendaylight and active stateful pce for intra and inter data center connectivity,” 2014.
[52] M. Tornatore, L. Gifre, B. Mukherjee, and L. Contreras, “abno-driven content distribution in the telecom cloud,” 2015.
[53] A. Aguado, V. López, J. Marhuenda, O. G. d. Dios, and J. P. Fernández-Palacios, "ABNO: a feasible SDN approach for multi-vendor IP and optical networks."
[54] A. Napoli, A. D, #39, Errico, G. Ferraris, and M. Bohn, “Elastic optical networks: The vision of the ICT project IDEALIST,” 2013.

[55] "ONOS project," 2015].
[56] D. Hood, "TR-502 ", 2014].
[57] S. Sharma, D. Staessens, and D. Colle, “Software defined networking: Meeting carrier grade requirements,” 2011.
[58] IETF, "RFC 3746 - Forwarding and Control Element Separation (ForCES) Framework," 2004.
[59] R. Martnez, R. Casellas, and R. Muoz, “PCE: What is It, How Does It Work and What are Its Limitations?,” 2014.
[60] EU-FP7. "SPARC - Split Architecture Carrier Grade Networks," http://www.fp7-sparc.eu/.
[61] F. J. R. Salguero, "Network Virtualisation -Recovering the position in the Telecom value chain."
[62] EU-FP7. "OFELIA," http://www.fp7-ofelia.eu/.
[63] T. S. R. Shen, "Experimental Demonstration of Reconfigurable Long-Reach UltraFlow Access: Software-Defined Dual-Mode Networks."
[64] W.-Q. Xu, and T.-J. Wu, “TCP issues in mobile ad hoc networks: Challenges and solutions,” *Journal of Computer Science and Technology,* vol. 21, no. 1, pp. 72-81, 2006.
[65] S. Floyd, and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Transactions on networking,* vol. 1, no. 4, pp. 397-413, 1993.
[66] S. Floyd, “TCP and explicit congestion notification,” *ACM SIGCOMM Computer Communication Review,* vol. 24, no. 5, pp. 8-23, 1994.
[67] J. Warner, "Buffer Sizes of Common Routers and Switches," https://people.ucsc.edu/~warner/buffer.html, 2012].
[68] J. F. Kurose, *Computer Networking: A Top-Down Approach Featuring the Internet, 3/E*: Pearson Education India, 2005.
[69] G. Appenzeller, I. Keslassy, and N. McKeown, *Sizing router buffers*: ACM, 2004.
[70] T. Cloonan, "Minimising Bufferbloat and optimising Packet Stream Performance in Docsis 3.0 CMs and CMTSs," 2013].
[71] K. Nichols, and V. Jacobson, “Controlling queue delay,” *Communications of the ACM,* vol. 55, no. 7, pp. 42-50, 2012.
[72] V. Jacobson, "A rant on queues. A talk presented at MIT Lincoln Labs, Lexington, MA, 2006."
[73] C. G. White, “Active Queue Management Algorithms for DOCSIS 3.0: A Simulation Study of CoDel, SFQ-CoDel and PIE in DOCSIS 3.0 Networks,” *Cable Television Laboratories*, 2013.
[74] G. White, "DOCSIS Best Practices and Guidelines for Cable Modem Buffer Control," *CM-GL-Buffer,* V01-110915, 2015].
[75] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A lightweight control scheme to address the bufferbloat problem." pp. 148-155.
[76] R. Braden, D. Clark, and S. Shenker, “RFC 1633. Integrated Services in the Internet Architecture: An Overview,” *IETF*, 1998.
[77] R. Braden, and L. Zhang, *Resource ReSerVation Protocol (RSVP)--Version 1 Message Processing Rules,* 2070-1721, 1997.
[78] J.-S. Li, and C.-S. Mao, “Providing flow-based proportional differentiated services in class-based DiffServ routers,” *IEE Proceedings-Communications,* vol. 151, no. 1, pp. 82-88, 2004.
[79] S. Blake, D. Black, M. Carlson, D. Davies, W. Wang, and W. Weiss, “RFC 2475. An Architecture for Differentiated Services,” *IETF*, 1998.
[80] E. Rosen, A. Viswanathan, and R. Callon, *Multiprotocol label switching architecture,* 2070-1721, 2000.
[81] M. Hassan, and R. Jain, *High performance TCP/IP networking : concepts, issues, and solutions*: Pearson/Prentice Hall, 2004.
[82] G. Armitage, “MPLS: the magic behind the myths [multiprotocol label switching],” *IEEE Communications Magazine,* vol. 38, no. 1, pp. 124-131, 2000.
[83] J. Joung, J. Song, and S. Lee, “Flow-based QoS management architectures for the next generation network,” *ETRI journal,* vol. 30, no. 2, pp. 238-248, 2008.
[84] K. Nichols, S. Blake, F. Baker, and D. Black, *Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers,* 2070-1721, 1998.

[85] A. Chapman, "Automatic quality of service in IP networks."
[86] I. Stoica, and H. Zhang, *Providing guaranteed services without per flow management*: ACM, 1999.
[87] R. Kuroda, M. Katsuki, A. Otaka, and N. Miki, "Providing flow-based quality-of-service control in a large-scale network." pp. 740-744.
[88] J. Domzal, R. Wojcik, and A. Jajszczyk, *Guide to Flow-Aware Networking. Quality of Service Architectures and Techniques for Traffic Management*: Springer, 2015.
[89] A. Kortebi, S. Oueslati, and J. Roberts, “Implicit service differentiation using deficit round robin,” *ITC19*, 2005.
[90] A. Kortebi, S. Oueslati, and J. W. Roberts, "Cross-protect: implicit service differentiation and admission control." pp. 56-60.
[91] I. SANDVINE, "Global internet phenomena report," Fall, 2011.
[92] D. Bowman, "From the Conference Floor: CableLabs Winter Conference.” Better Broadband Blog. Sandvine."
[93] J. Liddle, "Amazon Found Every 100ms of Latency Cost Them 1% in Sale."
[94] H. J. Chao, and X. Guo, *Quality of Service Control in High-Speed Networks*: Wiley, 2002.
[95] D. Medhi, *Network routing: algorithms, protocols, and architectures*: Morgan Kaufmann, 2010.
[96] R. Giladi, *Network Processors Architecture, Programming, and Implementation*: Morgan Kaufmann, 2008.
[97] P. Crowley, M. A. Franklin, H. Hadimioglu, and P. Z. Onufryk, *Network Processor Design*: Morgan Kaufmann, 2004.
[98] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz, “Tiny Tera: a packet switch core,” *IEEE Micro,* vol. 17, no. 1, pp. 26-33, 1997.
[99] IXIA, "Switch Performance Data Sheet," 2015].
[100] "Reducing Latency with VLANS."
[101] C. J. S. DeCusatis, A. Carranza, and C. M. DeCusatis, “Communication within Clouds: Open Standards and Proprietary Protocols for Data Center Networking,” *IEEE Communications Magazine*, 2011.
[102] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric."
[103] M. Ruffini, F. Slyne, C. Bluemm, N. Kitsuwan, and S. McGettrick, “Software Defined Networking for Next Generation Converged Metro-Access Networks.,” *Optical Fiber Technology*, 2015.
[104] R. Muñoz, R. Vilalta, R. Casellas, R. Martínez, F. Francois, M. Channegowda, A. Hammad, S. Peng, R. Nejabati, and D. Simeonidou, “Transport network orchestration for end-to-end multilayer provisioning across heterogeneous SDN/OpenFlow and GMPLS/PCE control domains,” *Journal of Lightwave Technology,* vol. 33, no. 8, pp. 1540-1548, 2015.
[105] C. Headquarters, “TL1 Command Reference for the Cisco ONS 15808 DWDM System,” 2003.
[106] D. Erickson, "The beacon openflow controller." pp. 13-18.
[107] S. McGettrick, F. Slyne, and M. Ruffini, “Experimental End-to-End Demonstration of Shared N:1 Dual Homed Protection in Long Reach PON and SDN-Controlled Core,” in OFC, 2015.
[108] "ZeroMQ."
[109] "proposed OpenFlow modification in WT-358," 2016].
[110] F. Slyne, N. Kituswan, S. McGettrick, D. B. Payne, and M. Ruffini, “Design and experimental test of 1:1 End-to-End Protection for LR-PON using an SDN multi-tier Control Plane,” in ECOC, 2014.
[111] S. McGettrick, F. Slyne, N. Kitsuwan, D. B. Payne, and M. Ruffini, “Experimental End-to-End Demonstration of Shared N: M Dual-Homed Protection in SDN-Controlled Long-Reach PON and Pan-European Core,” *Journal of Lightwave Technology,* vol. 34, no. 18, pp. 4205-4213, 2016.

[112] J. M. G. Josa, F. Slyne, V. Lopez, and M. Ruffini, “End-to-end Service Orchestration From Access to Backbone.,” in ONDM, 2016.
[113] R. Vilalta, V. López, A. Mayoral, N. Yoshikane, M. Ruffini, D. Siracusa, R. Martínez, T. Szyrkowiec, A. Autenrieth, and S. Peng, "The need for a control orchestration protocol in research projects on optical networking." pp. 340-344.
[114] A. Kpsel, H. Woesner, L. Bergesio, M. Su, and T. Rothe, “Design and implementation of the OFELIA FP7 facility: the European OpenFlow testbed,” 2013.
[115] M. Ruffini, and F. Slyne, "End-to-end testing of SDN-controlled broadband architectures through GEANT: the DISCUS experience. . ."
[116] S. McGettrick, D. B. Payne, and M. Ruffini, "Improving hardware protection switching in 10Gb/s symmetric Long Reach PONs." p. OW3G. 2.
[117] G. Talli, S. Porto, D. Carey, Brandonisio, A. Naughton, P. Ossieur, F. Slyne, S. McGettrick, C. Blum, M. Ruffini, D. Payne, R. Bonk, T. Pfeiffer, N. Parsons, and P. Townsend, “Demonstration of SDN Enabled Dynamically Reconfigurable High Capacity Optical Access for Converged Services.,” in OFC (Post-deadline paper), 2016.
[118] G. Talli, F. Slyne, S. Porto, D. Carey, N. Brandonisio, A. Naughton, P. Ossieur, S. McGettrick, C. Blumm, and M. Ruffini, “SDN Enabled Dynamically Reconfigurable High Capacity Optical Access Architecture for Converged Services,” *Journal of Lightwave Technology*, 2016.
[119] "NS3 OpenFlow Module-Issues using OpenFlow with p2p network."
[120] "Ethernet module for NS-3 with adaptation to the latest NS-3."
[121] R. S. Tomlinson, “Selecting sequence numbers,” *ACM SIGOPS Operating Systems Review,* vol. 9, no. 3, pp. 11-23, 1975.
[122] V. Jacobson, "Congestion avoidance and control." pp. 314-329.
[123] P. Karn, and C. Partridge, “Improving round-trip time estimates in reliable transport protocols,” *ACM SIGCOMM Computer Communication Review,* vol. 17, no. 5, pp. 2-7, 1987.
[124] B. Heller, N. McKeown, and B. Lantz, “A network in a laptop: rapid prototyping for software-defined networks,” 2010.
[125] “sdn/openflow-based unified control of 100 gb/s-class core/metro/access optical networks,” 2014.
[126] P. Ho, B. Wu, J. Xiao, and X. Jiang, “data center network placement and service protection in all-optical mesh networks,” 2013.
[127] J. Impagliazzo, “Computing curricula 2005,” *ACM SIGCSE Bulletin,* vol. 38, no. 3, pp. 311-311, 2006.
[128] J. Humble, and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation*: Pearson Education, 2010.
[129] IETF, "RFC repository," IETF, 2012.
[130] B. Forum, "Traffic Management Specification Version 4.1," 2010.