An outright open source approach for simple and pragmatic Internet exchange

Marc Bruyère

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue par :
Marc Bruyere

le 06 Juillet 2016

Titre :
An outright open source approach for simple and pragmatic Internet eXchange

---

École doctorale et discipline ou spécialité :

ED MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de recherche :
UPR 8001 Laboratoire d'Analyse et d'Architecture des Systèmes

Directeur/trice(s) de Thèse :
Philippe Owezarski

Jury :
Philippe OWEZARSKI, LAAS - CNRS
Olivier FESTOR, TELECOM Nancy LORIA
Olivier BONAVENTURE, Université catholique de Louvain
Thierry GAYRAUD, Universtié Paul Sabatier / LAAS - CNRS
Steve UHLIG, Queen Mary, University of London
Josh BAILEY, Google Inc
Arnold NIPPER, DEC-IX

An outright open source approach for simple and pragmatic Internet eXchange

Marc Bruyère

June 2016
Version: Final

University Toulouse 3 Paul Sabatier

Services et Architectures pour Réseaux Avancés
LAAS CNRS
Réseaux et Communications

Computer Networking

An outright open source approach for simple and pragmatic Internet eXchange

Marc Bruyère

1. Reviewer Olivier Bonaventure
Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM)
Université Catholique de Louvain

2. Reviewer Olivier Festor
Institut National de Recherche en Informatique (INRIA)
Telecom Nancy

Supervisor Philippe Owerzaski

June 2016

Marc Bruyère
An outright open source approach for simple and pragmatic Internet eXchange
Computer Networking, June 2016
Reviewers: Olivier Bonaventure and Olivier Festor
Supervisors: Philippe Owerzaski *Réseaux et Communications*
LAAS CNRS
Services et Architectures pour Réseaux Avancés
7 Avenue du Colonel Roche
31400 and Toulouse

Acknowledgement

First and foremost, my appreciation goes to my doctoral advisor, Dr. Philippe Owezarski for his support throughout my Ph.D. I thank them for this unique experience for wich I am deeply grateful.

I will like to thank the Jury for having accepted to participate in my Ph.D. defense. Beginning with Professor Olivier Bonaventure and Professor Olivier Festor who have consecrated much of their time review my dissertation and give detailed feedback on my work. To Professor Thierry Gayraud, Professor Steve Uhlig, Josh Bailey, Arnold Nipper who have participated in examining my work and providing valuable insights to the improvement of the quality of this thesis.

I will like to acknowledge all the participants in the research project ALIEN and ENDEAVOUR and The Toulouse IXP for the support they have given me through the years of my Ph.D. This support was essential to the accomplishment of this work.

I will like to thanks, Prof. Andrew W. Moore from Cambridge University, who trusted me from the beginning and opened my doors.

I am extremely grateful for the help provide by Dr. Gianni Antichi from the University of Cambridge. His assistance was outstanding.

I am thankful for all my colleagues from the Laboratory of Architecture and Analysis of Systems. Among them, I will like to mention Gilles Tredan, Olivier Brun, Pascal Berthou, Yann Labit, Slim Abdellatif and many others.

I wish to thanks my friends: Stephane Jungers, Frederic Jourdan, Gilles Reyna Sanchez and Sébastien Sauge and many others. Without your company, life would not have been the same.

Finally, I would like to dedicate this thesis to my wife Sumiko my two sons Yago, Yoan, my parents Marcel, Marricucia and Raymond, my sisters Nathalie and Mireille and my brothers Nicolas, Thomas.

Abstract

The Internet is indispensable for our life today and for our globalized financial economy. The effectiveness of the Internet as a public resource depends upon interoperability and trust. Free and open source software promotes the development of the Internet as a public resource. Physical location Internet eXchange Points (IXP) are highly valuable for the Internet as neutral exchange places where all type and size of networks can exchange traffic. The fundamental service offered by IXP is a shared layer2 switching fabric. IXPs are fastened by using close source technology for their switching fabric. Although this should be a basic functionality, today solutions never address their basic requirements properly. Today's network solutions are inflexible as proprietary closed implementation of a distributed control plane tight together with the data plane.

Software Defined Networking (SDN) is an emerging paradigm decoupling the control and data planes, on opening high performance forwarding plane with OpenFlow. The goal of this thesis is to propose an IXP pragmatic Openflow switching fabric, addressing the critical requirements and affording greater simplicity and flexibility. Few SDN solutions have been presented already but all of them propose fuzzy layer 2 and 3 separation. For a enhanced stability not all control planes functions can be decoupled from the data plane. As an other goal statement, networking testing tools are essential for qualifying networking equipment. Network hardware monitoring and testing being critical for computer networks, current solutions are both very expensive, inflexible and not open source. The experience in deploying Openflow in production networks has highlight significant limitations in the support of the protocol by hardware switches. Open source hardware solutions give total flexibility and reproducibility for testing and selecting suitable hardware equipment to go to production.

We present Umbrella, a new SDN-enabled IXP fabric architecture, that aims to promote strengthening the separation of control and data plane to increase robustness, flexibility and reliability of the IXP. Umbrella abolishes broadcasting with a pseudo wire and segment routing approach. We demonstrated for an IXP fabric that not all the control plane can be decoupled from the date plane. We show that Umbrella can scale and recycle legacy non OpenFlow core switch to reduce migration cost. Into the testing tools lacuna we launch the Open Source Network Tester (OSNT), a fully

open-source traffic generator and capture system. Additionally, our approach has proved less costly than comparable commercial systems while achieving comparable levels of precision and accuracy; all within an open-source framework extensible with new features to support new applications, while permitting validation and review of the implementation. Furthermore we present the integration of OpenFLow Operations Per Second (OFLOPS), an OpenFlow switch evaluation platform, with the OSNT platform, a hardware-accelerated traffic generation and capturing platform.

We demonstrate the real flexibility and benefit of the Umbrella architecture persuading ten Internet operators to migrate the entire Toulouse IXP. The hardware testing tools we have developed have been used to qualify the hardware which has been deployed in production. The TouIX has been running stable for a year. It is fully managed and monitored through a single web application removing all the complex legacy management systems.

Résumé

L'Internet, le réseaux des réseaux, repose, en tant que ressource publique, sur l'inter opérabilité et la confiance. Les points d'échange Internet (IXP), qui interconnectent des opérateurs réseaux de types et tailles différents, jouent à ce titre un rôle majeur en tant que lieux d'échange neutres et indépendants.

Le service fondamental offert par un IXP est une fabrique de commutation de niveau 2 partagée. Aujourd'hui, pour fournir ce service de base, les IXP sont obligés d'utiliser des technologies propriétaires qui ne répondent pas correctement à toutes leurs exigences. Cette situation est principalement dûe au fait que les plans de contrôle et de données sont intriqués sans possibilités de programmer finement le plan de commutation. Le "Software Defined Networking" (SDN), nouveau paradigme découplant les plans de contrôle et de données utilise le protocole OpenFlow qui permet de programmer le plan de commutation Ethernet haute performance. Contrairement à tous les projets de recherches qui centralisent la totalité du plan de contrôle au dessus d'OpenFlow, altérant la stabilité des échanges, nous proposons d'utiliser OpenFlow pour gérer le plan de contrôle spécifique à la fabrique de commutation. Dans cette thèse, nous proposons "Umbrella", une fabrique de commutation simple et pragmatique répondant à toutes les exigences des IXPs et en premier lieu à la garantie d'indépendance et de neutralité des échanges. Dans la première partie, nous présentons l'architecture "Umbrella" en détail avec l'ensemble

des tests et validations démontrant la claire séparation du plan de contrôle et du plan de données pour augmenter la robustesse, la flexibilité et la fiabilité des IXPs.

Dans la second partie nous verrons qu'avant toute mise en œuvre, il est primordial de tester chaque équipement afin de vérifier qu'il répond aux attentes, mais les solutions utilisées pour cela sont toutes commerciales et propriétaires, et ne répondent donc pas aux besoins techniques d'indépendance et de neutralité. Pour les besoins de tests de validations nécessaires des équipements, nous avons développé l'"Open Source Network Tester" (OSNT), un système entièrement open source "hardware" de génération et de capture de trafic. Avant la mise en œuvre d'équipement Openflow comme Umbrella, Nous avons rajouté à OSNT l"OpenFLow Operations Per Second Turbo" (OFLOPS Turbo) pour évaluer les commutateurs OpenFlow 10Gbps.

Le dernier chapitre présente le déploiement de l'architecture "Umbrella" en production sur un point d'échange régional. Les outils de test que nous avons développés ont été utilisés pour vérifier les équipements déployés en production. Ce point d'échange, stable depuis maintenant un an, est entièrement géré et contrôlé par une seule application Web remplaçant tous les systèmes complexes et propriétaires de gestion utilisés précédemment.

Contents

1 Introduction

> *On-line man-computer communication requires much development before men and computers can work together effectively in formulative thinking and intuitive problem solving.*
>
> **— J.C.R. Licklider and Welden E.Clark**
> (MIT - BBN (1962))

Nowadays, the Internet has revolutionized communications and is becoming the preferred medium of communication. The Internet is crucial for our everyday life as well as for the globalized financial economy. The global Internet traffic is growing exponentially[@24] [@48].

To face topological traffic engineering and network service issues of the Internet, Internet eXchange Points (IXP) have been designed and are replacing the global transit model of the Internet. IXPs are fabrics where Internet Service Providers, carriers, content providers and other Internet companies come together to exchange traffic. IXPs are essential for the Internet evolution as they empower high bandwidth low latency and inexpensive local traffic peering as opposed to transit traffic. Today there are already hundreds of IXPs around the world, mainly located in big cities in developed countries. Small and medium-sized cities often still rely on commercial peerings for their local traffic to reach IXPs, even if their physical locations are very close. Today, a large European IXP has to handle the 2013's global Internet traffic[@7]. In a few years, more than half of the global Internet traffic will be exchanged locally in metropolitan areas instead of being transported through long-haul links. Internet eXchange Points (IXPs) are the essence of the Internet.

IXPs are neutral metropolitan facilities where all types and sizes of networks can exchange traffic through "peering" interconnections. The fundamental service offered by IXPs is a shared layer 2 switching fabric that is hard to scale and to manage. Today's solutions fail to address some of the basic requirements [43] . They are inflexible, proprietary, with closed implementations. The networking vendors are struggling with building equipment with their control and data plane coupled, causing most of the limitations. IXPs operators maintain their switching fabric devices, configuring each of it manually. This process is time-consuming, inaccurate and expensive. IXPs operator teams are usually composed of highly skilled engineers

to be able to deploy and support operations. This is in contrast with the nature of IXPs which are usually no profit organizations. Part of the issue comes from the location discovery traffic (broadcast). Broadcast traffic is crucial for the switching fabric to run. It needs to be handled in very specific manner because it can consume critical resources from all the connected operators device. In some circumstances, it can even cause the complete IXP operations to stop. Some solutions and network architecture exist, but they are not perfect [17] [@63] [@6]. Distributed control plane or centralized approach does focus on reliability robustness. The full potential of programming from scratch to a specific function with SDN (a data plane) optimizes the solution. Introducing Software Defined Networking (SDN) into IXPs is a promising solution to long-standing fabric management issues and it has also been proved to enhance inter-domain routing [32] [38] [**iSDX**].

The rest of this thesis is organized as follows: chapter 2 introduces the specific context from the origins of the Internet to the IXPs ecosystems, types of organizations and technical environment. The last section of the chapter introduces why SDN exists and its benefits. In chapter 3, we present Umbrella, a reliable SDN interconnection fabric that complements and enhances previous SDN efforts. Umbrella revisits the separation of control and data plane within the interconnection fabric, to increase both robustness and reliability of the data path. We propose a new SDN fabric which is scalable, reliable, and with enhanced support of advanced SDN applications with strong control plane expectations. The need for tools capable of evaluating the performances of new SDN solutions lead to the definition and implementation of new systems for network testing and measurements. Indeed, proprietary solutions resulted in being too expensive and not flexible enough, while software based open source solution not scalable enough. In this scenario, chapter 4 presents the architecture of the Open Source Network Tester (OSNT) and OpenFLow Operations Per Second Turbo. Both systems, implemented on reconfigurable hardware (NetFPGA-10G), proved to be scalable and flexible.

Finally, chapter 5 reports the work that has been done to migrate the TouIX IXP located in Toulouse, France from a traditional to a full OpenFlow IXP using the Umbrella architecture. We have developed an SDN Web application TouSIX-Manager to provide a single point of management for all the members. All members can configure the fabric directly in a very basic way and monitor the traffic exchanged in detail.

The initial part of this work was conducted within the framework of the FP7 ALIEN [1] granted by the EU (Grant Agr. No. 317880). The ALIEN project, finished in Oct 2014, delivered an innovative network abstraction mechanism targeting the control and

[1] `http://www.fp7-alien.eu/`

management convergence and interoperability of heterogeneous network elements, building strong foundations for Software Defined Networks.

The second half of the thesis was supported by the ENDEAVOUR H2020 project [2] (Grant Agr. No. 644960). The objective of ENDEAVOUR is to address current limitations of the Internet interconnection model, as well as creating opportunities for novel services, bringing the possibility for new economic models around the ecosystems already established.

[2] http://h2020-endeavour.eu/

Context

2

> *Celui qui trouve sans chercher est celui qui a longtemps cherché sans trouver.*
>
> — **Gaston Bachelard**
> (Philosopher)

2.1 The Internet

2.1.1 Definition of Internet

The Internet is a global network composed of many voluntarily interconnected autonomous networks. This 1995 definition of the Internet was proposed by the US Federal Networking Council (FNC) [@33] :

"Internet" refers to the global information system that :

1. is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons;

2. is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible protocols; and

3. provides, uses or makes accessible, either publicly or privately, high level services layered on the communications and related infrastructure described herein."

2.1.2 Brief history of the Internet

The Internet revolution can be compared to the printing revolution with Johannes Gutenberg or closer to us the invention of the telegraph, the radio and the television. But with computers, the Internet represents one of the most successful examples of benefits for our society.

Origins of the internet The first person to publicly mentions social interaction that could be derived from networking was in a series of memos by J.C.R Licklider of MIT in August 1962 [@56] . He presented his views of a "Galactic Network" concept. What he described was a global network of computers, where users can access data and programs from everywhere.

The first paper about packet switching theory was published in 1961 by Leonard Kleinrock from MIT [53]. In the middles of the 60's different experiments were conducted at MIT, proving the feasibility of the packet switching concept. At the end of the 60's Defense Advanced Research Projects Agency (DARPA) put together the first plan for the "Advanced Research Projects Agency Network" ARPANET[89]. The DARPA is a U.S. Department of Defense agency responsible for the development of emerging technologies for military usage. ARPANET started to interconnect academic institutions using the first packet switch called Interface Message Processors (IMP) from Bolt Beranek and Newman (BBN) an MIT company. Computers were added quickly to the ARPANET during the following years, and work proceeded to fullfil a functionally complete Host-to-Host protocol and other network software. This Host-to-Host protocol, was called the Network Control Protocol (NCP). In 1972 Robert Khan (a DARPA Engineer) organized a large, very successful demonstration of the ARPANET at the International Computer Communication Conference (ICCC).

The "Internetting" foundation concepts The original ARPANET grew to become the Internet. Internet was based on the idea that there would be multiple independent networks of satellite networks, ground-based packet radio networks and other network interconnected through packet switching. The approach, is know as open-architecture networking. The Open-architecture networking is the keystone of the Internet foundation. It comes from the Robert Khan approach [103] during the early age of ARPANET, which would allow computers and networks all over the world to communicate with each other, regardless of what hardware or software the computers on each network used. It operates without a central governing body.

Four ground rules were critical to Kahn's early thinking:

- Each distinct network would have to stand on its own and no internal changes could be required for any such network to connect it to the Internet.

- Communications would be performed on a best effort basis. If a packet didn't make it to the final destination, it would shortly be retransmitted from the source.

- Black boxes would be used to connect the networks. These would later be called gateways and routers. There would be no information retained by the

gateways about the individual flows of packets passing through them, thereby keeping them simple.

- There would be no global control at the operations level.

Around 1973, Vinton Cerf and Robert Khan started to work together on the Transmission Control Protocol for the Internet Protocol. Based on this initial work, a few implementations of the TCP/IP protocols suite were developed [22]. This was the beginning of a long period of experimentation and development to evolve and mature the Internet concepts and technology. In 1983, NCP was replaced by TCP/IP for ARPANET [66].

During this period, the size of the Internet increased, and also challenged the capabilities of the routers. Originally, there was a single distributed routing algorithm for routing that was implemented by all routers in the Internet. As the number of networks in the Internet exploded, this initial design could not expand as necessary, so it was replaced by a hierarchical model of routing, with an Interior Gateway Protocol (IGP) used inside each region of the Internet, and an Exterior Gateway Protocol (EGP) used to tie the regions together. This design permitted different regions to use a different IGP, so that different requirements for cost, rapid reconfiguration, robustness, and scale could be accommodated.

NSF, The US National Science Foundation started in 1985 the NSFNET program. TCP/IP was mandatory to interconnect with NSFNET. In 1986 the Internet Engineering Task Force (IETF) was created to coordinate standardization effort. One of the first Request For Comments (RFC) out from the recent IETF is the "Requirements for Internet Gateways". The RFC985 defines the first Gateway to interconnect the ARPANET and NSFNET.

Various US Federal agencies started to interconnect together and to managed "interconnection points" for interagency traffic. The Federal Internet Exchanges (FIX-E and FIX-W) were built for this purpose. They served as models for the Network Access Points and "Internet eXchange Point" (IXP) facilities, the glue that binds of today's Internet architecture.

The pressing question for the future of the Internet is not how technology will become, but how the process of change and the evolution itself will be managed. The architecture of the Internet has always been handled by a core group of architects, but the form of that group has changed as the number of interested parties has grown [76]. Researchers are now pushing for a paradigm change in networking.

What we can retain from the History of the Internet The openness principle at the origin of the Internet is a practical demonstration of what works in inter-

networking. The Internet and its "gateways" have evolved and in the next section we will see how. In the next sections we describe how the Internet traffic and number of autonomous systems grow.

2.2 How is Internet traffic evolving ?

To date, 3,34 billion people are using the Internet. This represents a bit more than 40 percent of the human population. The graph from Figure 2.1 [@48] represents the distribution of global Information and Communications Technology (ICT) per type of access.

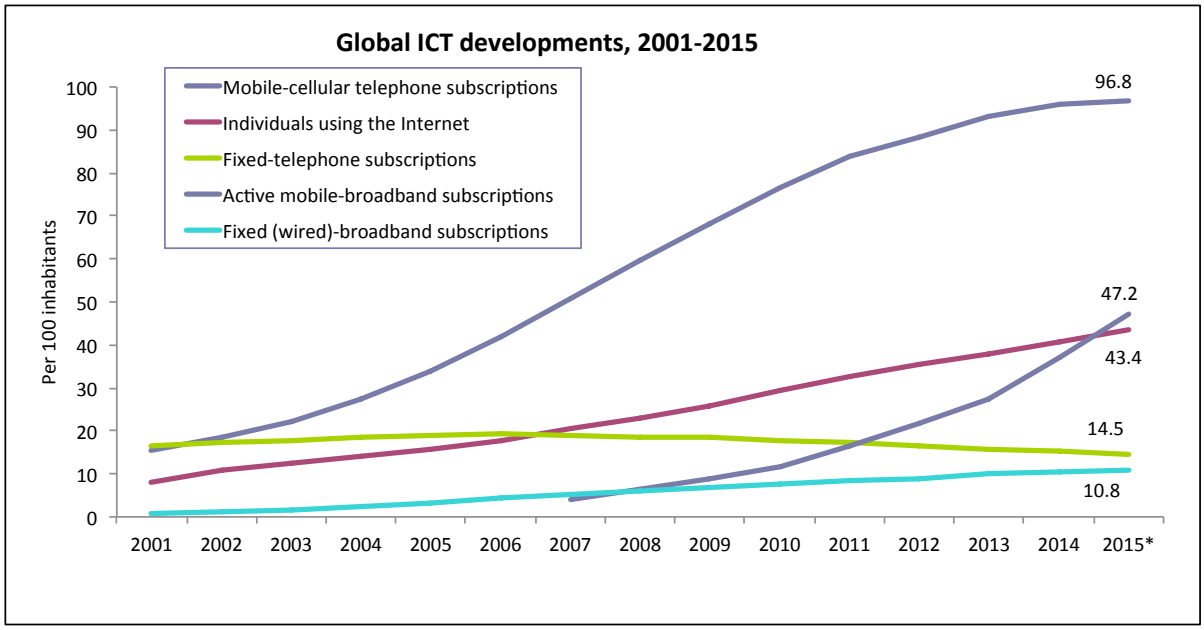

Fig. 2.1: Percentage of worldwide population accessing ICT from 2001 to 2015 - source ITU

There are two major rising trends : the active mobile-broadband (from the GSM Association [1] 4G-LTE and the future 5G) and the Internet of Things IoT (IoT refers to physical objects : vehicles, embedded electronics, and sensors more able to communicate through Internet). By 2020 experts estimate that the IoT will consist of almost 50 billion objects[@25].

Internet traffic forecast

The Internet traffic forecast from Cisco's Visual Networking Index [@46] predicts how and where the Internet traffic will grow:

[1] http://www.gsma.com/

Global IP traffic has increased more than fivefold in the past 5 years, and will increase nearly threefold over the next 5 years. Overall, IP traffic will grow at a compound annual growth rate (CAGR) of 23 percent from 2014 to 2019.

Metro traffic will surpass long-haul traffic in 2015, and will account for 66 percent of the total IP traffic by 2019. Globally, metro traffic will grow nearly twice as fast as long-haul traffic from 2014 to 2019. The higher growth in metro networks is due in part to the increasingly significant role of content delivery networks, which bypass long- haul links and deliver traffic to metro and regional backbones.

Traffic from wireless and mobile devices will exceed traffic from wired devices by 2019. By 2019, wired devices will account for 33 percent of IP traffic, while Wi-Fi and mobile devices will account for 66 percent of IP traffic. In 2014, wired devices accounted for the majority of IP traffic with 54 percent.

Regional Highlights

From Cisco's Visual Networking Index [@46] a highlight per region:

IP traffic is growing fastest in the Middle East and Africa, followed by Asia Pacific. Traffic in the Middle East and Africa will grow at a CAGR of 44 percent between 2014 and 2019.

IP traffic in North America will reach 49.7 exabytes per month by 2019, at a CAGR of 20 percent. Monthly Internet traffic in North America will generate 9 billion DVDs' worth of traffic, or 35.4 exabytes per month.

IP traffic in Western Europe will reach 24.7 exabytes per month by 2019, at a CAGR of 21 percent. Monthly Internet traffic in Western Europe will generate 5 billion DVDs' worth of traffic, or 20.8 exabytes per month.

IP traffic in Asia Pacific will reach 54.4 exabytes per month by 2019, at a CAGR of 21 percent. Monthly Internet traffic in Asia Pacific will generate 11 billion DVDs' worth of traffic, or 44.1 exabytes per month.

IP traffic in Latin America will reach 12.9 exabytes per month by 2019, at a CAGR of 25 percent. Monthly Internet traffic in Latin America will generate 3 billion DVDs' worth of traffic, or 11.3 exabytes per month.

IP traffic in Central and Eastern Europe will reach 16.9 exabytes per month by 2019, at a CAGR of 33 percent. Monthly Internet traffic in Central and Eastern Europe will generate 4 billion DVDs' worth of traffic, or 15.8 exabytes per month.

IP traffic in the Middle East and Africa will reach 9.4 exabytes per month by 2019, at a CAGR of 44 percent. Monthly Internet traffic in the Middle East and Africa will generate 2 billion DVDs' worth of traffic, or 8.8 exabytes per month.

Internet growth is exponential. Not only because of the evolution of Internet usage, but also because a significant number of the global population is still left unconnected. The metro traffic is growing more rapidly than the long distance one, as the network content is moving closer to the users.

How is the number of Autonomous Systems growing ?

The Internet expansion is also supported by the creation of new autonomous systems. **Definition (AS)**: An Autonomous Systems (AS) is the unit of administrative policy, either a single network or a group of networks that is controlled by a common network administrator (or group of administrators) on behalf of a single administrative entity (such as a university, a business enterprise, or a business division). As seen on Figure 2.2 1996 to present, we can observe a regular rate of new entering autonomous systems with a bit more than 2000 per year [@24].

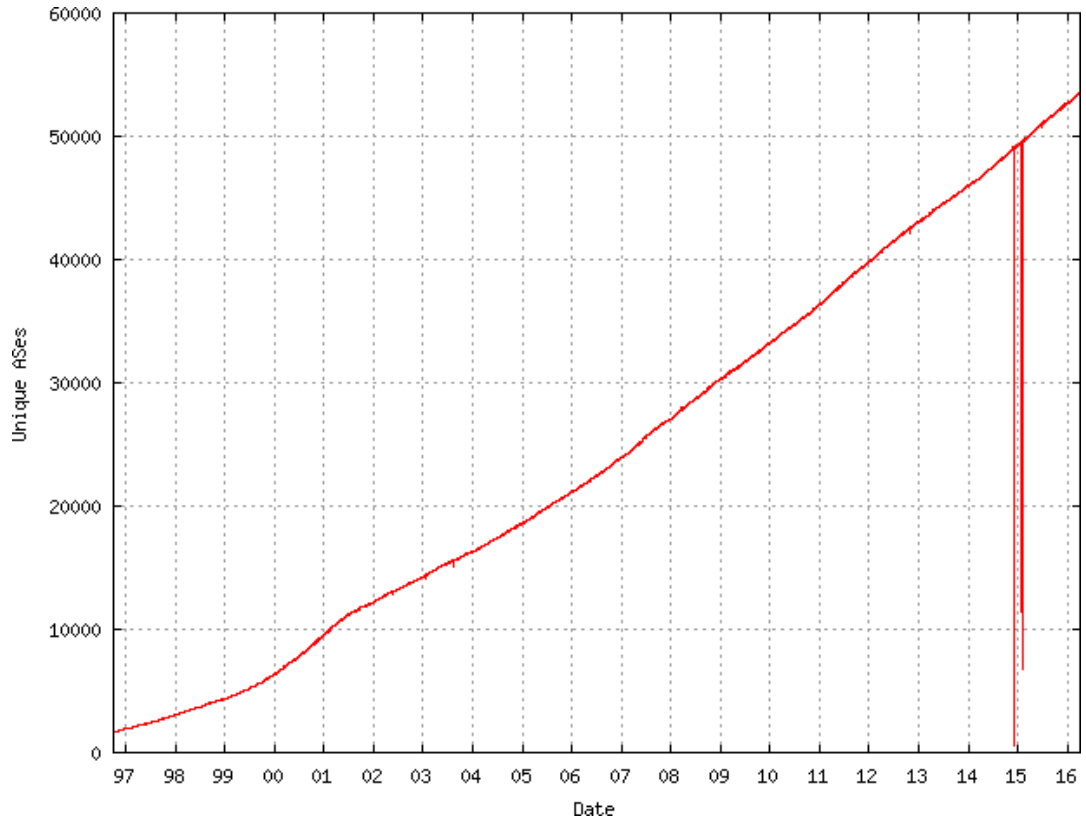

Fig. 2.2: Number of unique Internet Autonomous System - source CIDR-REPORT

How is growing traffic at large IXPs

The traffic from the largest european IXP, as see in Figure 2.3 [@7] increases exponentially . From the Cisco VNI archive, the total monthly accumulated Internet traffic for early 2013 was less than the traffic exchanged at the beginning of 2016 at AMS-IX. Almost 800 autonomous systems connected mid 2016 [2].

Not all IXPs provided statistics on their accumulated total monthly traffic exchanged. As seen in Figure 2.4 we can observe the terabits per second over five minute period. Large european IXP carry more than 5 terabits per second of peak traffic daily.

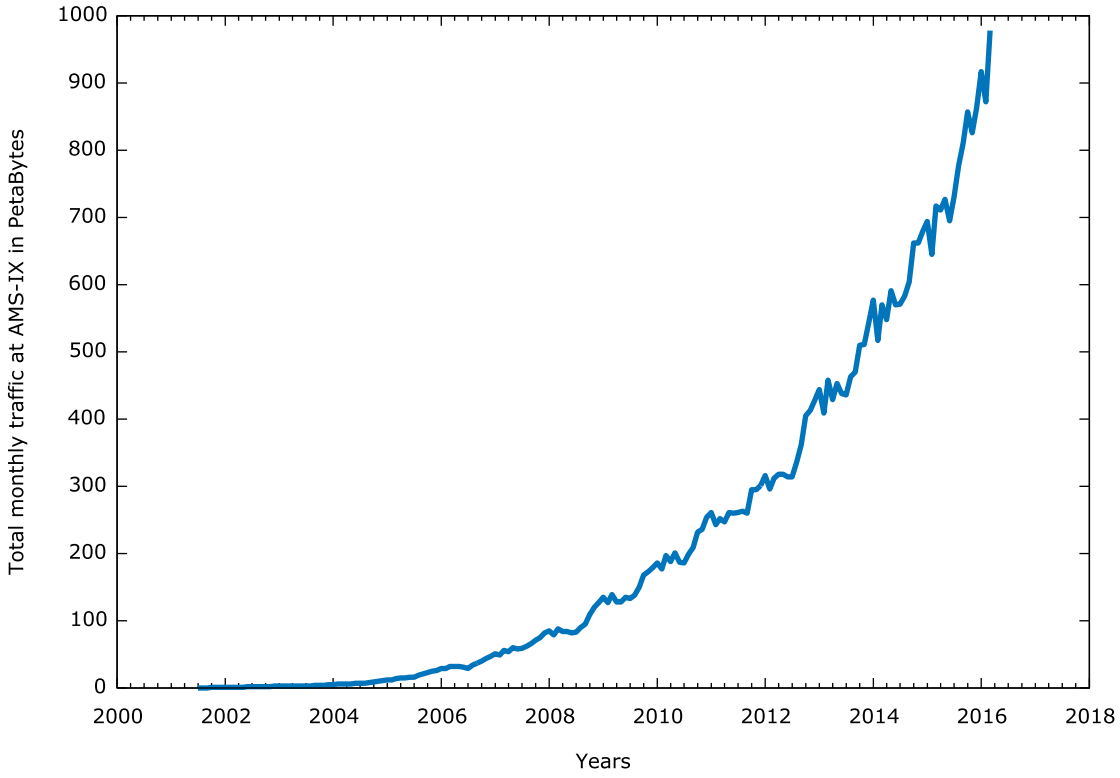

Fig. 2.3: Total monthly traffic in PetaBytes at AMS-IX

2.2.1 subsection conclusion

More and more people use Internet transported through the Internet. New network operators are appearing at a constant rate. There is still more than half of the worldwide population waiting to be connected to the Internet. This growth needs to be settled in metro areas, where a major part of the traffic is concentrated locally. In the next sections, we will look at IXPs which ease interconnecting those diverse networks and contents.

[2] `https://ams-ix.net/`

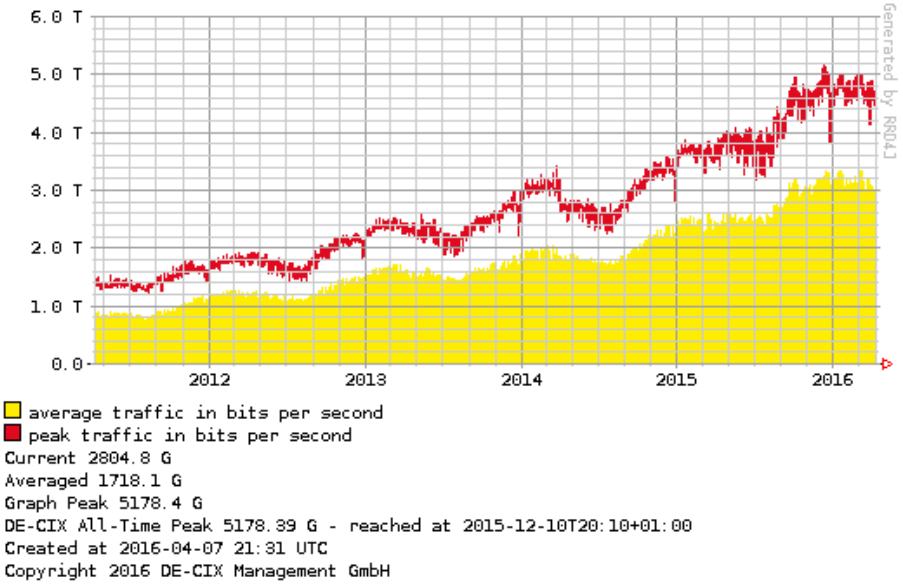

Fig. 2.4: Total traffic in TeraBits per second at DE-CIX Frankfurt

2.3 Internet eXchange Points introduction

The definition of an Internet Exchange Point

"Autonomous Systems" has the meaning given in BCP6/RFC1930[3] , "Guidelines for creation, selection, and registration of an Autonomous System (AS)".

The Euro-IX[4] definition of an Internet Exchange Point :

An Internet Exchange Point (IXP) is a network facility that enables the interconnection and exchange of Internet traffic between more than two independent Autonomous Systems.

An IXP provides interconnection only for *Autonomous Systems*.

An IXP does not require the Internet traffic passing between any pair of participating *Autonomous Systems* to pass through any third *Autonomous Systems*, nor does it alter or otherwise interfere with such traffic.

"Independent" means Autonomous Systems that are operated by organizational entities with separate legal personality.

The Internet is a large network of networks, a global communication network composed of thousands of individual networks interconnected in a densely populated

[3] https://tools.ietf.org/html/rfc1930
[4] http://euro-ix.net/

mesh. Each of the networks is, in some form, a portion of the Internet. Different kinds of networks exist; for instance, consider networks with various users, services or resources. To effectively be part of the Internet, each network needs to be able to send and receive traffic to and from any other network. Different networks can interoperate if and only if they all speak the same language: IP.

Internet exchange points are physical locations where networks interconnect and exchange traffic with each other. The practice of exchanging traffic between and among networks at an IXP is called peering. Internet service providers (ISPs) peer at IXPs, where they exchange traffic that originate from other networks or from other customer's networks. Peering is largely based on voluntary agreements by both networks as a result of acknowledging the value of being directly connected: IP packets are routed directly using the shortest and cheapest path between both networks. By exchanging traffic at an IXP, ISPs do not have to build out their networks to all their "peers," which cuts costs, frees up money, labor, resources, and allows for a more competitive market environment.

The Internet is large in scale and geographically spread over countries and continents. As a consequence, a majority of the networks cannot interconnect directly. Thus, most networks must use a third-party network to route packets to and from the rest of the Internet. This commercial service is known as "transit" and typically involves a payment based on a contractual obligation as opposed to settlement-free peering. This point is especially significant for networks in countries without IXPs that are more likely to route inter-network traffic via expensive transit facilities.

Lastly, IXPs and the dense interconnection network have increased the resilience of the European Internet. This density has helped the European Internet work through major events such as 9/11 and large scale outages in commercial networks, which has allowed the network and content delivery to continue to function. It is an aspect often overlooked but IXPs and interconnections play a crucial role in securing the Internet and all the applications that depend on it.

2.3.1 What are the benefits of an IXP?

Reduction of a network's operational cost. Direct exchanged traffic between networks reduces cost to the single price of the shared infrastructure. For an ISP to lower its transit access cost to access the Internet, such direct exchanged traffic is advantageous.

Keeping traffic local and reducing latency. IXPs ease direct network interconnection. The direct links created between them allow the traffic to remain local and

forwarded with the lowest possible latency. Latency is the time elapsed to transport IP packets from the source to the destination.

More autonomy and better control of network's own resources. Interconnecting through IXPs gives networks more control over their resources and autonomy, including traffic management and routing, as it decreases the network's dependency on third-party providers. ISPs that just use transit provider and do not peer directly or connect to an IXP, are fully dependent on the transit provider. Additionally the social community created around the IXPs bring certain human benefits.

More robustness and stability for the local Internet. Increasing the number of direct connection between networks increases the robustness and stability of the Internet in the case of network interruptions, denial of service (DoS) attacks. Some of the largest IXPs serve as a meeting point to more than 800 networks. Understanding the impact that IXPs have on both the regional and local levels is self-evident when the growth and connectivity of networks are divided in over time.

Enable competition by supporting new and smaller provider entrants. IXPs can facilitate competition by helping the entrance of new service providers reduce access costs. For example, new entrants do not have to build out their own networks to reach the other networks they are exchanging present at an IXP. Moreover, an IXP commonly provides a neutral traffic exchange point while bilateral interconnection with officials and/or larger networks can both include other barriers from access to cost. Developing competition is frequently a fundamental policy objective of liberalized telecommunication markets and policymakers are often brought by the self-regulatory secondary effect of IXPs. An IXP can attract content and other service providers. For example, some of the large content delivery networks were not interested in peering to all french ISPs but helped and pushed to create a neutral internet in Paris. The IXP made it easier to bring content closer to the market [70].

IXP operator. The IXP operator is responsible for the operation, management, maintenance and budgeting of the IXP's infrastructure. The IXP operator is not in charge of the operation and maintenance of third-party equipment located at the IXP (e.g., member routers, servers). This includes the cabling and switching equipment that composes the core of the IXP, but can also incorporate other services such as : servers, route servers, route collectors, reverse domain name system (DNS), time-servers and the IXPs' website, AS112 service. Responsibilities can also include working with the building operator to provide racks, power, cooling, security, and other infrastructure requirements of the IXP. The IXP operator is generally responsible for promoting the IXP and encouraging members, approved by the other partners in the IXP community. Furthermore, the IXP operator also works alongside participating members network operators at the IXP to make sure

that operations run easily, equipment is connected and configured correctly, and to manage technical support inquiries. The operator provides the necessary information and technical documentation upon request. The operator will consult and involve the IXPs' members to cooperate and grant policy and governance for the IXP, with other concerns and topics, through member meetings. While engagement and consultation are essential for any successful IXP, the way in which members are associated with the policy depends mainly on the chosen business models and governance as well as how the IXP is founded from the beginning.

IXP members/participants. A network operator can manage any network such as a government network, university or National Research, ISP network, private enterprise network (e.g., that of a bank), or a content distribution network (CDN) and Education Network (NREN). Many IXPs receive their funding from network operators peering at the IXP. The IXPs' funding model and the role the members play in the IXP, will depend on a large degree of its business models and governance.

To connect to the IXP, a network operator must establish its network to the IXP from its nearest Point of Presence (PoP), including connectivity to the IXP (e.g., dark fiber, microwave link or leased line), and place equipment at the IXP. The network operator must also arrange peering agreements with other network operators and configure its router(s) accordingly. It must then provide ongoing maintenance and management. The network operator should also monitor its traffic.

Building and facilities operator. The IXP neutrality of the IXP location is one of the most crucial parts. All the existing PoPs in a given metropolitan area require the presence of the IXP for every potential participant. Physical access to the IXP fabric is the key point of neutrality (in addition to neutral management of the IXP). In some circumstances, the IXP can rely on the facility operator for other infrastructures such as racks and cabling. The IXP operator needs to work with the facility operator to plan for any extension or negotiate continuous fees for use. Additionally, an IXP has to be located in a building with the adequate facilities to support it, which include meeting its power, cooling, space and security needs.

The role of IXP associations (IXPAs). Internet exchange point associations (IXPAs) have a significant role in developing best practice and knowledge exchange inside the IXP community. They support their members in many ways like promoting IXP to benefit the rest of the Internet community and to help them face their challenges. The IXPAs are knowledge centers. The IXPAs recognized a need to combine their resources to coordinate technical standards, develop common procedures, and share and publish statistics. The IXPAs are:

- For the African region : African AFIX [5]
- For the Asia Pacific region: APIX [6]
- For the European region: Euro-IX [7]
- For the Latin American and Caribbean region : [8]

The four IXPAs listed above have formed the Internet Exchange Point Federation (IX-F) [9] to build a global IXP community and help the development of IXPs throughout the world.

IXPAs provide support to IXPs by holding regular meetings. They maintain the IX-F database (which is the only database maintained on IXPs). They operate mailing lists where IXPs provide support to each other on a variety of topics technical, commercial, and regulatory topics. They provide documentation for best practices and tools that can be used by the networking community as well as IXPs.

2.3.2 IXPs in the world

IXPs operate in many countries around the globe. Figures 2.5 and 2.6 show the localisation of IXPs in Europe and around the world per IXPAs affiliation. There are 518 IXPs Worldwide in 120 Countries and 348 Cities. For the European region there are 217 IXPs in 49 Countries and 154 Cities [@31] [@78].

Emerging countries, small city operators and content delivery networks tend to find agreements for opening new IXP facilities. The Internet infrastructure extends and naturally interconnects locally pushed by the IXPs benefits. An IXPs can be started with only a few resources [@86]. The hard part is getting the local network operators community to acknowledge the mutual benefit of having an IXP to peer between each others and exchange traffic.

2.3.3 IXP technical components

IXPs are not usually, engaged in the peering agreements between connected ISPs. Whom an ISP peers with, and the conditions of that peering, are a matter for the two ISPs involved. IXPs do however have conditions that an ISP must respect to connect to the IXP. Also, since the physical network infrastructure is shared by all

[5] http://www.af-ix.net
[6] http://www.apix.asia
[7] http://www.euro-ix.net
[8] http://www.lac-ix.net
[9] http://www.ix-f.net/

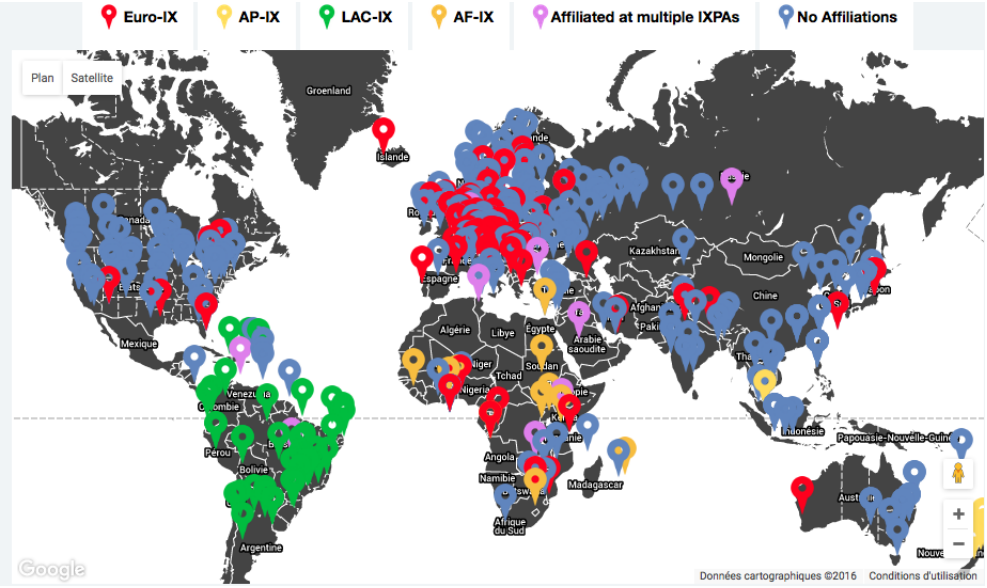

Fig. 2.5: EURO-IX map of the world IXPs per association affiliation - source EURO-IX

the connected ISPs, and activities of one ISP can affect the other connected ISPs, all IXPs have rules that establish the correct usage of the IXP.

Ethernet IEEE MAC Bridges standard which includes Bridging, Spanning Tree and others. It is standardized by the IEEE 802.1 working group.

Today's IXPs are all using Ethernet bridging with the MAC learning algorithm maintaining at least a single broadcast domain to enable the IXP members's routers to connect with each other. By MAC learning algorithm we mean to say the family of Ethernet IEEE MAC Bridges standard which includes Bridging, Spanning Tree and other. They are standardized by the IEEE 802.1 working group [@44]. IXPs Ethernet switching fabric are based on Transparent bridges and are so called because they appear as transparent to network hosts. When a transparent Bridge is powered on, it inform itself of the host locations by examining the source MAC address of incoming frames. For example, if a bridge views a frame arrive on port 1 from Host A, the bridge assumes that Host A can be reached through the connection to port 1. Through this process, transparent bridges build a table (the learning process). The Bridge uses its table for forwarding . When an Ethernet frame is received on one of the bridge's port, the bridge looks up the frame's destination MAC addresses in its internal table if there is a match. If the table contains the MAC address between the destination MAC address and any of the bridge's ports aside from the one on which the frame was received, the frame is forwarded out the designated port. If there is no match, the frame is flooded to all ports except the inbound port. Broadcasts and multicasts also are flooded in the same way.

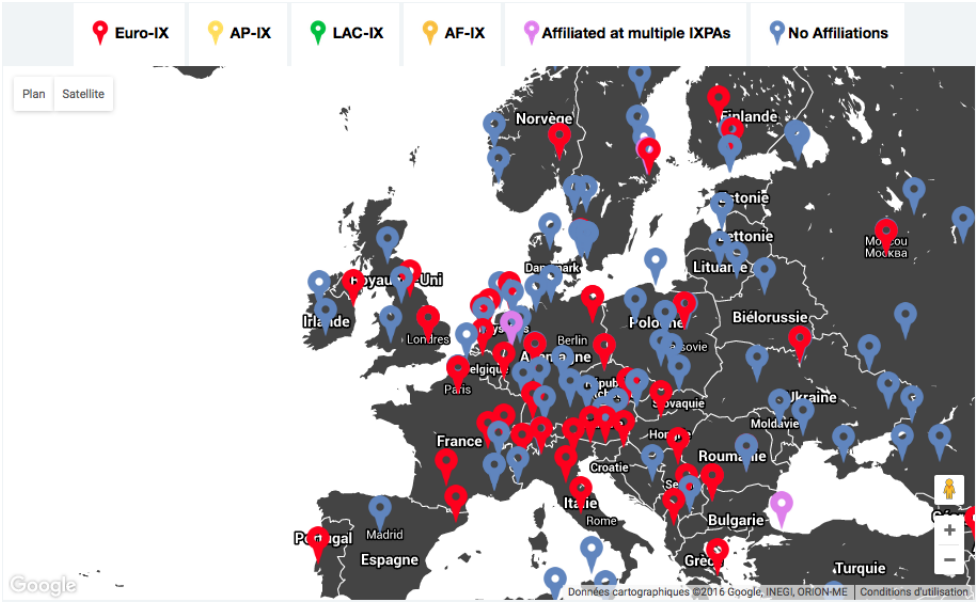

Fig. 2.6: EURO-IX map of the European IXPs per association affiliation - source EURO-IX

Not only can connectivity problems happen, but the reproduction of broadcast messages in networks with loops is a serious network issue. The frames are forwarded over and over again, using all available network bandwidth and block the entire broadcast domain. If the original frame is an ARP broadcast packet, all the connected hosts or routers will have to inspect all of them, which induces a very high usage of their CPUs.

Anyway loops can be seen as multiple paths and a network with multiple paths, from source to destination can improve overall network fault tolerance. Without a bridge-to-bridge protocol, the transparent-bridge algorithm breaks if a bridging loop is created.

On top of this interconnecting bridging domain, IXPs provide additional services, and adopt different technologies and/or architectures for scaling and securing their production environment. Below we will review possible technical solutions.

2.3.4 Interfaces Characteristics

IXPs have clear public rules for connecting to their infrastructure; this section reviews the fundamentals.

The customer interface. A clear demarcation interface between the IXP services and the members is required. It can be done either directly on the exchange or via a common demarcation point. This rule of separation is essential to determine the responsibility limits.

Ethernet physical interface. IXPs offer IEEE 802.3 Ethernet connectivity on a common switch infrastructure. Service offerings need to be available at least at the following IEEE defined rates (most seen rate first):

- 802.3z 1GE,
- 802.3ae 10GE,
- 802.3ba 40G/100G.

Media type could vary from copper to multi- or mono- mode fiber.

Traffic allowed to be forwarded. Only specific frames are switched by the fabric. The IXP fabric forwards frames with the following Ethernet types:

- 0x0800 – IPv4,
- 0x86dd – IPv6,
- 0x0806 – ARP.

MAC filtering. For security reason and to limit any other MAC to send unauthorized traffic, IXPs apply MAC address locking mechanism at the member interface port. Only the authorized and well-known participants routers MAC address can be forwarded by the switching fabric. **Public VLANs.** The IEEE 802.1q is a standard supporting Virtual LANs (VLAN), using Ethernet frame tagging techniques, permitting to have separated Layer 2 bridging domain on the same physical infrastructure (*e.g.*, IXPs use VLANs to separate between IPv4 and IPv6 traffic).

Private VLAN. Private traffic can be exchanged using a dedicated VLAN for two or more members who want to privately interconnect. The private VLANs use the same IEEE 802.1q standard but public traffic forwarded by the IXP switches needs to have precedence over all private traffic. IXP members should dedicate and have separate physical interfaces for their private traffic.

2.3.5 Interconnecting Links

IXPs are located in large and economically developed cities, where ISPs and others operators (*e.g.*, data centers, content providers) have infrastructures. In metropolitan area fibers are expensive and reduce their cost and multiplexing optical equipment are often use.

Multiple Points of Presence architecture. IXPs are rarely present at a single location. The PoPs are interconnected through various redundant paths. Different architectures and distributed control plane protocols are used by IXPs to interconnect between PoPs.

Layer 0 – Optical network. The IXP's PoPs are interconnected with optical fibers, which are subjected to stringent operational requirements such as optical path redundancy, optical aggregation with wavelength multiplexing. These requirements have pushed IXPs to use complex optical equipment. Multiplexing and optical path failover techniques are the primary feature used here.

2.3.6 Layer 2 – Resiliency of Connectivity

The IXP switching platform needs a backplane capacity sufficiently large to handle the aggregate traffic of all customers facing ports, without oversubscription. If individual switching elements contain multiple switch fabric modules, the same conditions apply during single component failures.

To maintain connectivity within the IXP fabric, IXPs typically use distributed Layer 2 protocols. Below we will review common technologies.

Spanning Tree. Spanning Tree is an example of old technology, but still the only cross-platform dynamic solution available to operators of IXPs for dynamically managing multiple redundant links in their architecture. The IEEE 802.1w Rapid Spanning Tree Protocol (RSTP) [@45] provides fast convergence in case of link failure. Member interfaces need to be configured as end-stations that are permitted to send frames without any convergence delay. RSTP has various drawbacks: for example, there is no load sharing between links, backup links are not used to forward traffic, and in some conditions (if a bridge still run Spanning Tree Protocol 802.1d in the middle of the domain), the convergence time is rather long.

Operational requirements have driven IXPs to look into new overlay architectures allowing them to resolve these scaling issues. The remainder of this section presents various solutions, several of which are already in use today.

Virtual Private LAN Services (VPLS). The current common state-of-the-art for providing a loop-free topology is VPLS, as defined in RFC4761 (VPLS using BGP signaling) and RFC4762 (VPLS using LDP signaling) [55]. VPLS works by creating an Ethernet broadcast domain on top of a mesh of Label Switched Paths (LSPs) in an MPLS (MultiProtocol Label Switching) network. In addition to providing a loop-free topology, VPLS also brings the possibility of balancing traffic over multiple distinct paths in the network, so that redundant links are always used simultaneously.

Transparent Interconnect of Lots of Links (TRILL). TRILL [81] is another approach to optimize traffic flows in switched Layer 2 environments. Much like a VPLS-based topology, TRILL provides an optimal forwarding path through the network for unicast traffic in an "all links active" topology. One of the advantages of TRILL is that it does not require overlaying the Layer 2 service onto an IP substrate.

Virtual Extensible LAN (VXLAN). VXLAN [62] is a technique to tag frames and transport them with UDP. VXLAN discovers and authenticates tunnel endpoints dynamically end to end. The Border Gateway Protocol (BGP) control plane is used to learn and distribute both MAC and IP addresses to avoid the need for flood-and-learn mechanisms. VXLAN uses multicast or unicast to minimize flooding and mitigate ARP flooding.

Ethernet VPN (EVPN). EVPN is an Ethernet Layer 2 VPN (Virtual Private Network) [94] solution that uses BGP as control plane for MAC address signaling and learning over the network as well as for accessing topology and VPN endpoint discovery.

VPLS, VXLAN and EVPN are all running on top of a Layer 3 transport network. The transport network can be constructed using a traditional IGP routing protocol such as OSPF or IS-IS. These solutions come with per packets overheads as they use tagging techniques to extended the network namespace.

2.3.7 Overview of the current solution complexity

Together with the OSI model abstraction layers Figure 2.7 [87] represents, how the packets are forwarded through MPLS/VPLS based IXPs fabric. The VPLS and EVPN that is actually the unique choice for the largest IXPs to scale are complex to install and maintain. They require an IP routed network using ISIS or OSPF and MPLS Label transport protocols. For VPLS to run, a large number of protocols need to be configured. All the VPLS active nodes are functioning on top of a state machine that is set up with a configuration file. IXPs need to maintain complex configuration files that are specific per vendor and per node networks application software versions. All maintenance change can cause the fabric to stop working if no precaution is taken.

2.3.8 Layer 3 Domain

The typical way to establish connectivity between two IXP members is to establish a direct BGP session between two of their respective border routers. Initially, if two IXP members wanted to exchange traffic via the IXP's switching fabric, they had to establish a bi-lateral BGP peering session through the IXP. However, as IXPs grew

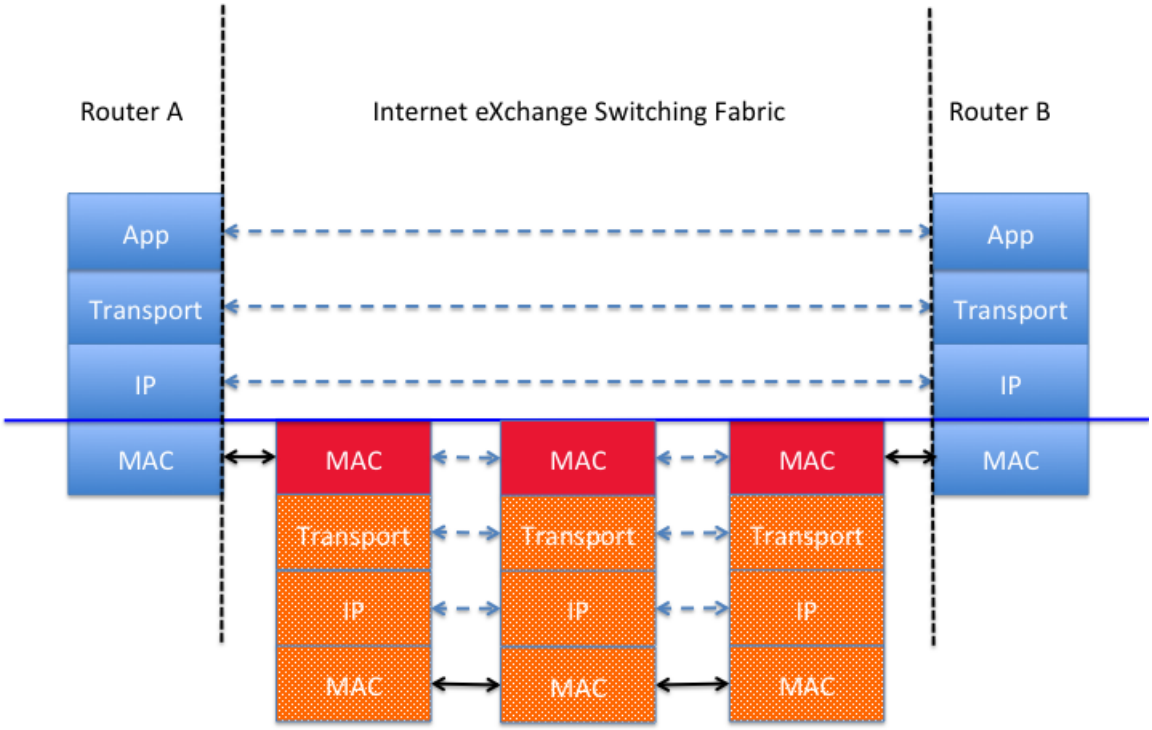

Fig. 2.7: Layer abstraction representation of MPLS based IXP fabrics

in size, to be able to exchange traffic with most or all of the other members at an IXP, and hence reap the benefits of its own membership, a member's border router had to maintain more and more individual BGP sessions. This started to create administrative overhead, operational burden, and the potential of pushing some router hardware to its limit.

Route Server (RS). To simplify routing for its members, IXPs introduced Route Servers [42] and offered them as a free value-added service to their members. In short, an IXP RS is a process that collects routing information from the RS's peers or participants (*i.e.*, IXP members that connect to the RS), executes its own BGP decision process, and re-advertises the resulting information (*i.e.*, best route selection) to all of the RS's peer routers.

If a route server service is offered, it supports both IPv4 and IPv6 and 4-byte AS numbers. The AS number used for the route server implementation is a unique AS number assigned by one of the RIRs. For redundancy, at least two RS are operated and are normally located in different PoPs.

The IXP IP space. In order to be independent of all of the connected parties, the IP space used on the "Public Exchange" is a Provider Independent space or other IP

space directly assigned by a IANA Regional Internet Registery (RIR). This applies to both IPv4 and IPv6. The IXP operator is responsible for obtaining address space from the respective RIR, as well as providing all material for justification, documentation, and applicable fees as required by the RIR.

2.4 IXPs and Net Neutrality

The network neutrality is a sensitive topic [12] [105] and transparency is often overlooked [77]. Transparency is what is required for a trustworthy neutral network. Neutrality is crucial for IXPs due to the amount of traffic exchanged through their facilities and the divergent operators, who are often competitors.

The simple definition network neutrality is: "the principle that all Internet packets should be treated equally."

Network neutrality from the network model layers like OSI or IETF point of view, advocates to operate a service at a certain network layer not to be influenced by any data other than the data interpreted at that layer. The IXPs are a layer2 fabric and need to stick to that layer to maintain a rigorous neutrality.

Accessing the IXP fabric is part of the neutrality. The locations of the IXPs fabric edge switches are important. All located data centers within their geographic reach should offer access to their switching fabric. To maintain their neutrality, IXPs non-profit organizations aim for diversity without sacrificing quality. They typically deploy in many different carrier-neutral facilities that are owned and operated by the different leading commercial data center and collocation companies, some of which operate successful for-profit IXPs themselves.

Due to their highly sensitive position in the Internet, IXPs have adopted a non profit type of organization. Particularly for the largest ones [@4][@27][@57][@72]. They are strictly neutral (i.e., open to any network and independent of third-party companies).

Network vendor lock-in is another relatively ignore aspect. Vendor lock-in makes a customer dependent on a vendor for products and services, unable to use another vendor without substantial switching costs. Network vendors have all their management approach and common line interfaces. IXP operators hardly change management philosophy and often remain attached to a specific "Command Line Interface". Secondly, network vendors are implementing network application in closed source using closed hardware [29]. IXPs also need to trust their network

vendor [@14]. The major network vendors produced closed source and closed source hardware box.

To help IXPs neutrality with a better vertical control of their entire stack of tools, software, and hardware equipment should be open source. Neutrality comes from transparency at all level, open source ease transparency.

2.5 The emerging new networking paradigm

Networking used to be simple. Nowadays, the number of control protocols defined by IETF or IEEE are colossal. Configuring a network requires knowledge of those control protocols work and how to configure them per vendor basis as there is no standardized configuration interface. The Internet and networks work because of the excellent operators abilities to master complexity. This paradigm change took some time to happen it could be explained by : The capacity to master complexity is not the same as extracting simplicity.

2.5.1 Abstractions Layers and network functions

Abstraction (from the Latin *abs*, meaning away from and *trahere*, meaning to draw) is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics. Through the process of abstraction, a software programmer hides all but the relevant data about an object in order to reduce complexity and increase efficiency.

Abstractions are key concept in programming [58] and in communication systems, the conceptual model Open Systems Interconnection (OSI) model is abstraction layers. The OSI abstraction layers are stacking layers of communications functions. It helps to visualize at the same layer end to end systems functions horizontally. The OSI model was defined in the 70s, when the Internet was incipient. The OSI model is still valid but history conducted networks vendors for many reasons to build networks equipments vertically integrated. The control plane that decides how to treat network traffic is bundled with the data plane which forwards this network traffic based on the control plane decisions.

The control plane which decides what to do with network traffic and the data plane that forwards that traffic are both tied together inside the networking equipment. The best analogy to feel how it is to operate a network today, is to picture yourself running through a glass maze. If you want to avoid collision you need to raise your arms in front of youself and walk instead of running [13] [85] [35] [52]. This situation reduces flexibility, delay innovation and any evolution of the network-

ing infrastructure. Besides that, network equipment are still proprietary close box and behave as old mainframe from the 70s, and are opposed to the open architecture initiated at the beginning of the 80s. Today equipment combines proprietary application-specific integrated circuit ASIC hardware, with closed often proprietary operating systems and closed proprietary network application.

Computer networks can be broken in three planes of functionality: the data, control, and management planes (the Fig. 2.8 give an illustration of he Layer network functions). The data plane corresponds to the networking equipment, which are responsible for forwarding data (often fully in hardware). The control plane serves the protocols used to populate the forwarding tables of the data plane elements. The management plane includes the software services, used to monitor remotely and configure the control functionality. Network policy is defined in the management plane, the control plane enforces the policy, and the data plane executes it by forwarding data respectively.

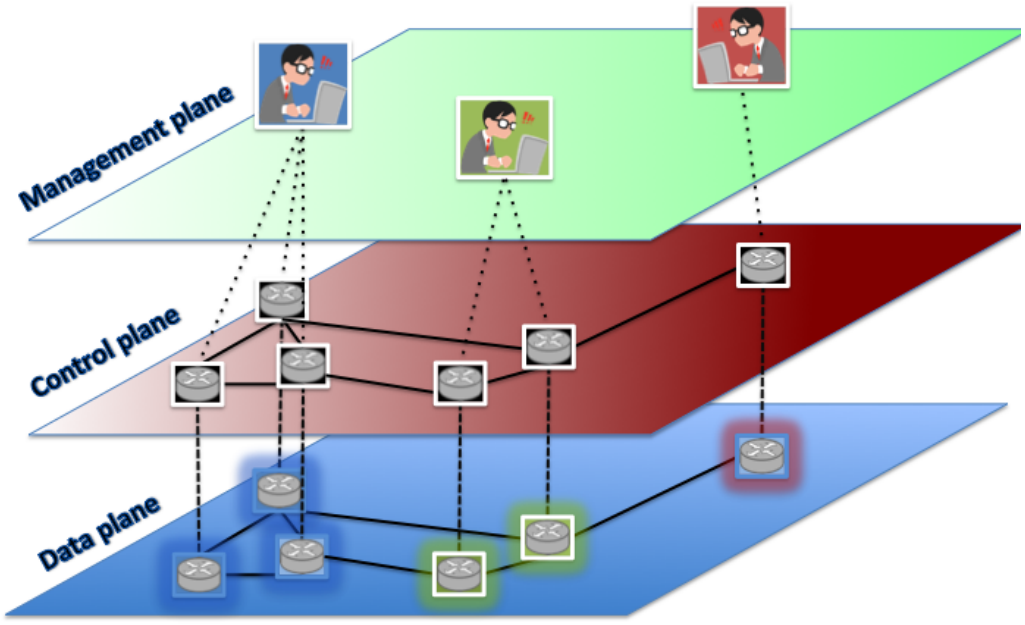

Fig. 2.8: Layered view of the networking functionality

2.5.2 What is the SDN ?

We define SDN as a network architecture by these following four points. A simplified view of this architecture is shown in Fig 2.9.

1. The control and data planes are decoupled. Control functionalities are removed from network nodes that will become dumb (packet) forwarding equipment.

2. Forwarding decisions are flow based, instead of destination based. A flow is defined by a set of packet field values acting as a match (filter) criterion and a set of actions (instructions). In the SDN/OpenFlow context, a flow is a series of packets between a source and a destination. All packets of a flow receive identical service policies at the forwarding devices [69] [36]. The flow abstraction provides a unified behavior of different types of network equipment, including switches, routers or firewalls and middleboxes [50]. Flow programming enables unparalleled versatility, restricted only by the capabilities of the implemented flow tables [64].

3. Control logic is transferred to an external entity, the SDN Network Operating System (NOS) or controller. The controller/NOS is a software platform that runs on commodity servers and provides the fundamental abstractions and resources to facilitate the programming of forwarding devices based on an abstract network centralized view. It is designed as a traditional operating system.

4. The network is programmable via software applications running on top of the controller/NOS that communicates with the data plane devices. This is a fundamental feature of SDN, considered as its main value statement.

2.5.3 OpenFlow

The decoupling between the the data plane and the control plane can be achieved by means of a well-specified programming interface between the switches and the SDN controller (as seen in Figure.2.9). The controller executes direct control across the state in the data plane components via this well-specified application programming interface (API). The most well-known example of such an API is OpenFlow [64] [@73]. An OpenFlow switch has one or more tables of packet handling rules (flow table). Each rule matches a subset of the traffic and performs matching and actions functions (dropping, forwarding, adding or modifying packet fields).

Depending on the rules placed by a controller application, an OpenFlow switch can be programmed to act as a router, a switch, a firewall or others type of network devices.

OpenFlow is a standard communications interface defined [@73] between the control and forwarding layers of an SDN architecture. OpenFlow allows direct

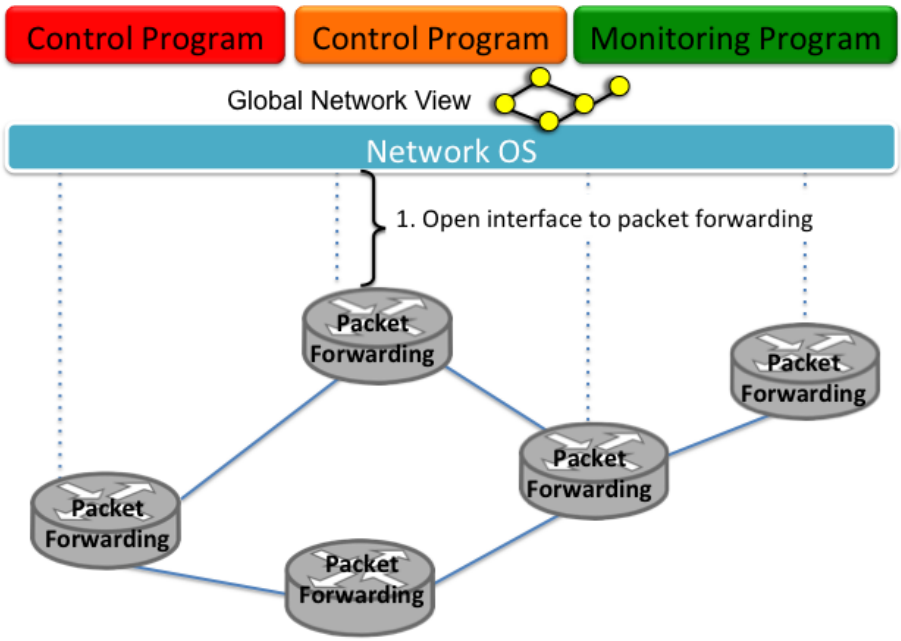

Fig. 2.9: SDN architecture

access for manipulating the forwarding plane of network devices such as switches and routers, both physical and virtual.

Section 2.3.7 shortly presents how complex large scale VPLS IXPs fabric are. Figure 2.10 presents with the same abstraction layer representation, how an ideal layer 2 SDN IXP fabric could be.

2.5.4 White-Box

White-box switches is the capability to use 'generic,' off-the-shelf switching (or white box switching), in the forwarding plane of a Software Defined Network (SDN). White-box switches are just that 'blank' standard hardware. They symbolize the foundational element of the commodity networking ecosystem required to enable organizations to select and pick the elements they need to fullfil their SDN objectives.

SDN gives access to the data and control plane abstraction layers with various open source projects, as we have seen in Section 2.5. Software project like Open Network Install Environment (ONIE) [10] contribute to free IXPs from the vendor lock-in.

[10] `http://onie.org/`

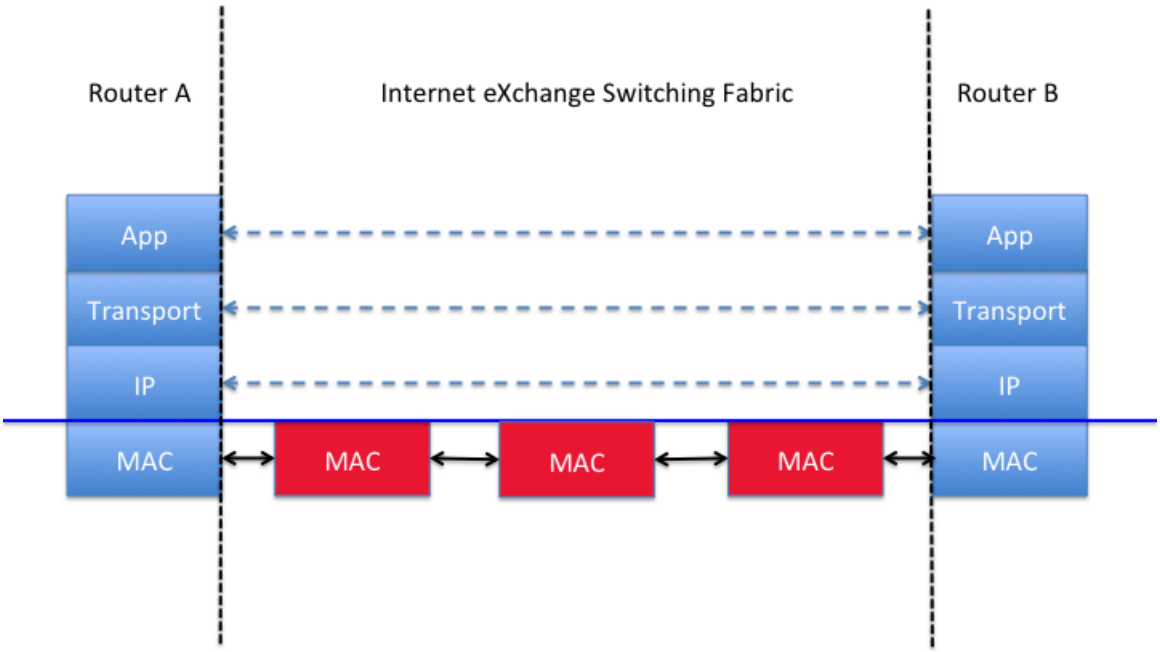

Fig. 2.10: Layer abstraction representation of an ideal layer 2 SDN IXP fabric

The Open Network Install Environment (ONIE) is an open source community that defines an open "install environment" for bare metal network switches. ONIE enables a white box and bare metal network switch ecosystem where end users have a choice among different network operating systems.

2.6 Conclusion

The historical Internet foundation from Robert Khan in section 2.1.2 has been overshadowed by the complexity of 40 years of exponential growth. Networking equipment is vertically integrated, and control and data planes are tied together, thereby not enabling fine-grain configuration or even programming of the data plane.

The heart of the Internet are the IXPs and their switching fabrics. The IXP operators play a key role in containing the exponential growth but the limitation of today's equipment restricts their choice and scalability.

The SDN new paradigm opens up new possibilities for proposing a dedicated IXP switching fabric to address these limitations. IXPs are layer 2 interconnecting facilities that offer neutral access. They must remain independent of layer 3 but provide an optional layer 3 service as the route server. This approach is explored in detail in the following chapter.

Umbrella architecture

3

> *"Modularity based on abstraction is the way things get done."*
>
> **— Barbara Liskov**
> (ACM Turing Award - 2008.)

3.1 The IXP environment

Internet eXchange Points (IXPs) are fabrics where Internet Service Providers, carriers, content providers and other Internet companies come together to exchange vast amounts of traffic [2]. IXPs are typically implemented as very simple layer-2 broadcast domains to which customers connect BGP-speaking routers and exchange layer-3 IP traffic. To make multilateral peering across the IXP fabric easier, many IXPs operate route servers [42, 87]. However, IXPs operate at the Ethernet level to ease physical interconnection.

Layer-2 Ethernet fabrics are at risk of network/component failure and network-loops, each leading to fabric failure. The IEEE Spanning tree protocols [98] [@45] are a possible solution for preventing loop. Multi Protocol Label Switching (MPLS) architectures like VPLS do scale and can react quickly and gracefully to some circumstances. However, Internet eXchange environments have a fairly static set of connected devices, changing only when a new customer router is physically installed or decommissioned. Such an environmentcan be optimized while still quickly reacting to network failures. To prevent loops, IXPs can deploy MAC address based access control filters. These filters ensure that, on a customer port, only traffic from the MAC address of the connected customer router is accepted. This reduces noise from unwanted traffic on the broadcast domain and eliminates Ethernet loops. Because IXPs are relatively static environments, new MAC addresses only appear or disappear from the fabric when a new router is connected or when a router is disconnected. Since the access control filtering requires that all customer MAC addresses are known to the IXP operator, it is possible to use them to program the forwarding tables of the IXP fabric as well, eliminating the need for active MAC address learning on the fabric. This creates the opportunity for SDN-enabled infrastructure, where the controller can program the devices through global knowledge of the network.

An IXP is agnostic about customers; thus customers may use routers from a variety of vendors and a range of capabilities. Routers with slower CPUs and (older) operating systems have problems handling the large amount of ARP traffic on larger IXP fabrics [104]. This would be significantly less problematic if ARP requests for a specific customer router were not sent to all customer ports. Ideally, the IXP fabric would send ARP and IPv6 Neighbor Discovery traffic only to the customer router for which the request is meant. Previous works have already demonstrated that OpenFlow (OF [64]) could be used to solve this problem [**boteanu2013**, 80]. However, these solutions require a continuously-active controller or software daemon that processes the ARP and Neighbor Solicitations, introducing scalability and stability concerns.

3.1.1 Current Practice

Figure 3.1 shows today's typical topology of a medium to large IXP [2]. The edge switches (also called access switches) connect the IXP member routers while the core switches aggregates all traffic thus interconnecting various physical locations. IXPs in Europe are usually located in big cities where multiple datacenters are present. IXP edge switches are commonly located in such datacenters allowing the interconnection among them. While some IXPs' topologies are characterized by the presence of only one hop in the core[1][2] (*i.e.* the packet traversing the fabric passes only one core switch), others allow multi-hop at the core level[3][4]. Umbrella forwarding approach, as shown in section 3.2, takes into account both configurations, making such an approach doable regardless the IXP topology. In addition, we show how in the first scenario is possible to enable the Umbrella ecosystem using legacy switches in the core, thus minimizing the amount of changes needed in a fabric to enable the proposed infrastructure.

There are two models for IXP operation. The older, now deprecated, is that the IXP exchanges all traffic between participating networks through a single router. This is usually called a layer-3 IXP. Nowadays, the most common model is the layer-2 IXP, in which each network provides its own router and traffic is exchanged via Ethernet switches [@51]. A layer-2 broadcast domain has by nature some side effects. All hosts belonging to the same broadcast domain can receive quite a significant amount of control packets, i.e., ARP requests, DHCP requests, discovery protocols: CDP or LLDP. Broadcasts packets increase the CPU utilization of the switches and decrease the overall network security, e.g., due to ARP spoofing. CDP (Cisco Discovery Protocol) packets in particular contain information about the network devices that might be useful for potential hostile conducts (i.e., software version, IP address,

[1] AMS-IX:`ams-ix.net/technical/ams-ix-infrastructure`
[2] DE-CIX:`www.de-cix.net/about/topology/`
[3] LINX:`www.linx.net/pubtools/topology.html`
[4] MSK-IX:`http://www.msk-ix.ru/eng/where.html`

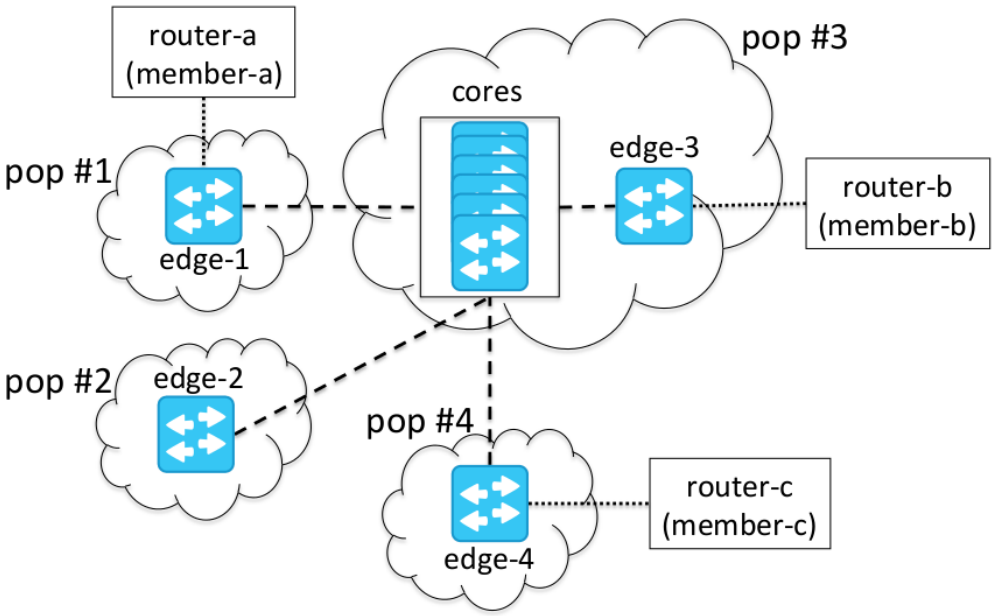

Fig. 3.1: Typical topology of a medium to large IXP.

platform, capabilities, and the native VLAN). Customer routers connected to IXPs typically exchange traffic with many other routers on the fabric. The larger the IXP, the higher the number of peers a router typically has. For all these peers, the ARP cache entry needs to be regularly refreshed. In addition, routers may have BGP sessions configured for peers that are not active. All together, the amount of broadcast ARP traffic on a large IXP fabric is already significant under normal circumstances. Even more ARP traffic is observed in downtime situations [34], when many routers attempt to resolve the IP addresses of peers that are not available because of a network outage, i.e., the ARP storm effect.

To study the amount of ARP traffic observed on a large IXP fabric, we analyzed traffic from AMS-IX and DE-CIX using the PCAP archive provided by the Hibernia IX monitoring service [40]. The Hibernia IX Monitoring service has been developed to ease debugging for Hibernia, Remote IX customers, Internet Exchange operators and regular Internet Exchange participants. The service provides traces recorded directly from interfaces connected to edge switches of various IXPs. We studied them to specifically understand:

- The growing rate of an IXP fabric.
- The amount of broadcast traffic in a large IXP fabric.

A key motivating observation is that IXP fabrics are growing steadily. We define the size of a fabric as the number of active peering router connected to it. As a

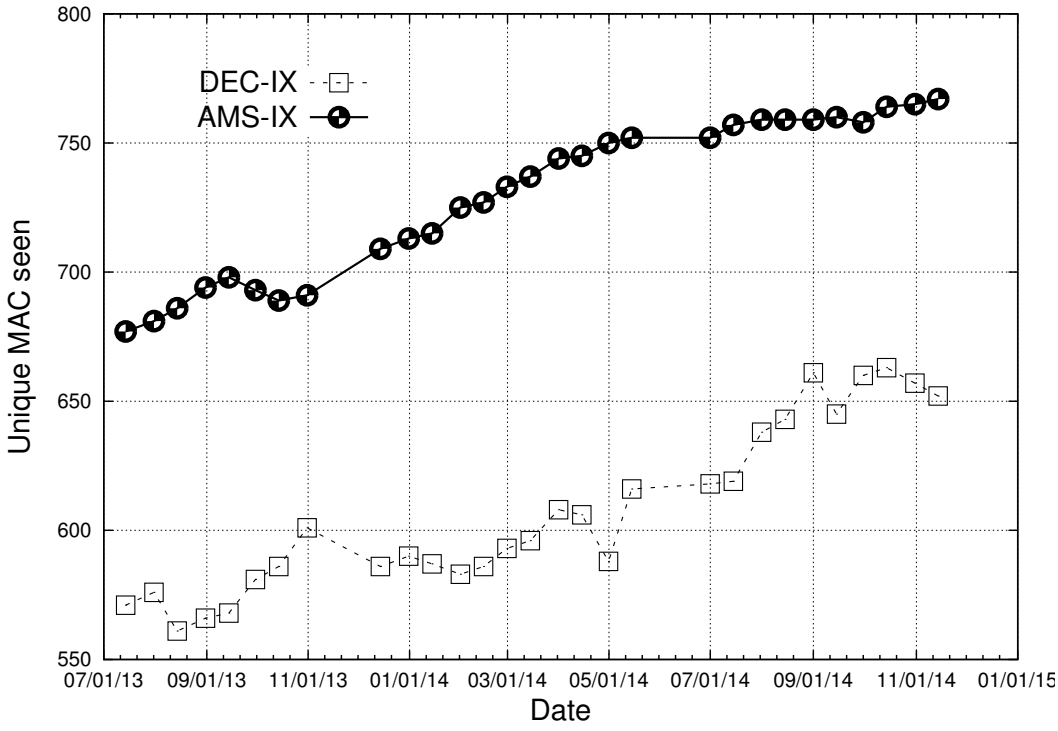

Fig. 3.2: Comparison of IXP fabric growth (DE-CIX and AMS-IX) - Source archived Hibernia PCAPs.

consequence of their size, both the amount of broadcast and neighbor discovery (IPv6 related) traffic increases.

Figure 3.2 compares the growth of the DE-CIX and the AMS-IX fabrics. Based on the aforementioned PCAP traces, we computed the number of unique MAC addresses seen every day. The plot shows a steady increase over time. Both fabrics experience a 10% yearly increase of unique MAC addresses. Although the values refer to the number of MAC addresses observed at the interfaces where traffic was captured, we expect the fabric to follow the same trend since they are made of a single broadcast domain.

Figures 3.3 and 3.4 show the amount of ARP requests and ICMPv6 Neighbor Discovery traffic (in frames per seconds) in the AMS-IX and DE-CIX fabrics. The large amount of ICMPv6 ND traffic is expected, since more and more services are moving towards a fully IPv6-based operation. Both figures reflect what we see in Figure 3.2. Larger numbers of participants in a shared broadcast domain translate into more location discovery traffic. In particular, over the last year, AMS-IX and DE-CIX have experienced an increase in the volume of location discovery traffic for both IPv4 and IPv6, of 50% and 25% respectively. This is a significant growth of broadcasted control traffic.

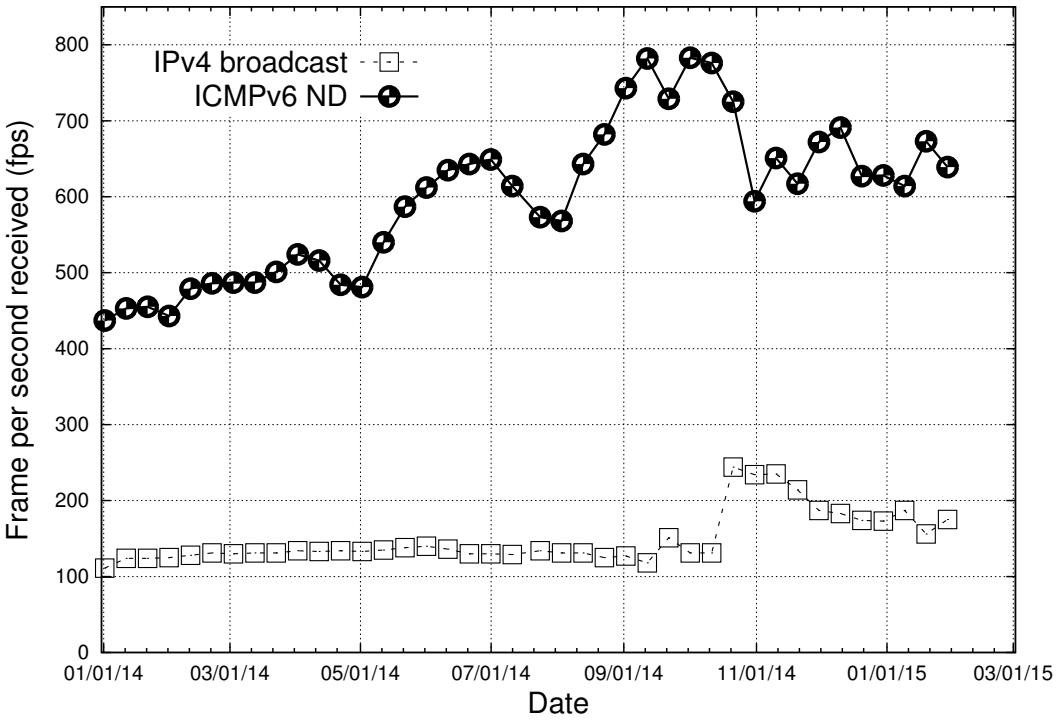

Fig. 3.3: Traffic report from AMS-IX - Source archived Hibernia PCAPs.

The amount of control traffic is even higher in the case of network outages, when many routers attempt to resolve the IP addresses of peers that are not available. The larger the network, the higher the probability of a failure. Given the growing amount of location discovery traffic under normal conditions, control traffic reduction techniques are necessary when an exchange starts to grow. ARP-Sponge [@5] represents the state-of-art solution to this problem. It manages to reduce control traffic to some extent. However, as we discuss next, it suffers from several defects.

ARP-Sponge

An ARP Sponge server listens on the ISP peering bridged domain for ARP traffic. When the number of ARP Requests for a certain IP address exceeds a threshold, the ARP Sponge server sends out an ARP Reply for that IP address using its own MAC address. From that moment, the IP address is *sponged*: all traffic to that node is sent to the ARP Sponge server. This prevents ARP storms by limiting the amount of ARP traffic. When the interface of a sponged IP address comes up again, it generally sends out a gratuitous ARP request packet. When the ARP Sponge server receives traffic from a sponged IP address, it ceases to sponge the IP address, thus no longer sending out ARP replies for that IP address. Although ARP Sponge prevents the escalation of ARP traffic, it suffers from several limitations:

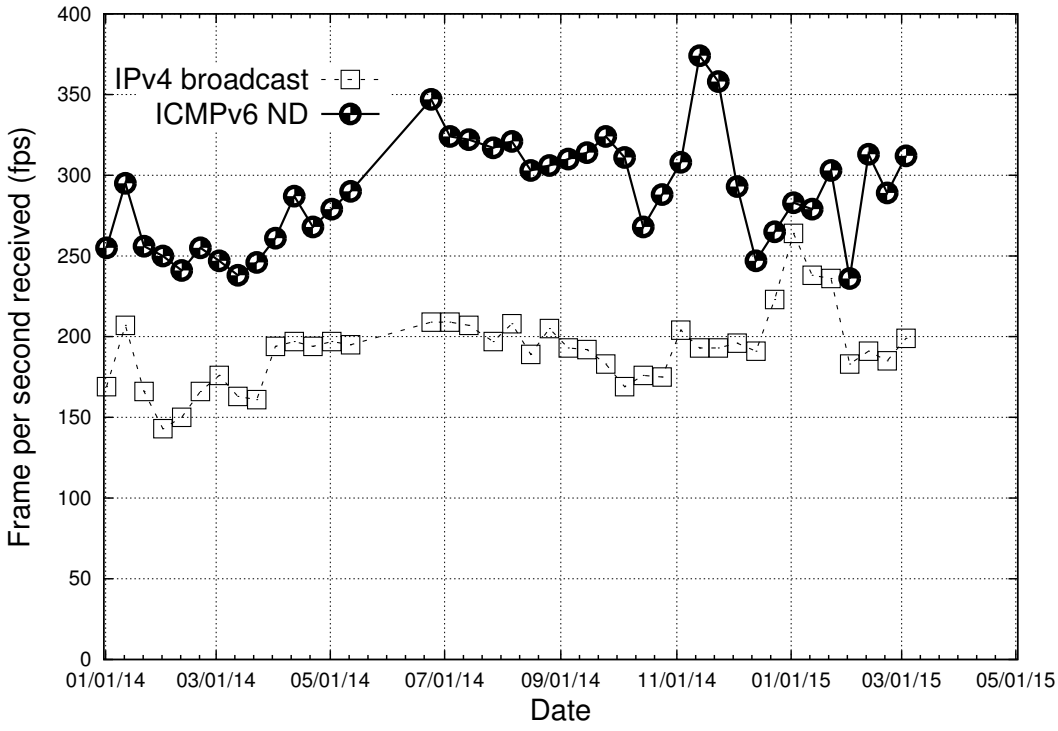

Fig. 3.4: Traffic report from DE-CIX - Source archived Hibernia PCAPs.

- It is a single point of failure. Even though multiple ARP Sponge servers can operate in parallel, this introduces complexity in the fabric.

- The ARP Sponge does not completely eliminate unwanted ARP traffic.

- It can cause the entire subnet to be re-programmed in the next-hop tables of customer routers. This is because the ARP Sponge server answers all ARP requests for all unreachable hosts in a given subnet. If the peering LAN subnet is, let's say a `/21`, each connected router will see *1021* addresses programmed in their next-hop table. Since ARP and/or next-hop table sizes have a limited size, this may be a problem if the router is connected to multiple peering LANs. This can be mitigated by having the ARP Sponge server only send replies for IP addresses that have been assigned to customers. However, this means that ARP requests for unassigned addresses remain unanswered, thereby not reducing the corresponding unwanted broadcast traffic.

- the ARP Sponge relies on heuristics to determine if an IP address becomes reachable again. In particular, it needs broadcasts or floods.

- The sponge server periodically sends ARP request for *sponged* addresses at a configurable frequency. If a reply is received then it *unsponges* the IP address

that replied. It may happen that the device that replied comes back but the sponge does not notice it[5].

The ARP-sponge therefore cannot be considered as final solution. Although it can reduce the effects of ARP storms, it suffers from limitations that make it unsuitable as a long-term solution. This motivates our architecture in Umbrella, which shifts typical control plane functionalities to the data plane, to improve scalability and reliability.

3.2 A new SDN architecture for IXPs

This section presents Umbrella, a new SDN-enabled IXP fabric architecture, that aims at strengthening the separation between control and data plane to increase both robustness and reliability of the exchange.

3.2.1 No broadcast traffic

IXPs apply strict rules [@3][43] to limit the side effects of a layer-2 shared broadcast domain, e.g., the router MAC address of the member that connects to the peering fabric must be known in advance. Only then the IXP will allocate an Ethernet port on the edge switch as well as an IP address from the peering IXP IP Public Space [71] and configure a MAC filtering ACL with that MAC address. As a consequence, the location of all member's routers is known to the IXP and currently it does not change as dynamically as assumed by the layer-2 protocols. In the future however, we believe that the IXP fabric should allow for arbitrary and dynamic locations of peering routers, which we want to enable in Umbrella. We exploit the ability of OpenFlow to rewrite the destination MAC address of a frame matching a given rule [75], enabling on-the-fly translation of broadcast packets into unicast. This can be exploited to eliminate location discovery mechanisms based on broadcast packets (i.e., ARP request, IPv6 neighbor discovery) and therefore remove the need for an active ARP-proxy daemon as proposed by previous SDN-enabled solution at the exchange [**boteanu2013**, **iSDX**, 80, 38]. Figure 3.5 shows a high-level vision of the proposed idea using Figure 3.1 as a reference scenario.

We propose a label-oriented forwarding mechanism to reduce the number of rules inside the core of the IXP fabric. Umbrella edge switches explicitly write the per-hop port destination into the destination MAC field of the packet. The first byte of the MAC address represents the output port to be used by the core switch.

[5]ARP-sponge manual: http://ams-ix.net/downloads/arpsponge

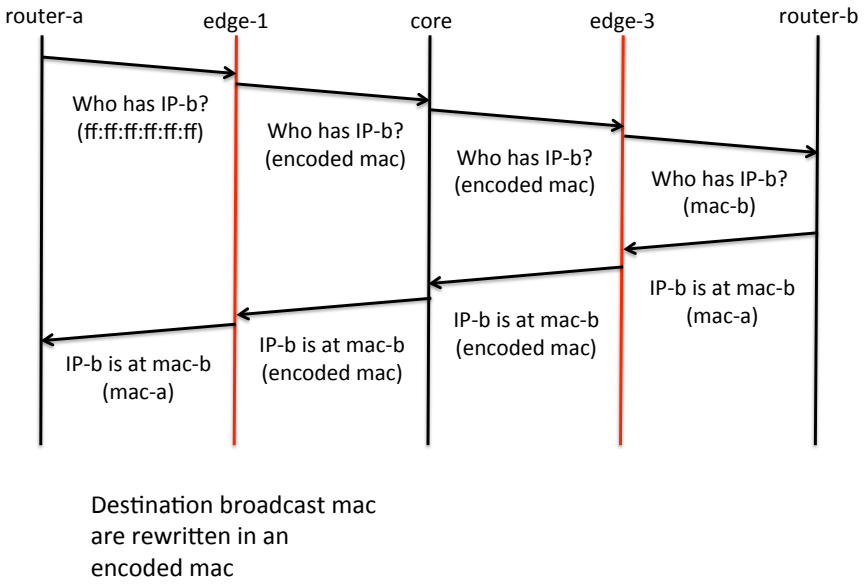

Fig. 3.5: Handshake in case of ARP request.

Table 3.1 shows an example of how a core switch flow table looks like in the Umbrella architecture. While this encoding scheme is currently limited to 256 output ports per hop, more bits in the port encoding (thus mapping more physical ports) can be used. The actual encode scheme taking advantage of usual ":" bytes separation. If a switch has more than 256 ports more than 8 bits can be used. The encoding needs to be selected in regards to the targeted topology. With Umbrella, the number of flow table entries per core switch will therefore scale with the number of active physical ports in the switch itself. This aspect is important to guarantee the fabric scalability.

Match on destination MAC (mask)	Output Port
01:00:00:00:00:00 (ff:00:00:00:00:00)	1
02:00:00:00:00:00 (ff:00:00:00:00:00)	2
03:00:00:00:00:00 (ff:00:00:00:00:00)	3
04:00:00:00:00:00 (ff:00:00:00:00:00)	4

Tab. 3.1: Core switch flow table example.

Let's take as example the topology shown in Figure 3.1 and consider the case where `edge-3` is connected to a core switch through port number 2 and to `router-b` through port number 3. Finally, let's take the case where `router-a` sends an ARP request (i.e., broadcast message) to `router-b`. `edge-1` receives the frame, rewrites the destination MAC address using the following encoding: `02:03:00:00:00:00` and forwards it to the right core switch. Once the frame reaches the core, it is redirected to output port 2 (i.e., the forwarding in the core is based on the most significant byte) to switch `edge-3`. Finally, `edge-3`, before forwarding the frame through the output port indicated in the second byte of the MAC address, rewrites that field with the real MAC address of `router-b`. In case the source and destination are directly connected to the same edge switch, no encoding is needed, and the broadcast destination address is directly replaced by the target MAC destination address by the edge switch. In an IPv6 scenario, the OF match pattern indicated in the edge switch needs to be on the IPv6 ND target field of the incoming ICMPv6 Neighbor Solicitation packet [67]. The matching table on the edge switch should maintain an association between IPv6 addresses and their location, as in the IPv4 case.

3.2.2 A label switching approach

The forwarding mechanism we proposed so far allows to reuse legacy switches in the core, limiting the burden (and costs) to upgrade an IXP fabric to the Umbrella architecture. In this scenario, a core switch only needs to forward packets based on simple access filtering rules, while the edge switches need OF-like capabilities to rewrite the layer-2 destination field.

While this approach is directly applicable to fabrics that rely on a single hop in the core (AMS-IX and DE-CIX), it is not applicable to fabrics with multiple hops (LINX and MSK-IX). With a single hop, the core switch would expect the output port to be encoded in the most significant byte of the destination MAC address. In the multi-hop case on the other hand, since a packet can traverse multiple core switches, a new encoding scheme is needed to differentiate the output ports at different core switches.

Figure 3.6 shows an example with multiple hops in the core. `edge-a` needs to cross two different core switches through `path b` to reach `edge-d`. This is a fairly common case in hypercube-like topologies, like the ones adopted by LINX or MSK-IX. In this scenario, following the Umbrella approach, we might propose an encoding of the layer-2 destination address where the most significant byte refers to the output port of the first core switch (i.e., `core-a`), the second byte to the second switch (i.e., `core-b`), and so on. Unfortunately, depending on the actual route being used, a core switch might be the first or the second on the path, making the proposed

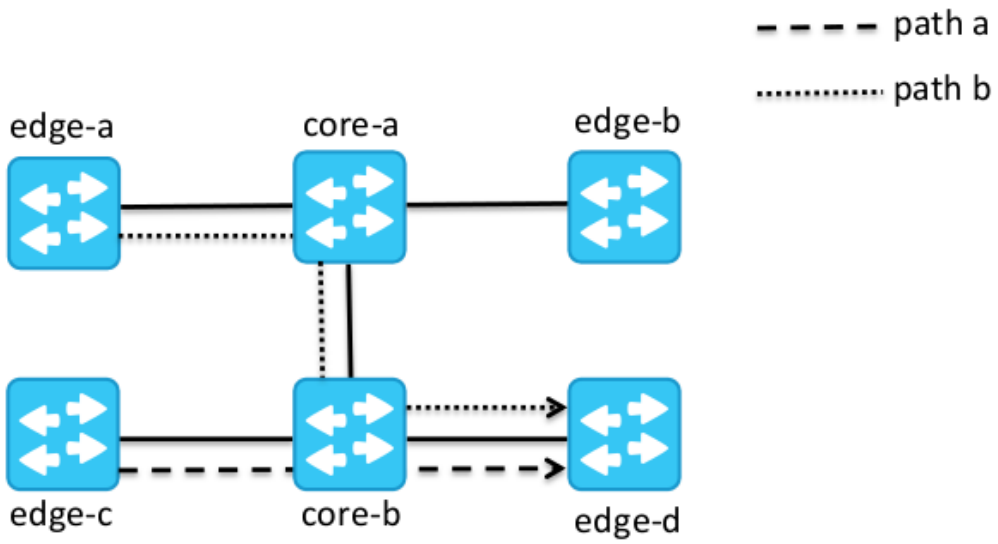

Fig. 3.6: Example of multi-hop in the core.

approach unfeasible. Another solution is to take into account also the `input port` of the frame in the forwarding rules installed in the core switches. Given the input port, it is possible to know where the switch is on the path and therefore look at the right byte in the layer-2 destination address. Unfortunately, this approach may not work in arbitrary topologies neither because of cores incapacity to differentiate path. Moreover, it will experience a rule explosion in the core, i.e., the number of forwarding entries grows quadratically with the number of possible input ports, making such an idea not very attractive..

These problems can be addressed using a label switching approach. Umbrella leverages source routing: the first edge switch is in charge of selecting the path. An ordered list of `output port` is encoded in the destination MAC address as a stack of labels. Each core node processes the frame according to the value on the top of the stack, and pops it before forwarding the frame (Figure 3.7).

With this configuration each switch needs only to look at the most significant byte of the address, no matter where it is on the path toward the destination. Popping out from the MAC destination address the last used label requires header rewriting capabilities, making this solution feasible only when OpenFlow-enabled switches are used in the core. In particular, every core switch must have 2 action tables: forwarding and copy-field[6]. This solution comes with two main practical limitations (which will be addressed in Section 3.4.2):

[6]OpenFlow 1.5 specifications allow to copy and rewrite part of an header field

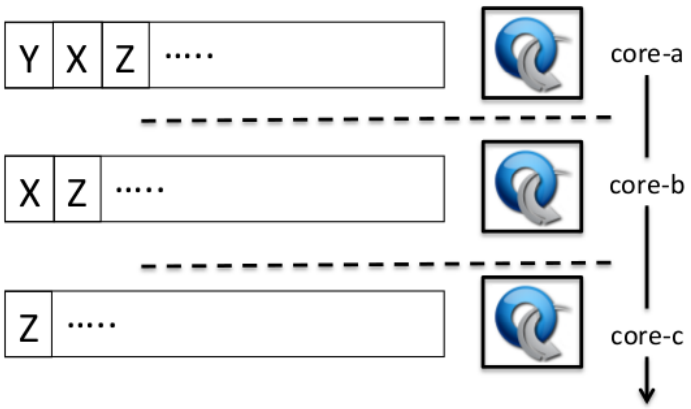

Fig. 3.7: Example of the Umbrella forwarding scheme.

- The maximum number of output ports that can be addressed per-hop is 256, as we embed the output port for each core switch in the most significant byte of the layer-2 destination address.
- The maximum number of hops inside the IXP must be less than 6, as we use the 6 bytes of the MAC address to encode the whole path of the frame.
- The paths are all determined by the controller, in case of an error in the path controller decision and the label with zero is reached, and the packet is dropped.

Despite these practical considerations, Umbrella can actually be seen more as a generic approach for IXP fabric operation, not just an ad-hoc solution to specific layer-2 issues. Therefore, we believe that the general concept of Umbrella enables a stronger separation between the control and data plane.

3.2.3 Umbrella and Route Servers

Originally, if two IXP member ASes wanted to exchange traffic through the IXP's switching fabric, they had to establish a bi-lateral (BL) BGP peering session at the IXP. Each BGP node connected to the fabric who wanted to obtain information about all reachable networks prefixes at the IXP had to establish BGP peerings using TCP connections with other IXP members. As IXPs grew in size [20], this solution proved to be impractical, as that implied monitoring too many BGP sessions, with the associated administrative overhead, operational burden, and risks to push IXP members router hardware to there limits. IXPs introduced `Route Servers` as an answer, and offered them as a free value-added service to their members [87]. Route Servers (RS) store all the incoming route information from IXP members and forward them without modification to other members. Thanks to the RS, an IXP member

can receive all the routing information available to the IXP through a single BGP session.

Umbrella can be used to handle the forwarding of BGP traffic within an IXP, both for standard bi-lateral peerings and RSs. For bi-lateral BGP sessions, the TCP connection is treated as pure data plane traffic crossing the IXP, and the traffic is handled by the switch rules, without control plane intervention. For RSs, the BGP traffic entering the fabric is directed to the RS thanks to a single rule at the edge switch, while the egress traffic is handled automatically through the existing rules on the edge switches. BGP traffic is very important as it provides reachability information, and therefore would justify preferential treatment inside the IXP fabric. Even though congestion is rare inside IXP fabrics as they tend to be over-provisioned, in the rare case of a DDoS [83], protecting specific control plane traffic such as BGP would at least prevent the attack from disrupting reachability inside the IXP fabric, despite not addressing its effect on the data plane. This example illustrates how simple the handling of specific control plane traffic can be with Umbrella.

3.2.4 Failure detection and recovery

Group fast failover is the mechanic since OpenFlow 1.1 to react to a link failure. In particular, a fast failover group table can be configured to monitor the status of ports and interfaces and switch forwarding actions accordingly, independent of the controller. Recovering from data plane failure is more challenging. In this scenario an active probing of the data plane status from the controller is needed. In particular, the Umbrella controller can be instructed to implement the LLDP Local Link Discovery Protocol [97] or BFD Bidirectional Forwarding Detection [1]. Once a data plane failure has been detected, Umbrella controller changes only the edge switch configuration with a preset path.

3.2.5 The case for a stronger control and data plane separation

Previous works have already demonstrated that OpenFlow (OF [64]) could be successfully used at the exchange [**boteanu2013**, 80, 38]. They consider an IXP fabric controlled by a central controller that acts as the control plane for all the peering routers at the IXP. In such an architecture, it would be natural to co-locate the SDN controller with the route server to ensure that both the SDN and the BGP control planes are able to talk with each other with minimal delay [38]. Figure 3.8 illustrates a SDN-enabled IXP where all the control messages that are exchanged by peering routers are sent to the SDN controller, that then acts on behalf of the peering routers and programs the switches of the IXP fabric.

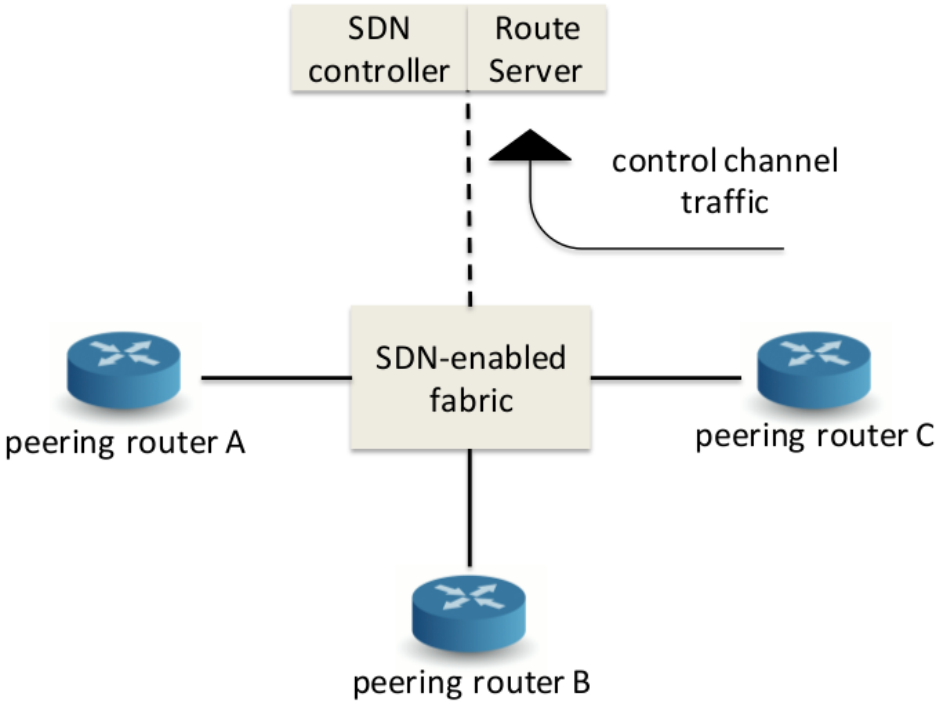

Fig. 3.8: SDN-enabled IXP: a case scenario.

We emulated the proposed scenario on Mininext [96], an extension layer to build complex networks in Mininet. We instantiated two virtual containers acting as peering routers, one for the Route Server and two more working as client hosts directly connected to one peering router each. The emulation never total CPU utilization never overused the hosting server capacity. We represented the exchange point as a single open vSwitch coupled with a Ryu [92] controller acting also as ARP-proxy (as suggested in the related literature [**boteanu2013**, 80, 38]).

One issue with such a design however is the coupling between the control and the data plane. Indeed, issues within the data plane may affect control plane messages, and therefore lead to a slow or unresponsive control plane, further aggravating the effect on the data plane. For example, imagine that ARP messages of a peering router A get delayed inside the IXP fabric to reach the SDN controller: this would impact all BGP sessions between router A and its peers. We show on Figure 3.9, the period of time during which the data plane or BGP would be disrupted, if an ARP message is delayed by a given amount of time. Figure 3.9 shows that even a small delay of a few tens of milliseconds for ARP messages may trigger disruptions on the data plane or on BGP that are multiple times larger than this delay. We used tc qdisc on the link between the switches and the controller containers. Given the amounts of traffic exchanged within IXPs fabrics, such disruptions of the data plane are not an acceptable cost of enabling SDN on the fabric. Therefore, we advocate a stronger

separation between the control and the data plane. Our approach is one possible solution towards a stronger separation.

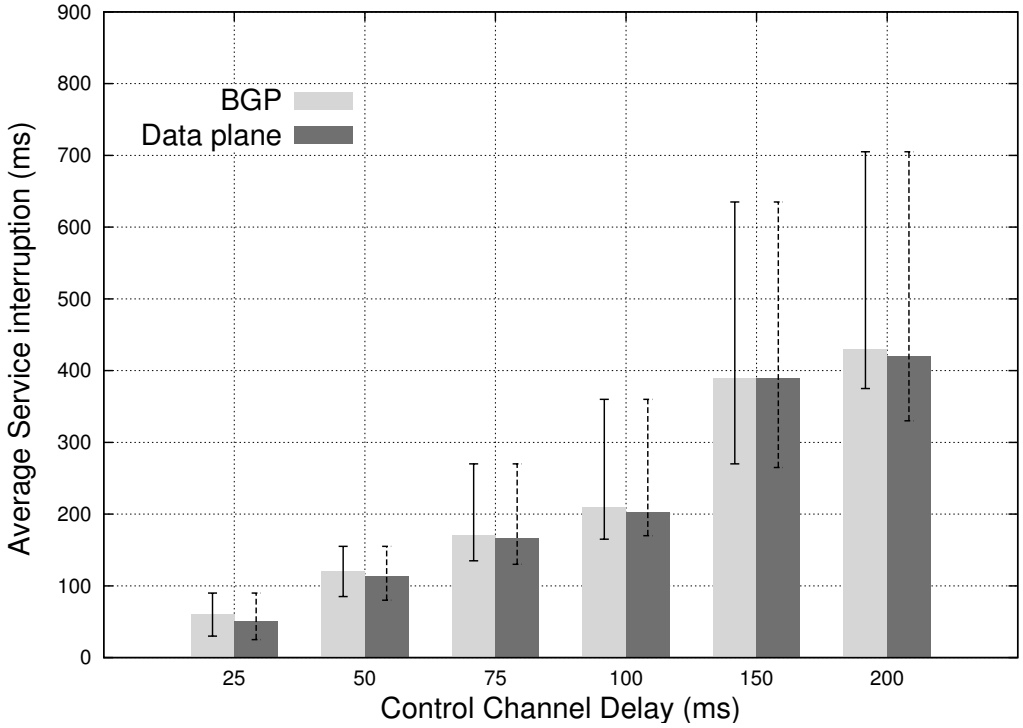

Fig. 3.9: The dependency between control and data plane.

3.3 Key benefits

Umbrella has been designed to be flexible. Indeed, it can be made layer-3 neutral or service (i.e., application) oriented, depending on the settings being used. It addresses the issues of a shared broadcast domain using layer-2 manipulation at the edge to enhance legacy fabric reliability. This section discusses the key benefits that Umbrella brings for an exchange.

Scalability & Reliability

The proposed encoding structure requires a small amount of rules in both core and edge switches (as we will demonstrate in Section 3.4) thus enhancing the overall scalability of the fabric. In addition, it alleviates the controller from processing location discovery packets, decreasing the data plane dependencies over the control plane. This is an important aspect, as the Umbrella data plane can keep operating even in scenarios where the control channel performances are poor.

Most legacy IXP architectures alleviate ARP storms through the ARP Sponge server approach. At the same time, pure layer-2 SDN-based architectures leave to the

controller the processing of location discovery traffic (i.e., ARP and NDv6). Both solutions suffer from scalability issues given the growth of IXP fabrics in terms of new MAC addresses, leading to a constant increase of broadcast traffic.

No central point of failure

Solutions relying on a single point for common operations, i.e., the controller for SDN-based architectures and ARP Sponge server for legacy IXPs, are subject to the single point of failure problem. Umbrella does not need the constant presence of the controller for such operations. The controller works in a pure proactive mode and it is only required to add, remove or change a router MAC entry at the edge. Given the static nature of IXPs in terms of routing [8], the controller does not have a central role in the Umbrella approach.

Service-oriented IXP operators

The label switching nature of the forwarding mechanism opens the possibility of making the IXP fabric service-oriented. A service-oriented IXP operator could create catalogues of network resources and related policies (e.g., QoS parameters and bandwidth) to which applications can be applied to a given flow from edge to edge port or matching others OpenFlow fields. As the path inside the IXP fabric is configured at the edge switch, it is possible to configure different paths for different applications, or redirect some flows to different paths depending on the activated services (e.g., firewalling, quality of service, monitoring). Note that this is just a feature which *can* be enabled. Indeed, Umbrella can also be used in a totally layer-3 (and above) neutral way, as currently done by IXPs.

Compatibility with legacy switches

As discussed in the previous section, if the topology being used in the IXP has only one hop in the core, legacy switches with MAC policy based routing access lists can be used in the core with Umbrella. Indeed, no additional feature than bitmask layer-2 destination matching and forwarding actions are applied in the core, making the architecture compatible even with non-OpenFlow compliant switches.

Pseudo-wire nature

Pseudo-wire [18] is an emulation of a point-to-point connection over a packet-switching network. As discussed above, with Umbrella, all the broadcast traffic (both ARP IPv4 and ICMPv6 ND) is converted to unicast at the edge, solving problems

related to a shared broadcast domain. Umbrella guarantees that each of the IXP members receives only the traffic it is supposed to see, also saving computational power at the edge for the processing and analysis of unwanted traffic, e.g., broadcast packets.

Visibility

In Umbrella, the actual path of the packets is encoded in the layer-2 destination address. This implies full visibility of the forwarding paths inside the IXP fabric, which can be exploited to improve data plane troubleshooting, and therefore general IXP operator management.

The adopted forwarding mechanism increases network visibility. The actual path of the packet is encoded in the L2 destination address simplifying the fabric debugging process. The path can be used to identify both possible misconfiguration or wrong configuration of the fabric. Improving the network awareness is a key aspect to improve the overall manageability.

3.3.1 General SDN-related benefits

Many others key benefits of the Umbrella ecosystem derive directly from the SDN approach derive from the proposed architecture. In the following we enumerate some key benefits that an SDN-based architecture can bring to a legacy IXP ecosystem.

Vendor-neutral and Future ready

A key point of the SDN paradigm concern the possibility to open and expose the control plane thus enabling the rising of the whitebox concept. IXP operators can change easily OpenFlow switch vendor without any modification on the control plane [21]. The Umbrella architecture uses only one optional OpenFlow matching field *ARP TPA* and *ICMPv6 ND TPA* to increase portability.

Manageability

Traditional networking devices support essentially two kinds of statistics gathering - per interface (or per VLAN, per ACL entry, or other overlay) byte and error counters, and packet sampling (sFlow et al). Interface and ACL counters require architectural context to interpret correctly (*e.g.*, for this ACL counter to increment. Where would the packet have come from?). Sampling requires not only architectural context but

post-sample parsing and correlation (*e.g.*, to recover source and destination layer 2 addresses, or to resolve a packet AS of origin at the time the packet was sent). This is particularly important when network architecture changes - ACL counters may have to be redeployed and sampling post-processing may have to be reinterpreted to make sure counters and samples both before and after the change can be interpreted in their correct architectural context. An SDN-enabled architecture folds overlay and interface counters together to build in architectural context. For example, to answer the question *how many IPv6 packets does peer A send to peer B*, one can inspect the counters on flows associated with B destination, on the network edge closest to A interface counters associated solely with A interface or B interface. That would not be sufficient to separate out this traffic, and using sampling would only give an approximate answer. Furthermore, because flow rules can be specific to types of traffic of interest, these can be broken out without affecting forwarding. For example flows matching the Ethernet broadcast address and from a certain Ethernet source address can easily be accounted for.

Security

OpenFlow-enabled switches can be used to enforce security and routing policies, thus minimizing well-known BGP potential protocol security risks [19]. The BGP router server information could be used to enforce the Umbrella switching fabric with the member announced policies with the BGP communities.

3.4 Experimental Evaluation

In this section, we evaluate several aspects of the Umbrella architecture. We first estimate the average number of flow rules needed at the edge, taking into account different IXP fabric sizes (Section 3.4.1). We then discuss the applicability of the label switching approach in real scenarios (Section 3.4.2). Finally, we estimate the impact of the ARP proxy, i.e., of the control plane directly relevant to Umbrella, over the data plane performance, and compare the results with the Umbrella approach 3.4.3

3.4.1 Flow table occupancy at the edge

Mapping the broadcast destination MAC address to the path inside the fabric requires additional flow rules at the edge of the fabric. This section provides a scalability analysis in terms of the required number of rules. From an ingress traffic point of view, the proposed architecture requires the OpenFlow switches at the edge to store three entries per peering router connected to the fabric. This is a necessary and

sufficient condition to enable routing in any situation (i.e., IPv4 ARP, IPv6 Neighbor Solicitation and data plane traffic). In the following we show a sample of the three entries referred to a given peering router:

“eth_type":0x806, “arp_op":1, “arp_tpa":“1.1.1.3",
“actions":[“type":“SET_FIELD",“field":“eth_dst",
“value":“18:01:00:00:00:00", “type":“OUTPUT",
“port":“49"]

“eth_type":0x86DD, “ip_proto":58, “icmpv6_type":135,
“ipv6_nd_target":“2001::1/128",
“actions":[“type":“SET_FIELD", “field":“eth_dst",
“value":“18:01:00:00:00:00", “type":“OUTPUT",
“port":“49"]

“dl_dst":“00:00:00:00:00:03",
“actions":[“type":“SET_FIELD", “field":“eth_dst",
“value":“18:01:00:00:00:00", “type":“OUTPUT",
“port":“49"]

The first entry handles the ingress IPv4 ARP traffic. It is important to lookup for the IP address of the peering router towards which the location discovery traffic is targeted. The destination MAC address field (i.e., broadcast) will be rewritten with a pre-defined path through the fabric. The second and third entries refer to IPv6 Neighbor Solicitation and data plane traffic. Note that all three entries share a common action in the rewriting and forwarding process, because Umbrella treats location discovery and data plane traffic in the same way. From an egress traffic point of view, each edge switch needs to rewrite the MAC destination field of the received frame with the correct target. As the value to be inserted in the destination MAC field depends on the output port of the edge switch, the number of rules here depends on the number of peering routers connected to the edge switch. The total number of rules for an edge switch is then the sum of the ingress and the egress ones.

Figure 5.5 shows the average number of rules needed per edge switch if the Umbrella architecture was to be used at different European exchange points. As these numbers depend on the number of peering routers connected to the fabric, we analysed data from PeeringDB [79, 61].

Given the number of flow entries required as shown on Figure 5.5, it appears that the Umbrella architecture is applicable in today’s IXP. Indeed, today’s OpenFlow-enabled

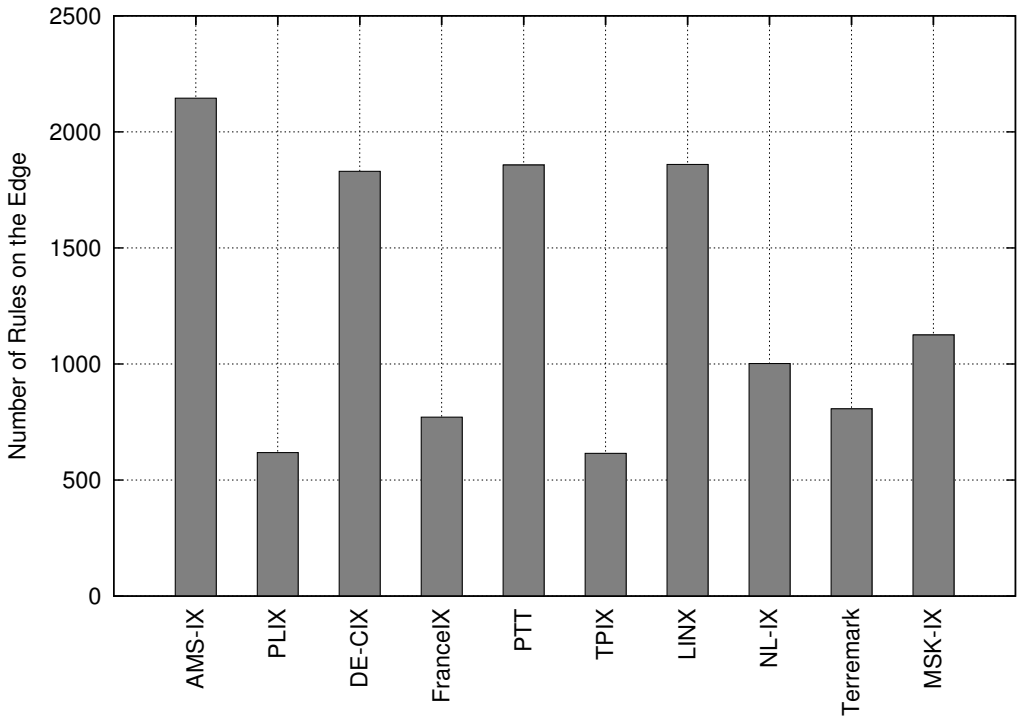

Fig. 3.10: Average number of rules required at an edge switch.

switches can already support from a few thousands flows (e.g., Pica8 switches[7]) to hundreds of thousands (e.g., Corsa[8] and Noviflow[9] switches).

3.4.2 Number of hops through IXP fabric

We already mentioned one of the practical limitations of our approach in Section 3.2.2: the maximum number of hops inside the IXP must be less than 6, due to our use of the destination MAC address field to embed the path of the frame at the edge (8 bit per hop). While this requirement is typically met by design in currently deployed fabrics which allow only one hop in the core, this might not be the case when multi-hop in the core is permitted, especially in the case of backup/recovery after some link failure, where routing inside the fabric may be allowed to follow longer paths. We therefore studied the publicly available topologies of the four largest European IXP fabrics, and estimated the number of hops a packet must traverse at the exchange in the worst case.

Figure 3.11 shows the per fabric maximum number of hops in case of normal operation and fallback due to a massive link failure. We define a massive link failure as the maximum number of link failures that the fabric can sustain while

[7] http://pica8.org/blogs/?p=565
[8] http://www.corsa.com/sdn-done-right/
[9] http://noviflow.com

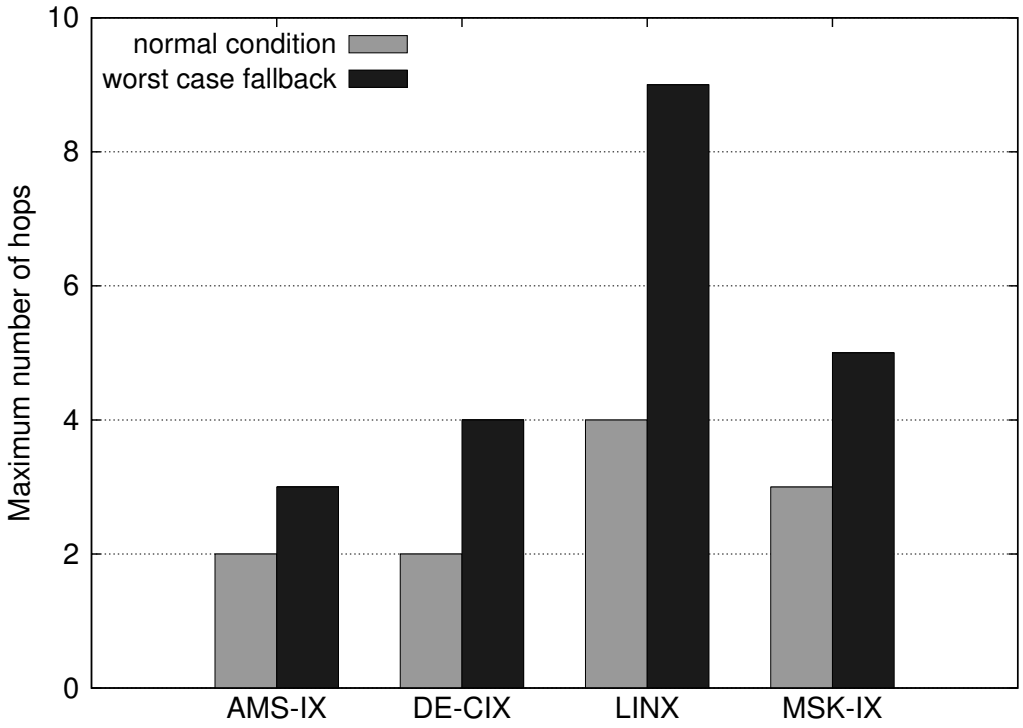

Fig. 3.11: Average and worst-case number of hops in large European exchange points.

still guaranteeing connectivity between each peering router connected to the fabric. The results shown during normal operation for the AMS-IX and DE-CIX fabrics are expected, since they allow only one hop in the core. The main difference between the two fabrics comes in case of a massive link failure. While AMS-IX duplicates the core switches (leading to a maximum of three hops), DE-CIX uses four core switches connected in a full mesh topology (leading to a maximum of four hops). Although our requirement is not met in the case of LINX in the fallback/recovery situation, we believe that this is actually not a real limitation of the proposed solution, as such failures would disrupt significantly the fabric anyway, leading to severe congestion inside the fabric. The LINX topology can be seen as a hypercube where each vertex is a core switch. The maximum number of hops during a massive link failure situation can therefore be as high as 9. If necessary, such situations should be handled through protection mechanisms, e.g., pre-computed MPLS paths, as the traffic pattern through the remaining part of the fabric will have to be carefully engineered.

3.4.3 Impact of unreliable control channel on data plane performance

To estimate the impact of an unreliable control channel on the data plane performance, and thus evaluate the benefit introduced by the Umbrella architecture, we

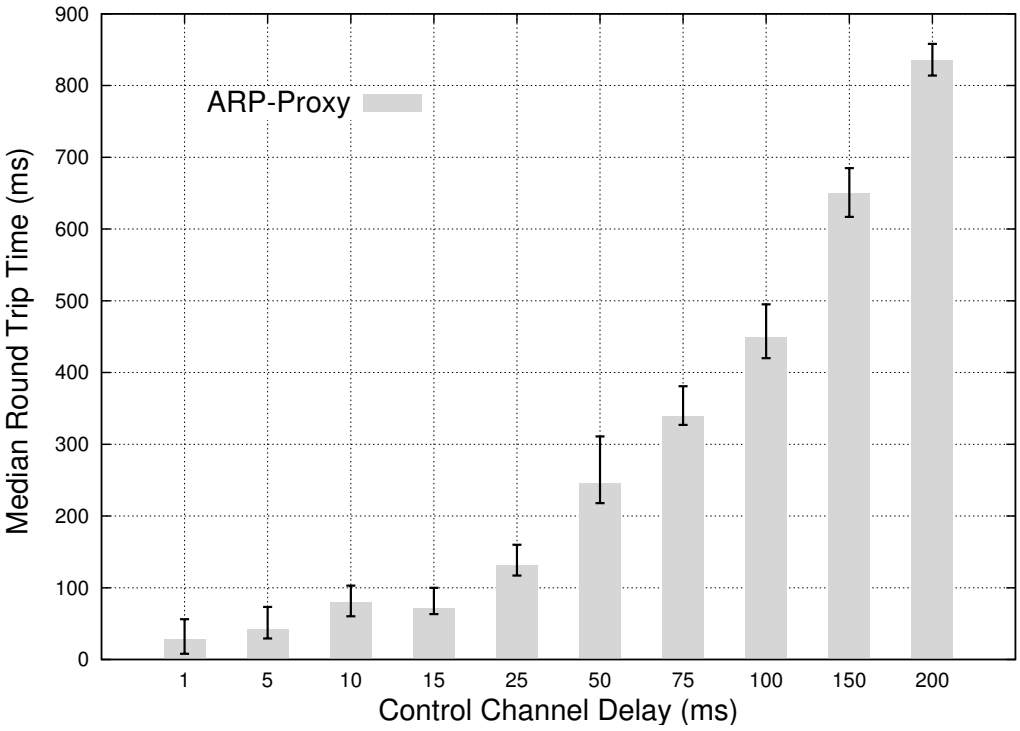

Fig. 3.12: Round trip time of an ARP packet when ARP-Proxy is active, the ARP-cache is empty and the control channel experiences different delays.

carried out a test campaign divided in three different phases. We first compared the effect of delay on an ARP-proxy based solution and Umbrella over a simple two hosts connection. Secondly, we studied the effects of packet loss in the same scenario. Finally, we performed a long run emulation of a bigger topology to understand the effects in a less constrained environment, closer to reality.

Impact of delay in the control channel

We emulated on Mininext [96] a scenario where two hosts are connected to each other through Open vSwitch. We pre-installed two flow rules to enable bi-directional communication between them, and another rule to redirect ARP packets to the Ryu controller, where the ARP proxy application is running. Figure 3.12 shows the median round trip time of ARP packets in the proposed scenario (the time between the ARP request is sent and the answer is received by the host), when the ARP-cache of the sender was empty and the control channel was affected by different delays. We see in Figure 3.12 that any delay in the control plane escalates into a RTT that is multiple times the added delay. This illustrates the strong relationship between ARP-cache entries and the control plane performance when an SDN architecture relies on the ARP-proxy mechanic.

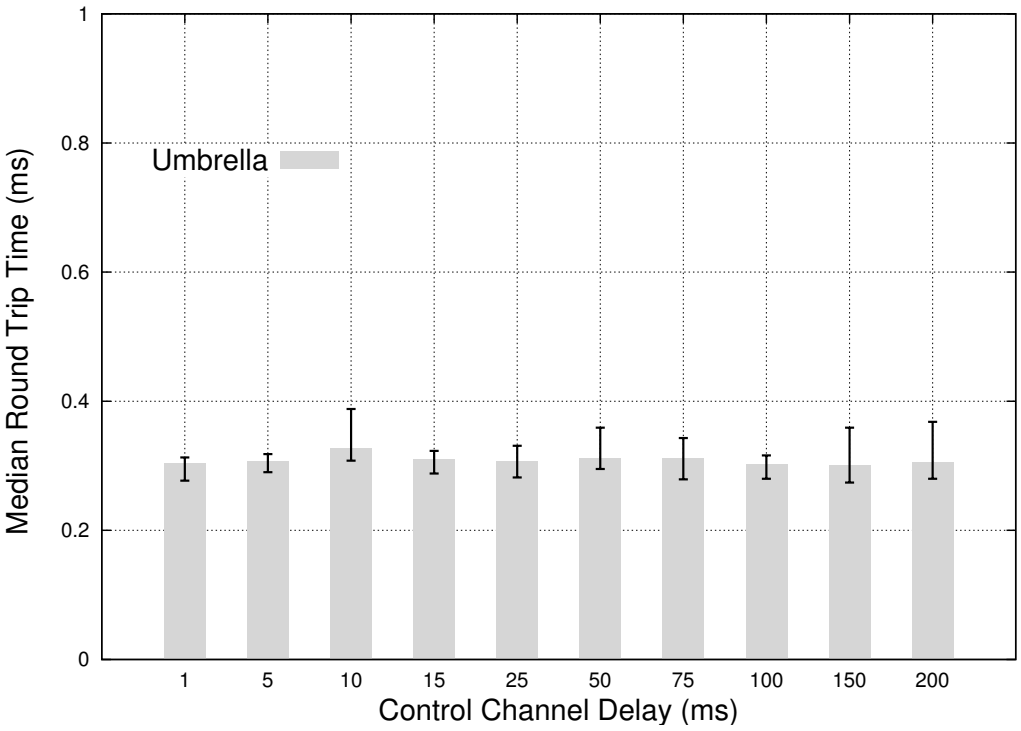

Fig. 3.13: Round trip time of an ARP packet with Umbrella, the ARP-cache is empty and the control channel experiences different delays.

Figure 3.13 shows our test result using mininext as section 3.2.5, as expected by design, the behavior of the control plane does not affect the round trip time of the ARP packets in the Umbrella case. Note that the test is also performed with an empty ARP-cache at the sender. With Umbrella, the ARP request sent by the host is forwarded through Open vSwitch to the receiving host, that sends the response directly to the sending host, without going through an ARP-proxy.

As IXPs are growing in number [11], size [20], and relevance [2, 23], it is important to restrict as much as possible dangerous dependencies between the control and the data plane, especially when the ARP-cache of a peering router fills up or an entry expires. Figure 3.14 shows the ratio between the RTT of ARP packets when a delay is introduced on the control channel and without the delay, as a function of the delay. We observe that as the delay on the control channel increases beyond 50ms, the ratio increases more than 10-fold when an ARP-proxy is used. Umbrella on the other hand is completely insensitive to control plane delay, and has a ratio very close to 1 for all values of delay.

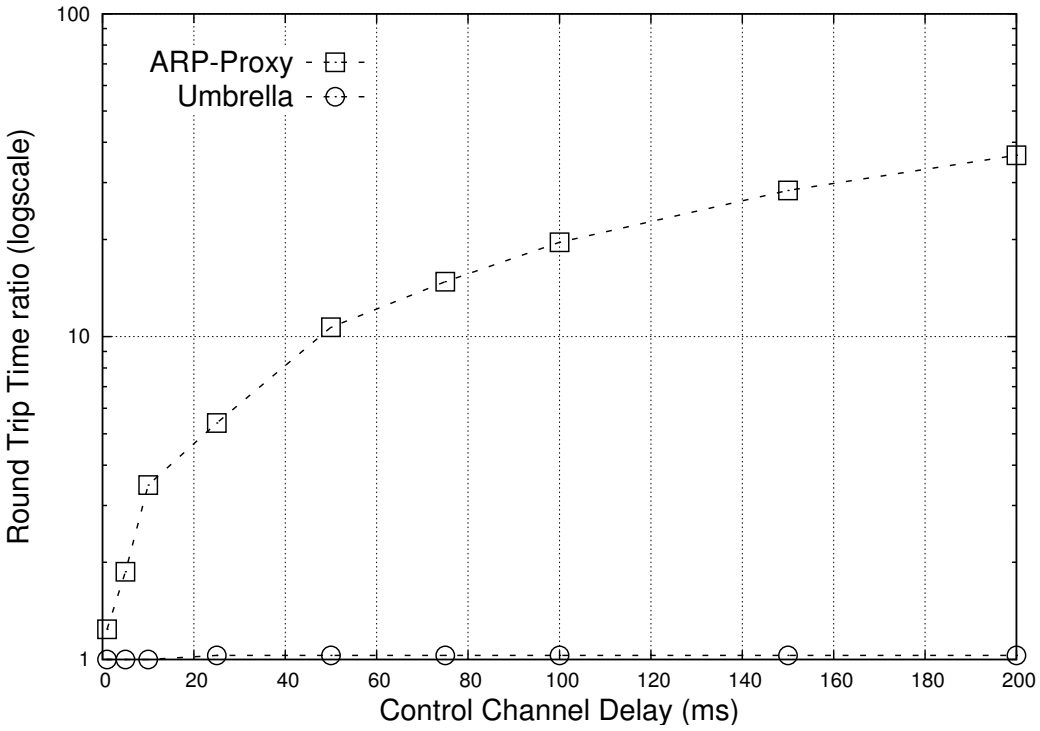

Fig. 3.14: Ratio between the RTT of ARP packets when a delay is introduced on the control channel and without the delay, as a function of the delay.

Packet of loss on the control channel

Delay on the control channel already illustrated clearly the benefit of Umbrella. Here, we study the impact of losses on the control channel on the RTT of ARP packets. Figure 3.15 and 3.16 show the median round trip time (again defined as the time between the ARP request is sent and the ARP response is received), under the ARP-proxy mechanic or the Umbrella architecture.

From Figure 3.16, we conducted more test in the same condition as in Section 3.2.5 Umbrella again displays its strong separation between the control and data plane, with an insensitivity between the packet loss rate on the control channel and the RTT of ARP packets. From Figure 3.15, we observe that the ARP-proxy solution again suffers more and more as the loss rate on the control plane increases. In this case, the ARP-proxy solution suffers because the TCP connection between the OF-enabled switch and the controller is sensitive to packet slow, slowing down the throughput between them, thus increasing notably the response time to the ARP request.

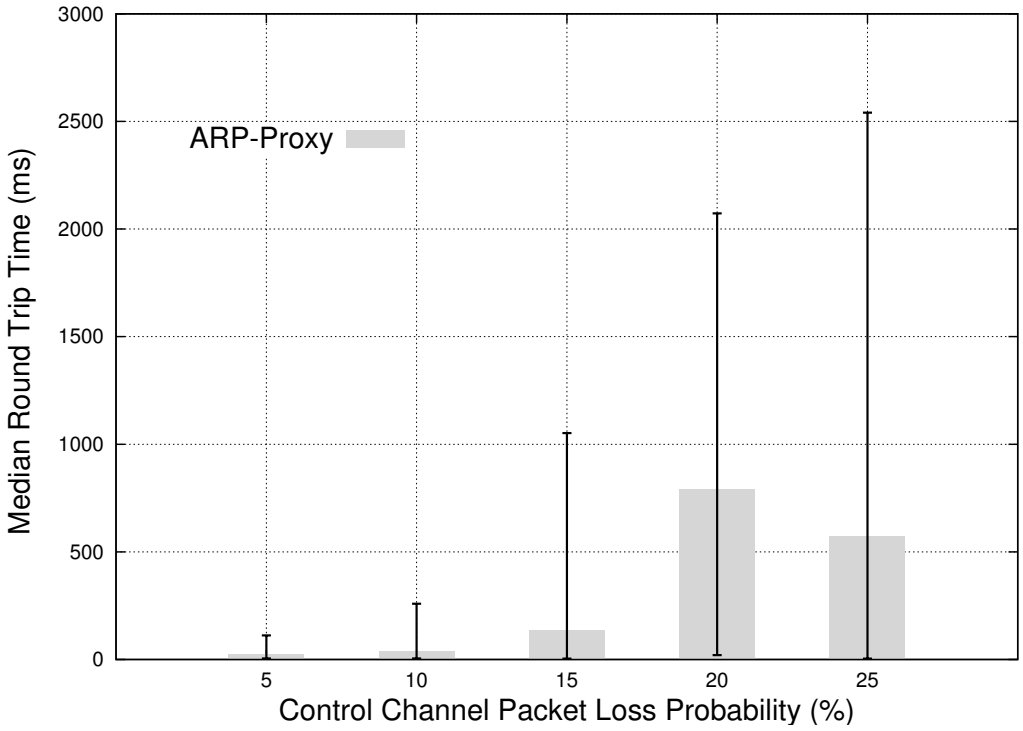

Fig. 3.15: Round trip time of an ARP packet when ARP-Proxy is active, the ARP-cache is empty and the control channel experiences different packet loss rates.

Impact of control plane on data plane throughput

The previous tests demonstrated the fundamental difference in behavior between an ARP-proxy based solution (as current SDN approaches rely on), and our Umbrella approach. In practice however, the behavior of the control channel (in terms of delay especially) will be more complex, and impact on the data plane is much less predictable than presented so far. To attempt to go towards slightly more realistic scenarios, we now touch upon the potential impact of the control plane on data plane throughput inside the IXP fabric. Note that by no means do we claim that our setup is realistic, nor captures the actual complexity of real-world interactions between control plane and data plane performance.

Similarly to the scenario proposed in Figure 3.8, we instantiated in Mininext 100 peering routers and connected them to an Open vSwitch instance. We connected each router to a different virtualized host and set up the switch flow table with all the rules to guarantee data plane connectivity, plus a default rule to redirect the ARP traffic towards the Ryu controller where the proxy ARP application were running. We set the bandwidth of each link to 10Mbps to be able to generate enough traffic on the data plane and observe degradation in data plane throughput. Each hosts also opens an iperf session towards the others routers and we measure the aggregate throughput on the switch.

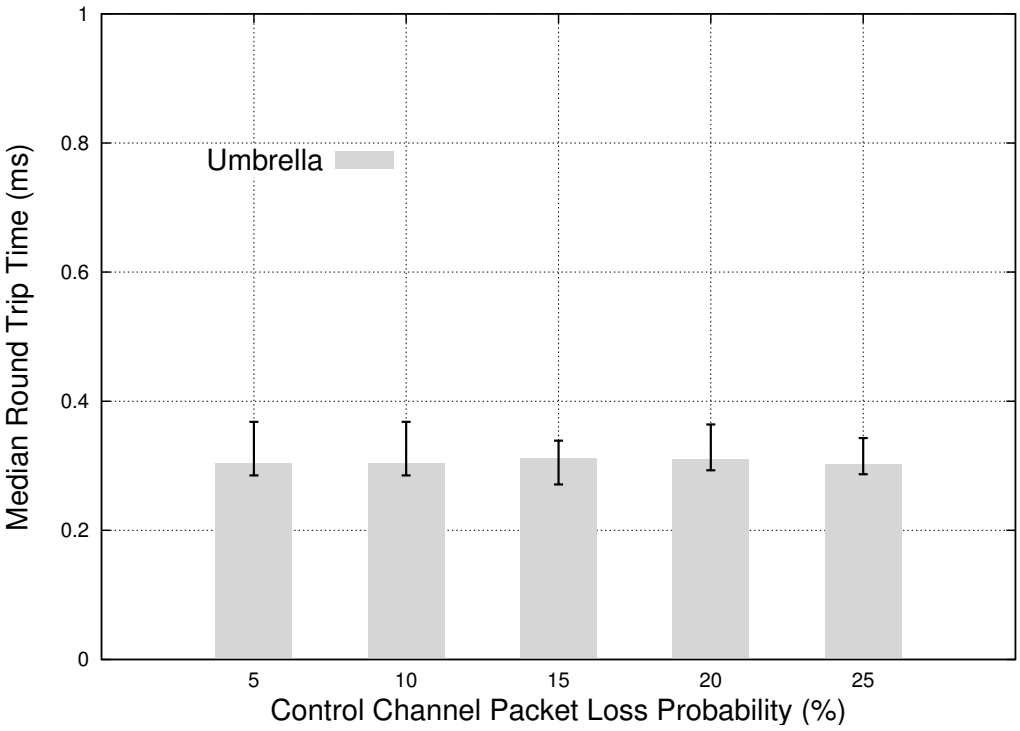

Fig. 3.16: Round trip average time of an ARP packet with Umbrella, the ARP-cache is empty and the control channel experiences different packet loss rates.

We ran our experiment in two different cases: ideal control channel (no delay nor loss) and control channel delayed by 10ms. Figure 3.17 illustrates the loss of data plane throughput due to the control plane delay. The drop in overall data plane throughput after five minutes is related to ARP-cache entries expiration, leading to data plane disruption for some of the hosts. Again, while this scenario is not aimed at making any claim regarding the actual impact of the control plane on the data plane in reality, it illustrates how much a stronger separation between the control and data plane might be beneficial and prevent some issues within the data plane.

3.4.4 Effects of the ARP-induced broadcast storm

We emulated a general IXP topology using Mininext [96]. The test topology created was composed by eight edges switches connected by two cores (Open vSwitch instances – OvS). The edges were attached to Mininext container hosts running Quagga and configured as BGP peering routers. Moreover, a physical route server and a real router (i.e., Cisco c3640), were also connected to the proposed topology through the interfaces of the server running Mininext. This is an important aspect as we want to quantify the impact against a real device, not in an emulated environment. The first test verified the ARP-induced broadcast storm effect on a peering router for a small IXP fabric. To study its impact over BGP connections in a legacy IXP scenario, we used OvSs as traditional layer-2 switches configured with the Spanning

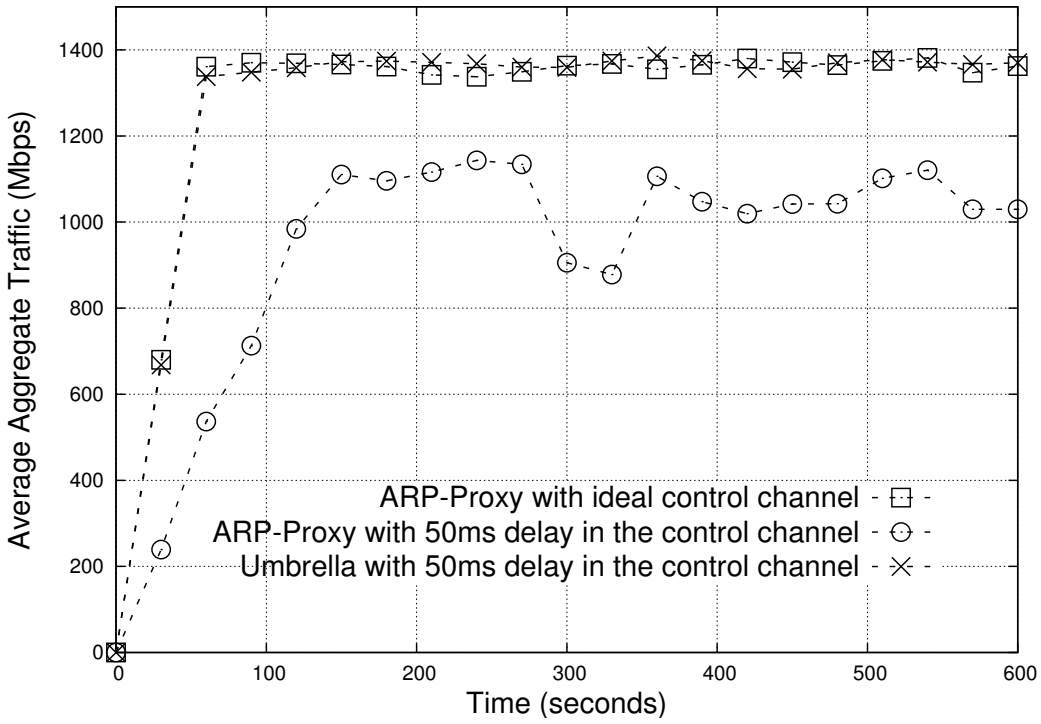

Fig. 3.17: Impact of data plane interfering with control plane traffic.

Tree Protocol (STP). We included a hundred Mininext nodes peering with the route server and the Cisco router. Recent router may behave differen

Figure 3.18 shows the evolution of the average CPU occupancy on the Cisco router before, during and after the broadcast storm. The grey zone represents the time span during which the router looses BGP connectivity with the route server. Before the ARP explosion, the average CPU occupancy is low, showing only a peak at the beginning, when the ARP resolution process starts. When the hosts start to generate ARP messages for non-reachable addresses (after minute 6), the packets reach the Cisco device, bringing its average CPU occupancy close to 100%. Afterwards, the router BGP session with the route server goes down and returns back only when the broadcast storm ends.

Recent routers may behave differently and may not have their BGP sessions flapping. However, their CPU occupancy will be high, and ARP induced storm will waste their internal control plane resources.

Figure 3.19 shows the evolution of the Cisco router BPG state machine during the test. Each block shows the message exchange between the router and the route server given the current BGP state. After minute 3, the router shows an average CPU occupancy higher than 90%, making it unable from processing the route server `KEEPALIVE` messages. As the router BGP hold timer expires, a `NOTIFICATION` message is released and the BGP session goes down. The BGP state becomes

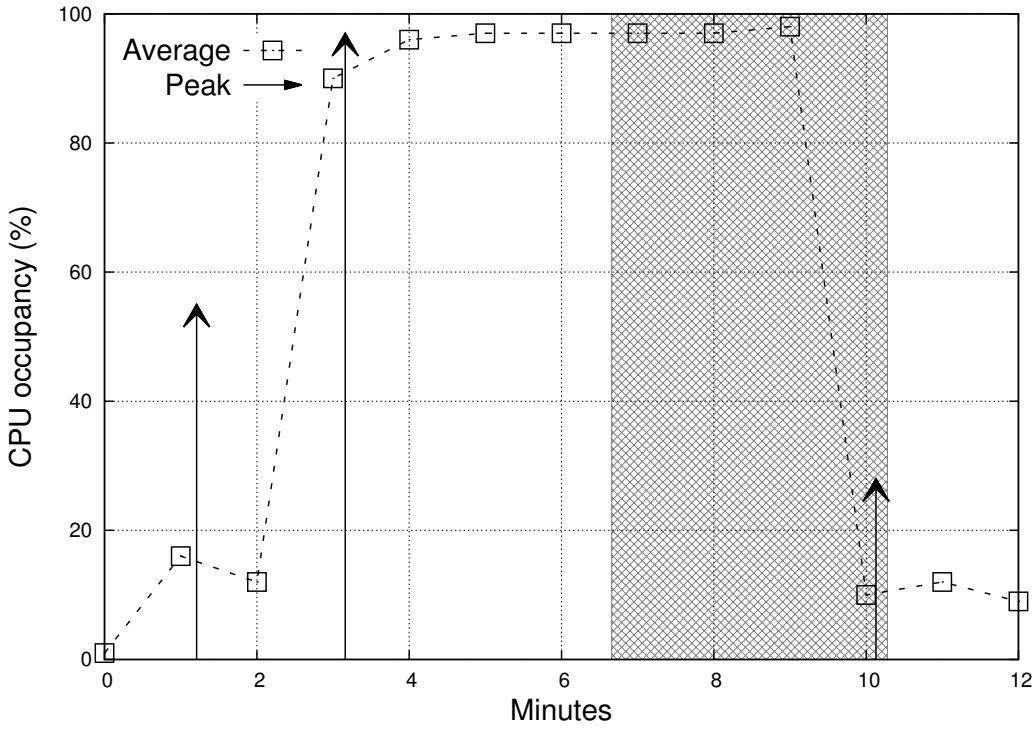

Fig. 3.18: Effect of an ARP-induced broadcast storm on a peering router CPU occupancy.

`Idle` and then `Connecting` as the router tries to re-initiate the TCP connection. Unfortunately, the high CPU load prevents the router from processing the route server SYN-ACK messages (as shown in the Active state block). It is clear the impact of the CPU load on the overall performance of the fabric.

Figure 3.20 compares the average CPU load on the Cisco router between (1) a legacy IP fabric, (2) a legacy fabric with the ARP-sponge server support and (3) an Umbrella enhanced scenario under broadcast storm event. We used a topology with 20 peering routers and configured a server to act as ARP-sponge[10]. The size of the emulated fabric is determined by the resources available in the server running Mininext. The legacy scenario experienced a CPU increase up to 100% on average (leading to a BGP failure), while the one supported by the ARP-sponge shows an average CPU occupancy of 50% with peaks of 90% (the server does not sponge automatically but has a pending value). As this is a small fabric, and ARP-sponge already barely reduces the impact of the storm, it is clear that ARP-sponge cannot keep pace with ARP storms in medium to large fabrics. The Umbrella scenario on the other hand is not affected by the event, keeping the BGP connection alive, regardless of the fabric size.

[10]The ARP-sponge code is available here: http://ams-ix.net/downloads/arpsponge/

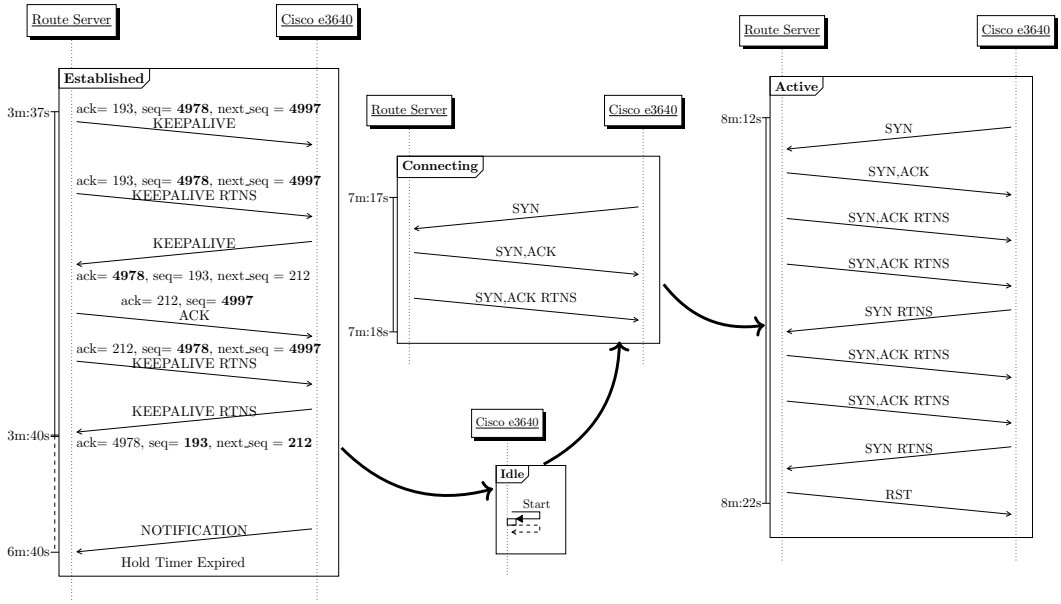

Fig. 3.19: BGP state machine evolution during the ARP-induced broadcast storm.

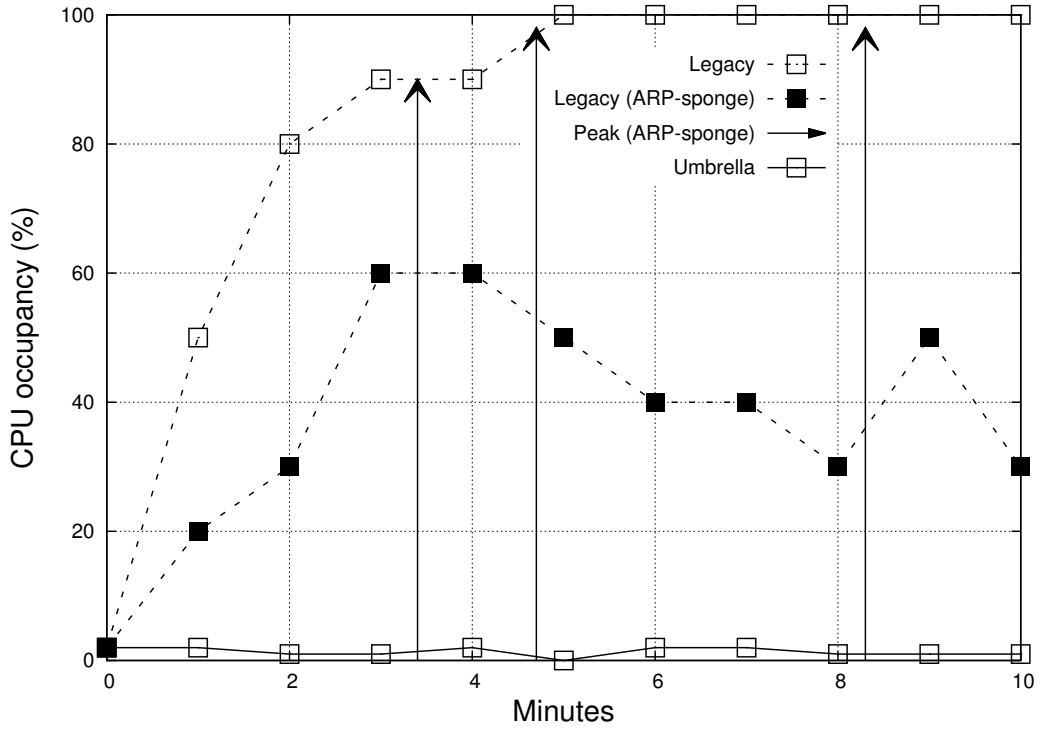

Fig. 3.20: Comparison between the CPU occupancy of a peering router connected to a legacy fabric, a legacy fabric with the ARP-sponge server activated and the Umbrella fabric for a small exchange point.

3.5 Conclusion

This chapter presented a SDN network design that proposed a stronger separation between control and data plane functionalities, leading to enhanced scalability, reliability and manageability. We proposed a new approach to IXP management that leverages SDN programmability to tackle part of the control directly within the data plane, called Umbrella. We built up a practical example of how such a process can be effectively implemented investigating on the impact of control channel disruptions over the data plane. Using our Umbrella approach at the exchange, we illustrated how to tackle broadcast packets directly within the data plane.

The architecture is scalable and can be used in a totally layer-3 (and above) neutral way (as currently done by IXPs), or used in the future in service-oriented IXPs. We see Umbrella as a first step towards SDN architectures less dependent on the control plane, supporting the controller in its role of an intelligent supervisor, rather than an active and dangerously critical decision point.

4 Networking open source hardware testing toolset

> *"In the earliest days, this was a project I worked on with great passion because I wanted to solve the Defense Department's problem: it did not want proprietary networking and it didn't want to be confined to a single network technology."*
>
> **— Vinton Cerf**
> (Internet Founder)

4.1 Introduction

Computer networks are the hallmark of 21st Century society and underpin virtually all infrastructure in the modern world. Consequently, society relies on the correct operation of these networks. To achieve compliant and functional equipment, effort is put into all parts of the network-equipment lifecycle. Testing validates new designs, equipment is tested throughout the production process and new deployments are rigorously tested for compliance and correctness. In addition, many owners of network equipment employ a relentless battery of testing and measurement to ensure the infrastructure operates correctly.

The continuous innovation that is such a desirable property of the Internet has also led to a dilemma for network testing. For a typical piece of new networking equipment there will be a multitude of related IEEE standards and standards-track IETF RFCs, each one requiring test cases to ensure correctness for network-equipment. This has led to a multi-billion dollar industry in network test equipment giving rise to companies such as Ixia, Spirent, Fluke, and Emulex/Endace among others.

However, such equipment has evolved with a number of undesirable characteristics: commonly closed and proprietary systems with limited flexibility well outside the reach of most universities and research laboratories. Even a modest two port 10GbE network tester capable of full line-rate costs upward of $25,000 and adding support for additional protocols, large numbers of TCP streams, and non-trivial traffic profiles quickly increases this price. This has been the case for two reasons. Firstly,

network test equipment capable of full-line rate with high-precision timestamping is a significant engineering challenge, leading to state-of-the-art and specialist physical components. Secondly, test equipment is often developed simultaneously with early prototype network equipment. Thus, modest numbers of units sold mean an expensive and slow time to develop test hardware and software.

This slow development cycle and high expense opens an opportunity for an open-source network tester. It is no longer necessary to build network testers on top of specialized, proprietary hardware. There are multiple open-source hardware platforms with the potential for line-rate across many 10GbE ports, for example, the NetFPGA-10G[1], Xilinx VC709[2] and Terasic DE5-Net[3]. Each of these fully-reprogrammable cards purports being capable of running at line-rate. For example, the NetFPGA-10G has 4×10GbE interfaces, is based on a Xilinx FPGA, and is available to the research and teaching community for less than $2,000 including firmware and software.

We therefore present the Open-Source Network Tester (OSNT[4]), primarily for the research and teaching community. Such a tester needs to be able to achieve full line-rate, provide sufficiently accurate timestamping and be flexible enough to allow new protocol tests to be added to the system. We believe that, as an open-source community grows, a low-cost open-source network tester will also prove valuable to the networking industry. We also envisage the enabling of new testing and validation deployments that are simply financially impractical using commercial testers. Such deployments may see the use of hundreds or thousands of testers, offering previously unobtainable insights and understanding.

In this section we present the proposed architecture for OSNT, describe our first prototype based upon the NetFPGA open-source hardware platform, and present early-day benchmarks illustrating the tester in operation. OSNT is portable across a number of hardware platforms, maximizing reuse and minimizing reimplementation costs as new hardware, physical interfaces and networks become available. By providing an open-source solution we invite everyone from the community to audit (and improve) our implementation as well as adapt it to their needs.

For the rest of this section, we initially motivate the OFLOPS Turbo an testing tool for the Next-Generation OpenFlow switch section 4.7 followed by a detailed presentation of the system design section 4.8 with a set OpenFlow use cases section 4.9, . Furthermore, we present an evaluation of the flow table manipulation capabilities for a representative set of 10 GbE OpenFlow switches section 4.10, discuss related work 4.11 and conclude our work with section 4.12.

[1] http://www.netfpga.org
[2] http://www.xilinx.com/products/boards-and-kits/EK-V7-VC709-CES-G.htm
[3] http://www.de5-net.terasic.com
[4] http://www.osnt.org

4.2 The OSNT Architecture

The OSNT architecture is motivated by limitations in past work: closed-source/proprietary solutions, high costs, lack of flexibility, and the omission of important features such as timestamping and precise packet transmission. Alongside flexibility there is a need for scalability. While our prototype work has focused on single-card solutions, our desire to reproduce real operating conditions means we must have a system that can test beyond single network elements; a production network needs be tested as close as possible to its real operating conditions — this means the OSNT system must also be able to recreate such real operating conditions.

From the outset it has been obvious that flexibility must be a key part of the OSNT approach. This flexibility is needed to accommodate the variety of different uses for OSNT. Four distinct modes of use have become clear.

- *OSNT Traffic Generator*: a single card, capable of generating and receiving packets on four 10GbE interfaces. By incorporating timestamps into each outbound packet, information on end-to-end delay and loss can be computed. Such a system can be used to test a single networking element, *e.g.*, switch or router, or a network encompassed within a sufficiently small area that different inputs and outputs from the network can be connected to the same card.

- *OSNT Traffic Monitor*: a single card, capable of capturing packets arriving through four 10GbE ports, transferring them to the host software for analysis and further processing. Alongside a range of techniques utilized to reduce the bottleneck of the PCIe bandwidth (packet-batching, ring-receivers and pre-allocated host system memory), packets are optionally hashed and truncated in hardware. The card is intended to provide a loss-limited capture system with both high-resolution and high-precision timestamping of events in a live network.

- *Hybrid OSNT system*: our architecture allows the combination of Traffic Generator and Traffic Monitor into a single FPGA device. Using high-precision timestamping of departing and arriving packets, we can perform full line-rate, per-flow characterization of a network (device) under test.

- *Scalable OSNT system* is our approach for coordinating large numbers of multiple traffic generator and traffic monitors synchronized by a common time-base to provide the resources and port-count to test larger network systems. While still largely untested yet, such a coordinated system has been a design objective from the outset.

The OSNT architecture is designed to support these needs for network testing using a scalable architecture that can utilize multiple OSNT cards. Using one or more synchronized OSNT cards, our architecture enables a user to perform measurements throughout the network, characterizing aspects such as end-to-end latency and jitter, packet-loss, congestion events and more.

It is clear that our approach must be capable of full line-rate operation. To this end we built our prototype upon the NetFPGA-10G platform — an open-source hardware platform designed to be capable of full line-rate. We describe our prototype implementation in section 4.5.

While there is a clear need that one or both of the traffic-capture and traffic-generator cores in our OSNT system be present in each use case; these two subsystems have orthogonal design goals: the capture system is intended to provide high-precision inbound timestamping with a loss-limited path that gets (a subset of) captured packets into the host for further processing, whereas the traffic-generator requires precision transmission of packets according to a generator function that may include close-loop control, (*e.g.*, TCP) and even (partial) application protocol implementation.

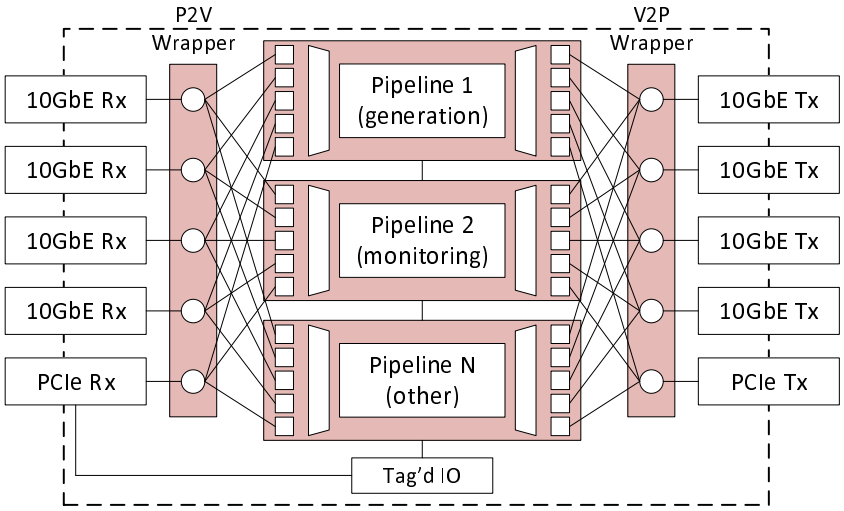

Fig. 4.1: NetV - an approach for NetFPGA Virtualization.

Given that we already had a proven starting design for both generator and capture engines [26, 10], along with a keen desire to reuse component, we were led to develop the *NetV* approach that virtualizes the underlying hardware platform[5]. The approach, shown in Figure 4.1, extends a hardware platform such as the NetFPGA, using *P2V*: Physical to Virtual and *V2P*: Virtual to Physical wrappers. The V2P hardware-wrapper is a per-port arbiter that shares access among each of the 10GbE and PCIe interface-pipelines. This permits multiple NetFPGA pipelines within a single

[5]Our reference prototype is the NetFPGA, but we believe that the architecture including approaches such as *NetV* will be generic across a range of hardware platforms.

FPGA fabric on a single board. In turn providing support for seamless integration of existing pipelines with strong isolation characteristics. For example, a traffic generator can co-exist with a high-precision capture engine. Each pipeline is tagged with a unique ID to ensure register accesses can be distinguished among different pipelines. In this manner, traffic generation and monitoring can be implemented either as standalone units or as a combined system on a single card. Using multiple pipelines in the same design does not affect the overall performances as long as they do not share data structures. The only limitation is given by the available FPGA resources.

Our design has focussed upon one particular architectural approach; this direction was selected to maximize code reuse at the expense of potential redundant gate-logic. Other OSNT architectures may be appropriate but are not explored here for sake of brevity.

4.3 Traffic Generation

The OSNT traffic generator both generates packets and analyzes return statistics. It is designed to generate full line-rate per card interface, and is scalable in a manner that allows for multiple traffic generators to work in parallel within a single OSNT environment. Traffic generation features include:

- support a large number of different traffic flows depending of the uEngine loaded
- flexible packet header specification over multiple headers
- support several standard protocols
- sufficient flexibility to test future protocols
- simulate multiple networking devices/end-systems (e.g. routers running BGP)
- allow timestamping of in and out-bound packets
- allow per-packet traffic-shaping
- statistics gathered per-flow or per flow-aggregate
- support for negative testing through malformed packets

In addition to the above features, OSNT can be customized to support different protocols, numbers of flows and many other features in each given application context.

Figure 4.2 illustrates the high-level architecture of the traffic generation pipeline. The center of the pipeline is a set of micro-engines, each used to support one or more protocols at network and transport-layers such as Ethernet, TCP or UDP and application-protocols such as BGP. Each micro-engine either generates synthetic or replays captured traffic for one or more of the selected egress interfaces. A basic micro-engine is a simple packet replay: a set of pre-defined packets are sent out a given number of times as configured by the software. Each micro-engine contains three building blocks: Traffic Model (TM), Flow Table (FT) and Data Pattern (DP). The Traffic Model contains information about the network characteristics of the generated traffic, such as packets' size and Inter-Packet Delay (IPD). It is a compiled list of these characteristics, extracted by the host software and installed into the hardware. Each parameter is software defined, permitting arbitrary rate distribution patterns: *e.g.*, Constant Bit Rate (CBR) or Poisson distribution. The Flow Table contains a list of header template values used by the micro-engine when generating a packet. Each packet-header is defined by the Flow Table. In this manner, multiple flows with different header characteristics can be generated by a single micro-engine. The micro-engine takes each header-field and manipulates it in one of several ways before setting it: a field may remain constant, incrementally increase, interleave, be set randomly or algorithmically. The number of flows supported by the Flow Table depends on the trade-off between trace complexity and the number of fields to be manipulated. The Data Pattern module sets the payload of a generated packet. The payload can be set to a random pattern, or a pre-specified pattern. A pre-specified pattern allows a user to set the payload of packets to a unique pattern so that the user can execute specific network tests such as continuous-jitter measurement. It also provides in-payload timestamping of departing packets and capabilities for debugging/validating received packets.

Packets generated by the micro-engine are sent to a per-port Arbiter. The arbiter selects among all the packets destined for a port from each micro-engine. Ordering is based upon the required packet departure time. A Delay Module (DM) located after the arbiter will delay packets by each flow's Inter-Packet Delay. A Rate Limiter (RL) guarantees that no flow exceeds the rate assigned to it at each port. Lastly, the packet goes to the (10GbE) MAC, from which it is transmitted to its destination. The Traffic Model (TM) are not be implemented yet.

The traffic generator implementation can also receive incoming packets and provide statistics on them at either port or flow level. This allows the use of the traffic generation subsystem as a standalone unit without an additional external capture

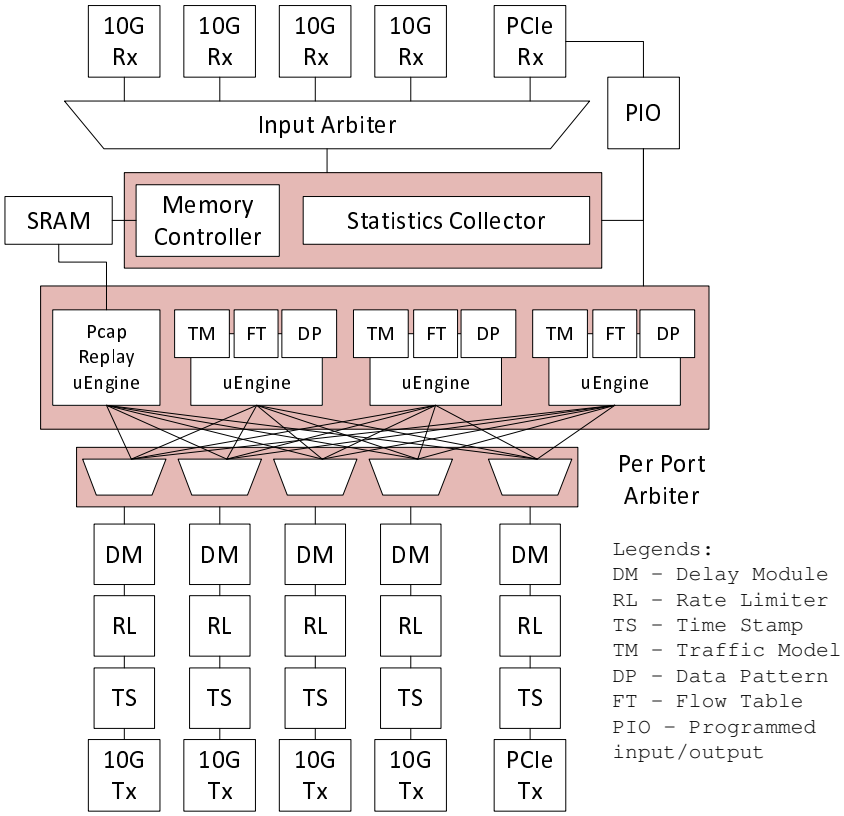

Fig. 4.2: The architecture for OSNT traffic generation system.

subsystem. To this end, packets entering the card through a physical interface are measured, the statistics gathered and the received packets discarded. The gathered statistics are relayed to host software using the programmed input/output (PIO) interface.

The traffic generator has an accurate timestamping mechanism, located just before the transmit 10GbE MAC. The mechanism, identical to the one used in the traffic monitoring unit and described in section 4.4, is used for timing-related measurements of the network, permitting characterization of measurements such as latency and jitter. The timestamp is embedded within the packet at a preconfigured location and can be extracted at the receiver as required.

As for the software side, we provide an extensible GUI to interact with the HW (*e.g.*, load a PCAP trace to replay in HW, define the per-packet inter-departure time, etc.).

4.4 Traffic Monitoring

The OSNT traffic monitor provides four functions:

- packet capture at full line-rate
- packet filtering permitting selection of traffic-of-interest
- high precision, accurate, packet timestamping
- statistics gathering

Figure 4.3 illustrates the architecture of the monitoring pipeline that provides the functionality enumerated above. The 5-tuple (protocol, IP address pair and layer four port pair) extraction is performed using an extensible packet parser able to recognize both VLAN and MPLS headers along with IP in IP encapsulation. Further flexibility is enabled by extending the parser implementation-code as required.

A module positioned immediately after the Physical interfaces and before the receive queues timestamps incoming packets as they are received by hardware. Our design is an architecture that implicitly copes with a workload of full line-rate per port of minimum sized packets. However this will often exceed the capacity of the host-processing, storage, etc., or may contain traffic of no practical interest. To this end we implement two traffic-thinning approaches. The first of these is to utilize the 5-tuple filter implemented in the "Core Monitoring" module. Only packets that are matched to a rule are sent to the software, while all other packets are dropped. The second mechanism is to record a fixed-length part of each packet (sometimes called a *snap-length*) along with a hash of the entire original packet. The challenge here is that if a user is interested in all packets on all interfaces it is possible to exhaust the host resources. We quantify the PCIe bandwidth and the tradeoff for snap-length selection in section 4.5.1.

As for the software side, we provide a python-based GUI that allows the user to interact with the HW components (*e.g.*enable cut/hash, set filtering rules, check statistics). A C-based application that comes with it records the received traffic in both PCAP or PCAPNG format. This allows offline use of common libpcap-based tools (*e.g.*TCPDump, Wireshark.) These tools do not work directly with OSNT: the device driver secures performance by bypassing the Linux TCP/IP stack. We refer the reader to the OSNT website for further information about the software API.

Timestamping

Providing an accurate timestamp to (incoming) packets is a critical objective of the traffic monitoring unit. Packets are timestamped as close to the physical Ethernet device as possible so as to minimize FIFO-generated jitter and permit accurate latency

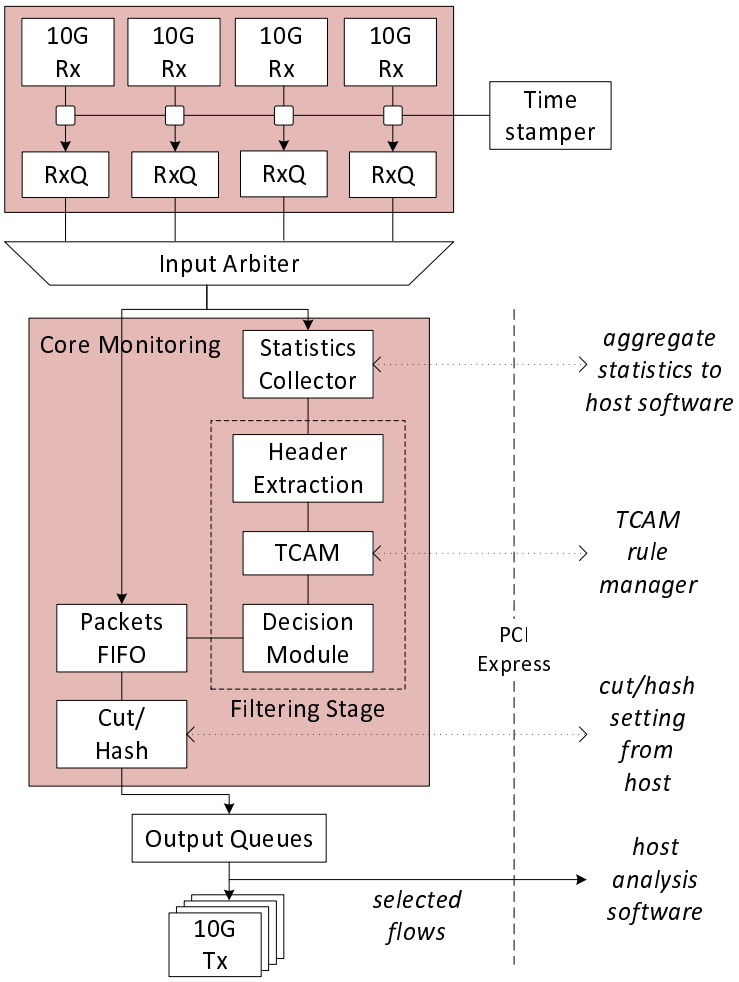

Fig. 4.3: The architecture for OSNT traffic monitoring system.

measurement. A dedicated timestamping unit stamps packets as they arrive from the physical (MAC) interfaces. Each packet is appended with a 64-bit timestamp.

Motivated by the need to have minimal overhead while also providing sufficient resolution and long-term stability, we have chosen to use a 64-bit timestamp divided into two parts, the upper 32-bits count seconds, while the lower 32-bits provide a fraction of a second with a maximum resolution of approximately 233ps; the practical prototype resolution is 6.25ns. Integral to accurate timekeeping is the need to correct the frequency drift of an oscillator. To this end, we use Direct Digital Synthesis (DDS), a technique by which arbitrary variable-frequencies can be generated using synchronous digital logic[95]. The addition of a stable pulse-per-second (PPS) signal such as that derived from a GPS receiver permits both high long-term accuracy and the synchronization of multiple OSNT elements. The selection of a timestamp with this precision was a conscious effort on our part to ensure the abilities of the OSNT design are at least as good as the currently available commercial offerings.

4.5 OSNT NetFPGA-10G Prototype

Our prototype implementation of the OSNT platform has been on the NetFPGA-10G open-source hardware platform. The NetFPGA system provides an ideal rapid prototyping target for the work of OSNT. Since its original inception as an open-source high speed networking platform for the research and education community [59] and, through its second-generation [15], the NetFPGA has proven to be an easy-to-use platform. The NetFPGA project supplies users with both basic infrastructure and a number of pre-worked open-source designs intended to dramatically simplify a users' design experience.

The NetFPGA-10G card, as shown in Figure 4.4, is a 4 port 10GbE PCIe adapter card incorporating a large FPGA fabric. At the core of the board is a Xilinx Virtex-5 FPGA: XC5VTX240T-2 device. Additionally, there are five peripheral subsystems that complement the FPGA: four 10Gbps SFP+ Ethernet interfaces, a Gen1 PCIe subsystem provides the host-bus adapter interface, and memory consists of a combination of both SRAM and DRAM devices. The memories were selected to provide minimal latency and maximal bandwidth over the available FPGA I/Os. The fourth and fifth subsystems are expansion interfaces and the configuration subsystem. The board is implemented as a three-quarter length PCIe adapter, but can also operate as a standalone unit outside the server environment.

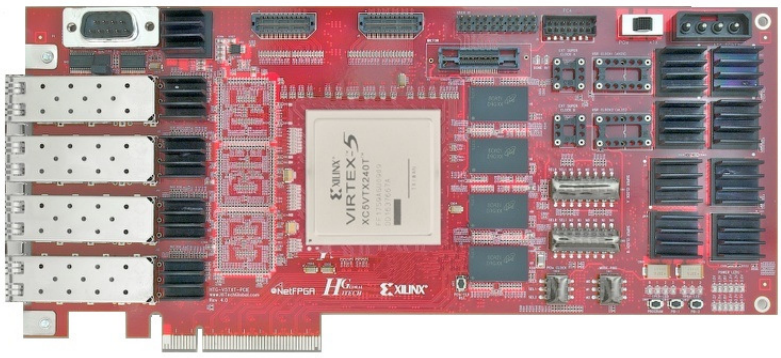

Fig. 4.4: The NetFPGA-10G board.

4.5.1 Experiences with our prototype

By building our prototype on the NetFPGA-10G platform we have inherited several platform constraints. Despite having a large FPGA device, design decisions must trade resources. One example of this is in the sizing of TCAM tables for filtering. Table size is traded directly against overall design size. In our prototype implementation, the tuple-based filtering tables is limited to 16 entries.

While the internal NetFPGA datapath has been designed to accommodate full line-rate, minimum-sized packets, the PCIe interface lacks the bandwidth to transmit all traffic to or from the host. The NetFPGA-10G provides a first generation, 8-lane PCIe implementation. This interface uses an MTU of 128 bytes and without careful packing a naïve implementation of DMA and device driver may achieve as low as 33.5% utilization (for transactions of 129 byte packets). Furthermore, even for an ideal scenario this interface imposes a limit of around 13.1 Mpps for an MTU of 128 bytes or a little over 15 Gb/s. It is clear that capture-to-host of all four interfaces when operating at 10Gb/s into the host is not practical. Alongside flow-filtering the traffic-thinning technique of selecting a snap-length places a known limit on the maximum amount of data that needs to be transferred over the PCIe to the host.

The option to add a hash of the original packet, along with a fixed snap-length, means that we can reduce the potential number of bytes per packet to a known upper boundary. Although the hash adds an overhead of 128 bits per packet, it permits practical packet identification which in turn means we can perform end-to-end latency measurements as well as identifying specific loss-events. The ability to do bandwidth limiting in this way allows us to achieve a maximum rate of approximately 21.7 Mpps provided we use non-naïve DMA and device-driver mechanisms.

Fortunately, there has been considerable progress in non-naïve DMA and device-driver mechanisms to reduce the bottleneck of PCIe bandwidth; packet-batching, ring-receivers and pre-allocated host system memory have all seen use in past dedicated capture systems [65]. Recent efforts such as netmap achieve rates of 14.8 Mpps into user-space for single port commodity 10GbE interface cards. Our architecture is not limited to a current hardware-implementation; the OSNT system when running on more advanced hardware such as the Xilinx VC709, using the third generation PCIe, has sufficient bandwidth to support full size payloads for all four 10GbE ports. In fact, the open-source nature of OSNT means that having this system operate effectively on any future NetFPGA platform, other platforms from Xilinx or indeed from other FPGA vendors is no more complicated than the porting of any open-source project.

Figure 4.5 shows the capture engine performance results. The system has been validated for one and two ports against 100% line utilization (packets sent back-to-back) across a range of packet sizes. The IXIA . In the first case, OSNT is able to record all received traffic, without loss, independently of packet length. Additionally, using two ports at the same time, the system is able to record traffic without experiencing any kind of loss up to 14 Gbps (PCIe Gen1 limitation); the impact of the cut/hash feature at reducing traffic across the PCIe is clear.

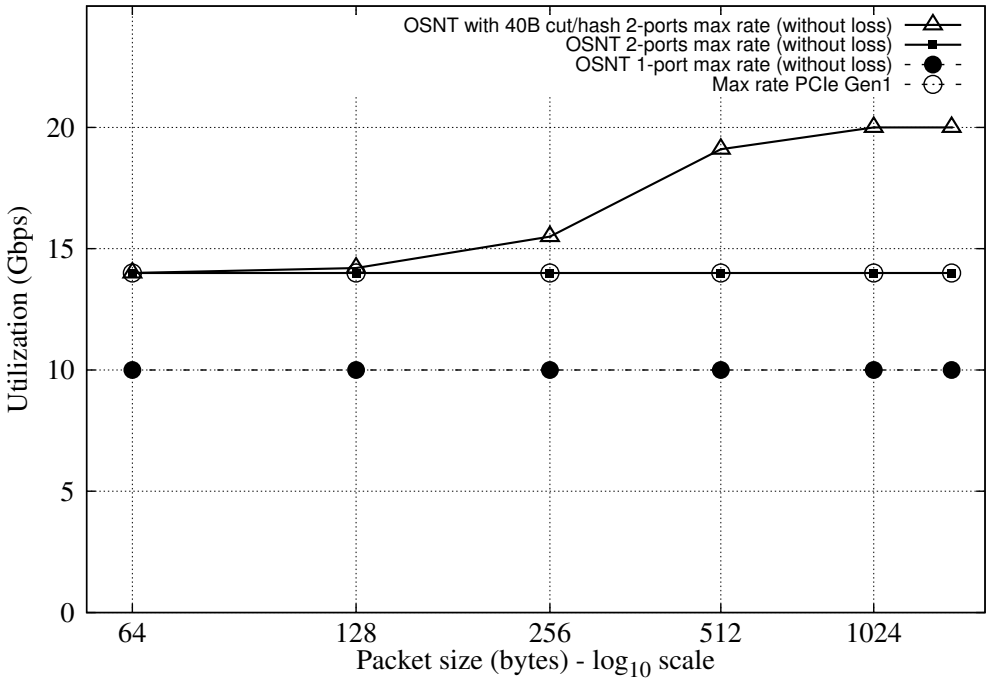

Fig. 4.5: The OSNT per-packet capture engine performance for various presented traffic loads.

We validated the OSNT performance against the IXIA 400T and similtaneously confirmed these results via a parallel capture using optical-port splitters to an Emulex EndaceDAG 9.2, each equipped with 2x10G ports. IXIA provides the capability of both generating full line rate traffic and full line rate monitoring; permitting validation of both capture and generation capabilities. The Endace DAG provides full line rate capture and high-precision time-stamping and offers a further confirmation mechanism.

Testing of the traffic-generator we were able to confirm to our satisfaction that the OSNT Traffic Generator is able to generate full line rate over two ports independently of the packet length. Tests were conducted over a range of packet-sizes with results compared directly against IXIA-based generators. In all experiments data was generated (and measured) on all four NetFPGA ports with a combination of IXIA and Endace packet-capture and measurement.

4.6 ONST Conclusion

In this section we introduced OSNT, an open source network tester. We described the OSNT architecture which permits a flexible combination of multiple packet-processing pipelines using a new virtualization technique, NetV. While the NetV virtualization approach was designed with the NetFPGA in mind, this technique is not

bound to that hardware and should be able to provide flexibility and versatility across a range of uses. Using the NetV approach we showed how the OSNT system can implement both traffic-generator and network monitor functions. We also described our prototype implementation using the rapid-prototyping NetFPGA platform and characterized aspects of that implementation.

The OSNT platform provides a network tester that is able to combine desirable software flexibility with the advantages of being built upon an open-source hardware platform. The versatility of OSNT is in its suitability for a range of applications, from the testing of single items of networking equipment to the characterizing of large distributed networks.

The OSNT system is available to the research community through the NetFPGA project. Any user who owns a NetFPGA card can simply use it, with no additional hardware expense. We envisage the project being extended and enhanced by the research community, and users are encouraged to contribute further features and capabilities, as well as to share their own experience using OSNT.

The promise of OSNT is an exciting one. In the field of network measurement alone, high-precision, loss-limited capture has led to remarkable progress in the characterization and understanding of the modern Internet. The OSNT traffic monitor overcomes the two biggest issue with these capture deployments to date — the cost and lack of flexibility — while also, by virtuel of being open-source, providing an auditable test system that encourages repeatability in network science.

4.7 Introduction

Research on SDN technologies and primarily its predominant realisation, the OpenFlow protocol, has developed a wide range of applications to improve network functionality. OpenFlow control applications can improve network management, monitoring and performance, while being backwards-compatible with data plane protocols and end-host network stacks. As a result, within only a few years since the definition of the first version of the protocol, many vendors have introduced production-level support in an effort to transfer innovative research output to the market.

Nonetheless, such continuous network innovation introduces a dilemma for network testing, as relevant evaluation platforms remain closed and proprietary and provide limited flexibility. To achieve compliant and functional equipment, effort must be put into all parts of the network-equipment life-cycle, from design to production. The problem of network testing is further augmented in the OpenFlow protocol context. The protocol philosophy introduces new performance challenges in network

device design, dissimilar to the challenges of traditional network equipment, and sets testing flexibility as a primary requirement. OpenFlow introduces a reduction in the network control timescales which closely approximate flow control timescales. As a result, OpenFlow switch implementation influences the control architecture of the network and its overall performance. A switch which provides poor performance in the support of a protocol functionality can become a bottleneck for architectures that rely heavily on the specific functionality; *e.g.*high latency in `flow_stats_reply` functionality of a switch can be critical for traffic monitoring control applications. Protocol support variability is equally critical for the consistency of the control plane. Most switches do not provide any guarantees on the installation order and latency of a sequence of flow table updates; as a result, such updates need to be carefully sequenced to not violate policy [37], effectively increasing the insertion latency.

The OpenFlow protocol, currently defining its 1.5 version, has greatly transformed its design over the years, reflecting the deployment experience and requirements of a constantly widening range of network environments. As a result, the OpenFlow community *requires* a performance testing platform capable to co-evolve with the protocol and to support rapid prototyping of experimentation scenarios which highlight the impact of new protocols' features. Furthermore, the increase in link capacity augments the precision requirement for meaningful packet-level measurements. For example, 10 GbE links have become the de-facto solution for the aggregation layer of modern datacenter networks, a first-class citizen of the SDN ecosystem, and require high measurement precision, on the order of sub-μsec, for certain network application classes. An estimation error of 100μsec in the policy enforcement of a security application may translate to unauthorized transmission of multiple KBs of sensitive information. This paper claims that in order to fully exploit OpenFlow protocol capabilities in production environments, we require a flexible and high–precision open–source measurement platform. The openness and flexibility is important in order to establish an evolvable community–based tool.

This section presents an effort to enhance the measurement capabilities of the OFLOPS [90] switch evaluation framework with support for the emerging protocol requirement. Specifically, we present the OFLOPS integration with the NetFPGA-10G platform[6] and the Open Source Network Tester (OSNT)[7] platform [9]. OSNT enhances OFLOPS with sub-μsec precision and 20 Gbps full bi-directional traffic generation and capturing (when traffic thinning techniques are being used), providing a highly flexible and open platform for OpenFlow experimentation at high data rates.

[6] `http://www.netfpga.org`
[7] `http://www.osnt.org`

4.8 OFLOPS Turbo design

A schematic of the design of the OFLOPS Turbo design is depicted in Figure 4.6. The OFLOPS Turbo architecture consists of a software and a hardware subsystem. The software subsystem runs the core OFLOPS functionality, along with the measurement module of the user.

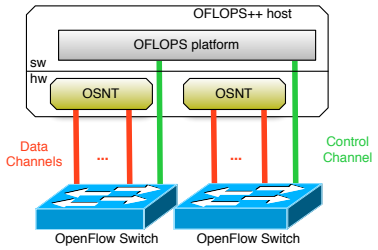

Fig. 4.6: OFLOPS++ design

A measurement module contains both the control and data plane functionality of the experiment. The hardware subsystem consists of a series of netFPGA10G cards running the OSNT design and is responsible to fulfil the data plane requirements of the experiment. The user can interconnect the OFLOPS Turbo host with one or more switches in arbitrary topologies and measure with high precision specific aspects of the network architecture, both on the data and control plane. For the rest of the section, we describe the design of the OSNT hardware design and the OFLOPS software architecture.

4.8.1 OSNT: Open Source Network Tester

The OSNT is a fully open–source traffic generation and capturing system fully described in section 4.1. Its architecture is motivated by limitations in existing network testing solutions: closed-source/proprietary, high costs, inflexibility, and lack of important features such as high-precision timestamping and packet transmission. Primarily designed for the research and teaching community, its key design goals were low cost, high-precision time-stamping and packet transmission, as well as, scalability. In addition, the open–source nature of the system gives the flexibility to allow new protocol tests to be added to the system. The prototype implementation builds upon the netFPGA10G platform – an open-source hardware platform designed to support full line-rate programmable packet forwarding. The combination of Traffic Generator and Traffic Monitor into a single FPGA-equipped device allows a per–flow characterization of a networking system (*i.e.*, OpenFlow-enabled switch) within a single card. Using one or more synchronized OSNT cards, the architecture enables a user to perform measurements throughout the network, characterizing aspects such as end-to-end latency and jitter, packet-loss, congestion events and more.

The OSNT *Traffic Capture* subsystem is intended to provide high–precision inbound timestamping with a loss-limited path that gets (a subset of) captured packets into the host for further processing. The design associates packets with a 64-bit timestamp on receipt by the MAC module, thus minimizing queueing noise. The timestamp resolution is 6.25nsec with clock drift and phase coordination maintained by a GPS input. The OSNT *Traffic Generation* subsystem consists of several micro-engines, each of which generates traffic according to a given generator function. The traffic generator has an accurate timestamping mechanism, located just before the transmit 10GbE MAC. The mechanism, identical to the one used in the traffic monitoring unit, is used for timing-related measurements of the network, permitting characterization of measurements such as latency and jitter. When enabled, the timestamp is embedded within the packet at a preconfigured location and can be extracted at the receiver as required.

4.8.2 OFLOPS: Open Framework for OpenFlow Switch Evaluation

Measuring OpenFlow switch implementations is a challenging task in terms of characterization accuracy and precision. Predominantly, production-level OpenFlow-switch devices provide proprietary OpenFlow drivers with minimum logging capabilities. Characterising the performance of an OpenFlow-switch requires a black box approach to the problem, where multiple input measurements must be monitored concurrently in order to characterise the behaviour of the switch.

OFLOPS is an holistic measurement platform which enables the development of custom OpenFlow-based experiments. The platform provides a unified API that allows developers to control and receive information from the data and control channels, as well as, SNMP switch-state information. Experimenters can develop measurement modules on top of OFLOPS, implementing custom OpenFlow applications and measure their performance through the data plane.

Currently OFLOPS provides a wide range of elementary testing modules evaluating the performance of flow actions, flow table management, flow counter extraction, OpenFlow-based packet injection and capturing and detecting potential performance penalties due to OpenFlow operation co-interaction. OFLOPS uses OSNT primarily to control the data plane of the network using a simple interface, provided by the OSNT user-space library. The re-desgined DMA engine of the netFPGA10G platform improves OFLOPS user-space capturing and transmission capabilities, compared to the earlier OFLOPS hardware design for the NetFPGA1G platform.

4.9 Use Cases

Latency matter. Some applications are more affected by latency rather than throughput (*i.e.*, voice, networked games, interactive sessions). The increasing need for low latency systems brought also the need of new high performance hardware based measurement platforms able to sustain high rates and to mantain high accurancy as testing and troubleshooting systems. OFLOPS Turbo have been designed to insure such an accuracy on four 10Gbps port unifying testing for switching latency, performance switching, OpenFlow actions set maximum table entry, OpenFlow actions latency and any combination of those. We have been working with OFLOPS Turbo for those deferent usage. We propose in the following some other use cases.

4.9.1 White-Box dilemma

Nowadays more and more white-box ethernet switches support OpenFlow protocol. White box networking refers to the ability to use generic, off-the-shelf switches and routers within the forwarding plane of a software-defined network. White-box network devices are open to different Operating Systems along with their OF-agent and applications. The majority of white box networking opertaing systems are linux-based because of the many open and free tools available. Different implementation of the same standard can bring to different performances. For example, different white box switch vendors are using the same chipset (most of 48x10Gbps + 4x40Gbps 1RU switch are using the Trident chipset from Broadcom) but with a different design (CPUs, memories, implementation). OFLOPS Turbo can compare performances of different OF-agent software and OS specific limitations or compare their hardware specific implementation.

OFLOPS++ is able to do high latency precision elementary OpenFlow testing to help identifying the possible limitations in hardware or in software. When comparing various combinations of white box and OS with OF-agent we can easily identify the best hardware and software.

4.9.2 The ALIEN HAL testing phase

The ALIEN [8] Hardware Abstraction Layer[30] is designed to support OpenFlow protocol implementation regardless of protocol's version on non-compatible OpenFlow network devices. This abstraction layer supports various hardware platforms in terms of data plane architecture and closed control plane protocols. OFLOPS Turbo might

[8]The ALIEN project is a FP7 project with the grant agreement number 317880

be used during the ALIEN HAL testing phase of the project for the most important requirements who are on the OpenFlow implementation by each partner.

The ALIEN HAL have been implemented on different hardware platforms: DOCSIS EZ-CHIP, OCTEON, BROADCOM, x86, GEPON, NetFPGA, ATCA with OCTEON, ADVA DWDM. The OFLOPS Turbo high latency precision measurement testing will not be requireed for all the platform but the OpenFlow minimum properties testing will be necessary to prove they all pass the ALIEN HAL implementation minimum requirement. With OFLOPS Turbo the full benchmark testing could be conduct on the high performance platforms testing (x86, Broadcom, ACTA with Octeon, the Dell Split-Data-Plane with Octeon, NetFPGA and EZ-CHIP). There is no minimum performance requirement in the ALIEN project but the performance benchmarking will definitively help for positioning the platform or push the developers to improve their implementation.

4.9.3 IXP: Internet eXchange Point

An Internet eXchange Point is a physical point where Internet traffic can be exchanged and today they are in rebirth specially with SDN/OpenFlow [32]. The IXP who wants to invest in a new infrastructure based on SDN/OpenFlow switches need to select the ultimate performer switch. For facilitating the migration different vendors are proposing switches with hybrid capacities who are able to continue to run the legacies features on a part of ethernet interfaces and running in open flow mode on the rest. But hybrid switches has some specific limitation due to the CAM configuration to be able to mix the OpenFlow tables with the FIBs and ACL legacy tables. Based on the specifications of the new SDN IXP architecture OFLOPS Turbo is a great help to accelerate the benchmark scenarios.

4.10 Performance Evaluation

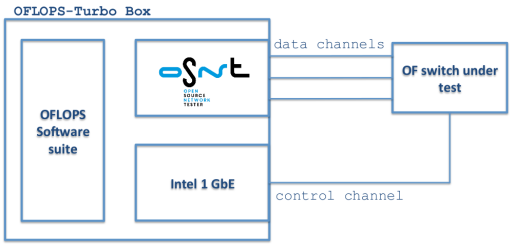

Fig. 4.7: The proposed testbed.

In this section we use OFLOPS to evaluate two representative 10 GbE switches: the *Pica8 P3922*; and the *Dell Force10 S4810*. The Pica8 P3922 switch [82] is a 48 ports

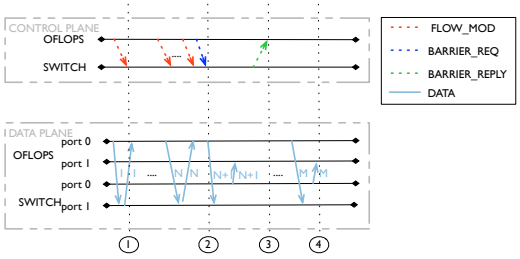

Fig. 4.8: Flow insertion measurement.

10 GbE SFP+ and 4 ports 40 GbE QSPF+ port switch. The switch provides two operational modes: *L2/L3* and *OVS* mode. L2/L3 mode provides hybrid OpenFlow support along with other standardised data link and network control protocols, like OSPF and RIP. OVS mode functions as a standalone OPen VSwitch switch with a forwarding table mapped to the TCAM table of the switch using a custom device driver. Vendor specifications do not define the maximum flow table size supported by the silicon, but we have successfully tested fast-path data plane support for up to 1000 unique flows in OVS mode. The Pica8 L2/L3 mode allows users to define the flow table size through the switch user interface and can be configured to support up to 500 flows. The Dell Force10 S4810 [28] switch is a 48 10 GbE QSFP+ ports. Its firmware supports hybrid OpenFlow functionality and exposes three hardware tables: a 512 wildcard entry table supporting full OpenFlow tuple matching, a 20000 entry table supporting only destination MAC address matching and a 6000 entry table support IP source and destination matching. Both switches use a similar StrataSGX trident silicon version [41]. The two switches reflect two popular approaches in OpenFlow support: mixing protocol functionality with legacy protocols; or developing a single-purpose OpenFlow-optimised firmware.

We setup **OFLOPS-Turbo!** in a dual socket 8-core Intel Xeon E5 server with 128 Gb RAM memory, equipped with a NetFPGA 10G card running the OSNT design and an Intel 1 GbE card for the control channel. Three ports of the NetFPGA card are connected directly to the measured switch, while the switch control channel uses the switch management port and connects directly to the 1 GbE card, similarly to the topology in Figure 4.7. During experiments we connect the NetFPGA card with a GPS receiver, to correct its time reference and provide nano-second measurement precision. In addition, in order to establish a common time reference between the control and data plane traffic of the measurement, the control channel is replicated to the fourth port of the NetFPGA card through an optical wiretap with negligible processing delay.

4.11 Related Work

Network testers, and open-source network testers are not new; uniquely, OSNT brings the incorporation of designs that operate intimately with the hardware. Our efforts ride the established tradition of network measurement and testing that exists in the network research and academic communities.

A small sample of open-source and community projects include: Iperf [47] and later Netperf [68], developed to provide performance tests of throughput and end-to-end latency. Traffic loads from previously captured *pcap* files could be transmitted using Tcpreplay [100]. Netalyzer [54] uses bespoke server and client infrastructure to measure many aspects of Internet performance and behaviour. Swing [102] provided a closed-loop traffic generator: first monitoring and characterizing, and then regenerating system load replicating the measured characteristics. Early attempts at both flexible and feature-rich traffic generation led to the Ostinato [99] traffic generator. The netmap [88] achieves near-optimal host throughput but is still restricted by the underlying hardware for timestamps, traffic-shaping and maximum-rate capacity. A final example, Bonelli *et al.* [16] describe a near-line-rate traffic on a 10Gbps link that uses multi-core multi-queue commodity hardware, albeit without the flexibility or guarantee of full line-rate throughput, precise traffic replay timing and sufficient packet capture timestamp accuracy and precision.

Commercial network testers are provided by a number of companies: Ixia and Spirent dominate, but other test equipment manufacturers also have network-test offerings. Despite their ability to perform at high line-rate, a criticism common to all these systems is the cost and inflexibility. Supporting newly-designed protocols is often expensive while supporting newly-designed physical line standard can result in an entirely new system.

In the measurement community the ubiquitous *pcap* program, *tcpdump*, has been the tool of choice for network capture. However, capture-system performance (and rates of loss) are dictated by the underlying host: a combination of hardware, operating-system, device-drivers and software. Additionally, it is rare for these software systems to provide any common clock across the captures, making end-to-end latency measurements complicated and inaccurate. There have been software/hardware efforts in the past that incorporate GPS-coordinated high-precision hardware timestamps and use device-driver designs intended to mitigate loss under load [65]. However, this work was limited to 1GbE and serves now only to provide a motivating example. NTP is a mature time synchronization method; however, it can only achieve an accuracy better than 1ms under limited conditions [101]; making it unsuitable for high precision traffic characterization.

In contrast to the large range of commercial offerings available to generate traffic; the high-precision capture market has few commercial systems and is dominated by the Endace DAG card.

Several previous NetFPGA-based projects using the previous generation NetFPGA 4×1GbE platform have also provided traffic-generation [26] and traffic-monitoring [10]. The architecture of OSNT has been heavily informed by the designs, limitations and experience with these systems.

4.12 OFLOPS Turbo Conclusions

In this last section we have presented OFLOPS Turbo, an open and flexible OpenFlow testing framework for the next-generation of OpenFlow switches. IXPs can take benefit of OFLOPS Turbo and OSNT for their day to day testing and monitoring. OFLOPS Turbo takes advantage of the OSNT hardware design for the netFPGA10G platform and provides support for 10GbE traffic generation and capture, coupled with a high precision timestamping functionality. We believe that OFLOPS Turbo will provide the tool for the OpenFlow community to better understand the performance impact of OpenFlow implementations and motivate a community of rigorous OpenFlow testing.

5 Real deployment

> *Innovation distinguishes between a leader and a follower.*
>
> — **Steve Jobs**
> (CEO Apple Inc.)

5.1 Introduction

This chapter reports the work that has been done to migrate the TouIX IXP located in Toulouse, France from a traditional to a full OpenFlow IXP using Umbrella architecture. TouIX is a non-profit neutral Internet eXchange Point organization founded in 2005. It provides an interconnected network infrastructure at 4 PoPs around the city of Toulouse, and is interconnected to the Paris FranceIX and LyonIX IXPs. FranceIX shares is bridge broadcast domain directly through a layer 2 link to TouIX on a dedicated VLAN. LyonIX shares is prefixes through its route server how routes all traffic between the two IXPs. TouIX was renamed TouSIX (Toulouse Software Internet eXchange) in late 2014 and, to the best of our knowledge, from May 2015 is the first Internet Exchange Provider in Europe to fully leverage OpenFlow for its day-to-day operations.

The remainder of the chapter is as follows: section 5.2 gives technical details on how the TouIX IXP was built, and lists its related technical and network services issues. Section 5.3 describes the installation and deployment of the OpenFlow equipments. It especially motivates the selection of Pica8 switches, explains how technical issues have been solved and presents preliminary results. Section 5.4 then presents the new software management tools that have been designed for TouSIX. Section 5.5 exhibits a significant set of measurements and evaluation performed on the new TouSIX IXP, exhibiting the benefits of the migration to an SDN based IXP. Section concluding the chapter in section 7.

5.2 TouIX: Legacy architecture and raised issues

Figure 5.1 shows the topology of TouIX. The primary and first site to be located was *Cogent*, where the BGP route server was installed. The Cisco equipment[1] being used in the fabric was configured with three different VLAN tags: admin[2], TouIX_VLAN[3] and France_IX_VLAN[4]. As the Cisco equipment being used was getting older and not appropriate for the fabric anymore, the idea of shaping a completely new Internet exchange to ease the management came to mind.

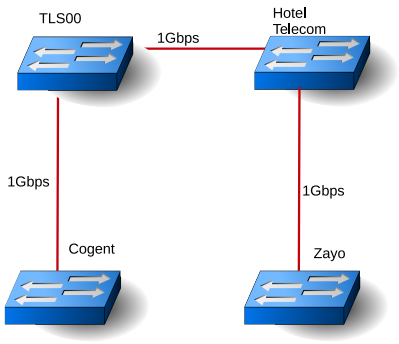

Fig. 5.1: TouIX legacy architecture

The overall management operations of the TouIX IXP fabric represent a complex task that is not easy to perform with existing tools. It was especially hard to analyse and fix several issues. There was for instance the case of several broadcast storms experienced by the TouIX fabric. The lack of an appropriate monitoring infrastructure as well as an irregular appearance of these storm phenomenons made their root causes very hard to identify. The Cisco switches being used did not provide appropriate filtering features to restrict these kinds of undesired traffic that might harm the fabric. There was also a strong requirement for improving the protection of the peering routers against undesired traffic, as well as improving the stability and manageability of the whole infrastructure to prevent the aforementioned phenomenons. Furthermore, the TouIX architecture is suffering from a lack of redundancy of spare equipments.

These various operations issues together with the huge management complexity have motivated the change from the legacy TouIX to TouSIX, in particular :

[1] TouIX used Cisco WS-C3550-12G (TLS00), WS-C3560G (Cogent), WS-C2960G-24TC-L (Zayo) and WS-C2960G-24TC-L (Hotel Telecom)
[2] Used for admin operations on the TouIX network
[3] TouIX DataPath
[4] FranceIX is the larger IXP in France and TouIX allows its members to access France-IX

- Incorrect member configurations can create loops and broadcast storm [34], or expose information through discovery protocol like Link Local Discovery Protocol LLDP or Cisco Discovery Protocol CDP. Incorrect IXP switch configurations can also create service disruption.

- The service management complexity forces administration staff members to be familiar with legacy equipment of specific vendors, generally working on a vendor specific Command Line Interface (CLI). A simple web based application will help to simplify all maintenance and management operations.

- TouIX is lacking from a monitoring system able to mesure the amount of traffic exchanged between members, and to identify types of traffic, for instance.

These operational issues then drove the desire of a more generic and easily programmable IXP fabric. TouSI aims at introducing an OpenFlow based IXP fabric managed through a web application, and with a fine grained monitoring, for better controlling the network, and more easily managing and operating it.

5.3 TouSIX architecture

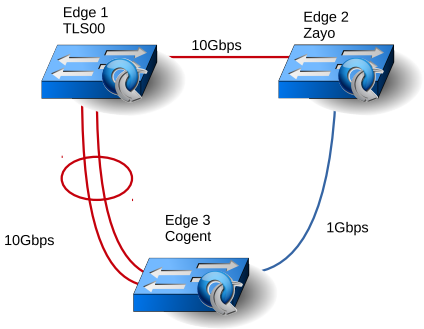

Fig. 5.2: TouSIX architecture

Figure 5.2 exhibits our final choice for the TouSIX topology.

This new network covers three points of presence, all linked directly to each other. The link between TLS00 and Cogent is an aggregated link. The main purpose of this type of interconnection is to increase the availability between these two switches. This functionality is entirely managed by the virtual switch via LACP, making it invisible for the controller[5]. One link (Cogent-Zayo) is set as a passive

[5]It is made possible by creating a logical port for the OpenFlow agent of the switch.

link, for privileging the bandwidth usage of the two other links, and then being able to benefit from some redundancy in case of link failure, or simply to switch path of data for adapting traffic engineering policies to traffic evolutions, for instance. This switching capability is implemented using the Fast-Failover capability of the OpenFlow protocol.

For TouSIX, we selected the Pica8 switches to support our architecture. The main motivation for these equipments is the implementation of Open vSwitch [74] on PicOS. It allows us to handle an open-source solution for switching, and facilitate the simulation of the topology in a virtual environment. In the OpenFlow architecture, it is also required for managing the OpenFlow agents of the switches, to install a controller. It has been added on top of the BGP route server, as it is the only available server in our new TouSIX architecture. At this stage of the TouSIX development, the OpenFlow controller is using for communicating with all OpenFlow agents a dedicated VLAN of the ancient (but still running) TouIX architecture. The Ryu controller [93] has been selected for both our OpenFlow rules tests, as well as for the day to day operations.

The key element for migrating to an SDN based IXP relates to the definition of the OpenFlow rules. In our OpenFlow installation, we can divide the forwarding rules in two classes. The first one is related to the data plane, and to the forwarding of traffic at layer 2, as we made the choice of having a full layer 2 data plane management. Layer 2 forwarding is then achieved by matching directly the destination MAC address with the MAC addresses registered in the TouSIX-manager for the TouSIX users (Ethernet is used as the support technology). The second class of forwarding rules is related to the control plane. They concern specific cases that can be encountered with broadcast traffic as with ARP and NDP [67] (for the IPv6 case) traffic. In that case, the destination MAC address is a broadcast address. To avoid the aforementioned broadcast storms issues, these ARP or NDP packets are not broadcasted on the TouSIX network, but sent to a single user. In that case, the matching destination address is considered at the layer 3 level, considering the IP address contained in the ARP (or NDP) request, thus solving the ARP (or NDP) request. All these addresses are recorded in the TouSIX-manager. Recorded MAC addresses in the TouSIX-manager are, for each user, the one of their hardware Ethernet board. On the other side, the related IP addresses are assigned by the TouSIX administrator. Extra OpenFlow rules have been created for implementing this way of resolving ARP or NDP requests that permits fixing the ARP and NDP storm issues.

More generally, all packets that do not match the OpenFlow rules are dropped. Thus, unauthorized traffic is dropped at the port entry for all the members.

The application of rules in the switches takes advantage of the secure mode. This way, if the session with the OpenFlow controller is lost, the rules in production are kept active (otherwise, the switches would have run in the ancient legacy mode for the forwarding. This forwarding relies on the MAC learning principle that is the source of ARP storms). Therefore, the switches are less dependent on the state of an OpenFlow controller, i.e. even in case of failure of the OpenFlow controller, the network switches can continue working in an efficient way. This is as strong improvement for the reliability and liveliness of the IXP.

5.3.1 TouSIX architecture liveliness validation

Before going further on the deployment phase, it is first needed to check that this new architecture can bring significant improvements for the IXP management. It especially aims at verifying the liveliness of this new architecture. The first part of this validation deals with testing if the switches are correctly configured and efficiently handle link failures by testing the Fast-Failover capabilities. Second, all the OpenFlow rules deployed on the switches have been tested to check whether they work or not. All the tests are done by running the testbed depicted on Figure 5.3.

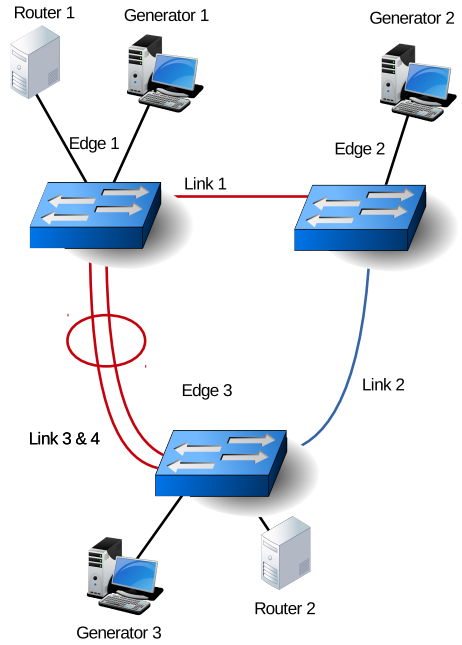

Fig. 5.3: TouSIX test topology

One host is attached to each switch. They generate traffic to all other hosts on the topology. In addition, two routers are present to maintain a BGP session. The list of considered failures is:

Tab. 5.1: TouSIX test results

Test#	Test bed state	Generators	Router2	Router1
0	All links & equipments up	OK	OK	OK
1	Link1 down	OK	OK	OK
2	Link1 up	OK	OK	OK
3	Link2 down	OK	OK	OK
4	Link2 up	OK	OK	OK
5	Link3 down	OK	OK	OK
6	Link3 up	OK	OK	OK
7	Link4 down	OK	OK	OK
8	Link4 up	OK	OK	OK
9	Link3&4 down	OK	OK	OK
10	Link3&4 up	OK	OK	OK
11	Edge1 power Off	OK	OK	OK
12	Edge1 power On	OK	OK	OK
13	Edge2 power Off	OK	OK	OK
14	Edge2 power On	OK	OK	OK
15	Edge3 power Off	OK	OK	OK
16	Edge3 power On	OK	OK	OK
17	Edge1&2&3 power Off	OK	OK	OK
18	Edge1&2&3 power On	OK	OK	OK
19	Ryu Off	OK	OK	OK
20	Ryu back On	OK	OK	OK

1. Link failure

2. Electrical failure of the switches

3. Complete shutdown of the network (T_0 test)

4. Shutdown of the OpenFlow controller

The results of these tests were positive. No particular incident has been detected on simulations or on the testbed, as shown on Table 5.1. This table represents all the tests executed. A test is considered as successful if the behaviour during the test of all devices (switches and routers) is as expected for a legacy network in similar cases. In particular, it is checked that the traffic generated by the hosts is forwarded on the correct path. If there is no path available, it is verified that packets of the sending generator are dropped. Similar verifications have been done for the BGP session between Router1 and Router2.

5.3.2 Deployment

The deployment of the new TouSIX architecture, changing radically from the legacy TouIX one, is not an easy scenario. Due to the nature of the deployment (that is a

complete migration of a commercially running infrastructure), we need to really care about the network behaviour during the migration period. In addition, one of the main attention for this migration is to keep a viable backup solution, for rolling back the network in case the migration fails. It is also essential to maintain the connectivity and quality of service for all connected members during the migration period, including the ones that are not ready to migrate to TouSIX at the date of the migration. This involves keeping both TouIX and TouSIX infrastructures running for several weeks. It is also needed not to impact the traffic sent toward the FranceIX IXP.

As a consequence, we applied some modifications on the TouSIX fully planned services:

- FranceIX VLAN access, which is available on the TouIX network, will not be transferred to the first version of the TouSIX architecture.

- A Layer 2 gateway between TouSIX and TouIX has been added. This is intended to keep the route server and the not-ready-for-migration members be connected to TouSIX members and services.

Figure 5.4 shows the new TouSIX topology at this partial level of the migration. The layer2 gateway is located on Edge3, connected to the two routers which can then communicate with any other TouSIX routers. Moreover, the passive link is not available at this stage of the migration (for simplifying the process).

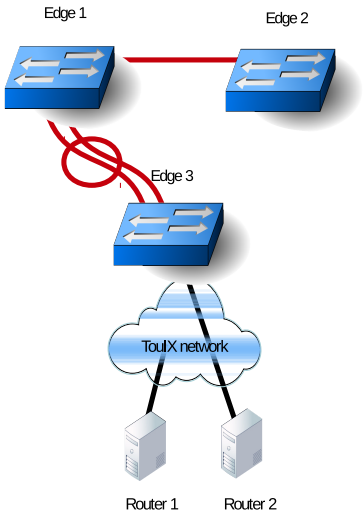

Fig. 5.4: TouSIX Migration topology

Similarly, the aggregated link was to difficult to migrate at this first stage of the OpenFlow deployment; activating this aggregated link was working perfectly on simulation, but not on the production architecture. It seems that there is an inconsis-

tency between simulation and real world with the current PicOS implementation. Up to now, this link activation has not been made possible on the production network. Fixing this issue is one of the objectives for migration to a second TouSIX architecture.

5.4 The TouSIX-Manager

The new TouSIX IXP being set-up, an efficient and convenient management tool of the control plane is required. For convenience, this tool, called TouSIX-Manager, has been designed and developed as a web platform. Its main objectives are:

- Simplify the maintenance of the TouSIX IXP.
- Ease the integration of new services through the OpenFlow protocol.

One of the requirements for the TouSIX-Manager is to be independent of any specific controller. The functionalities of the TouSIX-Manager controller consist of:

- Controlling the OpenFlow rules needed for a good functioning of the network. For instance, in case of a switch software or hardware reboot, the TouSIX-Manager must restore a stable state of the OpenFlow rules. It can be defined on a database or a local file.
- Performing OpenFlow operations without passing by any software abstraction layer to any avoid controller framework API dependency.
- Sending periodically pre-defined informations (openflow agent status, interfaces status and all flow stats) from the network to a database. These informations will be used for raising OpenFlow alerts or display statistics.

This was achieved by using the Ryu controller. As a consequence, our main architecture is divided into two main components: One or several controllers, and web applications bundled to create a fully functional website. The communication between the website and the controller uses HTTP requests.

On Figure 5.5, the website designed for managing the new TouSIX IXP is divided into four main components. The first component aims at generating all the OpenFlow rules needed for the exchange point. For any topology defined and stored on the database, all or per-switch rules can be generated on request. It is also possible to generate a certain set of rules, which correspond to functionalities we want to implement in our topology. The result is then stored in JSON format [91] in the

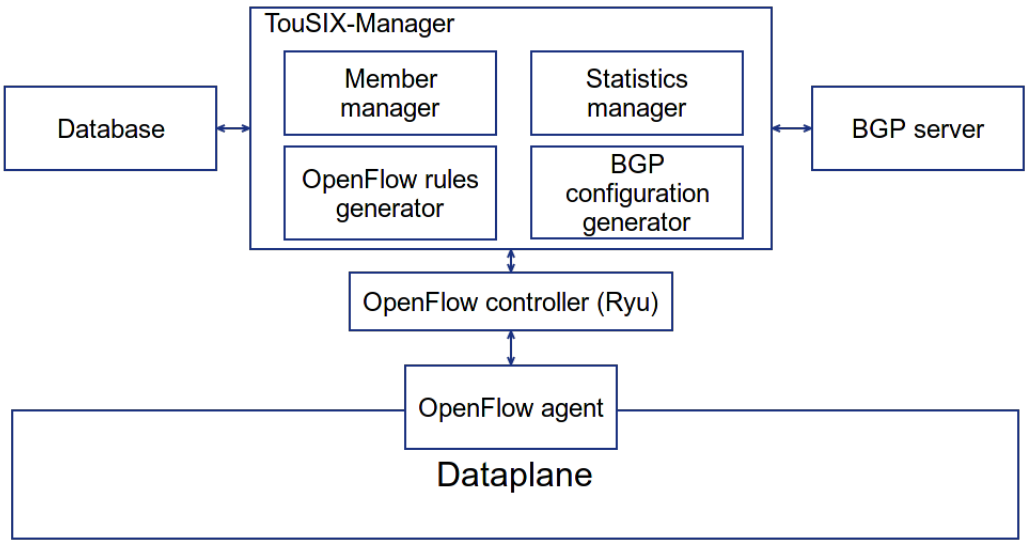

Fig. 5.5: Architecture of the TouSIX-Manager.

database. On another side, this component aims at managing the OpenFlow rules into the controller (Ryu in our case).

The second functionality of the website is to store information related to OpenFlow counters. By matching certain fields on an OpenFlow rule, without any forwarding action, we can retrieve the number of packets matching this rule. With a periodic monitoring of these counters, we can establish the bandwidth used by the matching packets. Thus, it is possible to aggregate values for representing global traffic, like in Figure 5.6.

On the current version of the TouSIX-Manager, per-member statistics on IPv4, IPv6, ICMPv6 and ARP activities are provided. More statistics can of course easily be monitored according to specific operator requirements.

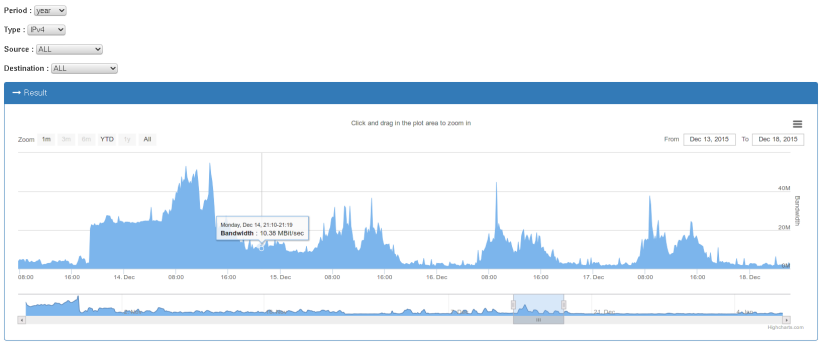

Fig. 5.6: Example of TouIX IPv4 traffic measurement provided by the TouSIX-Manager.

The third component manages all the members and users created on the website. The behaviour of this module is what we could expect from a network equipment management tool. It is not as exhaustive as other specialized solutions [49]. But, as

an improvement, the process for adding a new member to the IXP is automated. The administrator of the IXP only needs to validate data sent by the new member, and assign one access port. But there is no more technical intervention needed. Thus, it reduces the human error induced by the member or the IXP administrator.

The last functionality is an automatic configuration of our BGP server. If some changes happen on the TouIX network (eg. a new member wants to get connected), it can generate a convenient configuration for the BGP server. This component, with respect to the OpenFlow rules deployment, allows the IXP administrator to create a fully automated procedure for managing the exchange point when adding/removing members.

5.5 Evaluation

This section addresses two sets of performance measurements: one for assessing the benefit of an SDN based IXP on the management point of view, and the other for evaluating the statistic manager performance.

Rules generation profiling At this point of the TouSIX development, we want to know how fast the generation of OpenFlow rules can perform. Specifically, we need to measure the time the web application takes when some procedures (adding a member on the topology for example) are called. The time required for deploying rules on the TouSIX topology[6] will not be evaluated in the context of this thesis as it occurs only if the switches goes down, focusing only on the duration of TOUSIX-manager web procedure calls. For this purpose, some procedures of our web application have been tested with the native python profiling module [84].

Figure 5.7 depicts the total duration for generating all possible rules. As a result, the generation of rules takes much less than one second. Note that for the day-to-day operation of the TouSIX IXP, around 500 OpenFlow rules are required. With the current TouSIX configuration, it is not possible to have more realistic rules, and then push further the rule generation procedure evaluation. Figure 5.7 also exhibits an the time for computing all the rules necessary for running the TouSIX IXP. More test should be conduct with an improved implementation. As the rules required for the IXP routers member to communicate are very limited the delay is around 100 millisecond. It just exhibits that because of initialization time, the generation of the first rules is impacted by limited additional delay. For more than 500 OpenFlow rules, it seems that the rules generation procedure could be reaching its scalability limits. However, as it is not possible to extend this evaluation with more than 500+

[6]The deployment of rules that consists in sending and installing OpenFlow rules on OpenFlow agents is performed right after their global generation

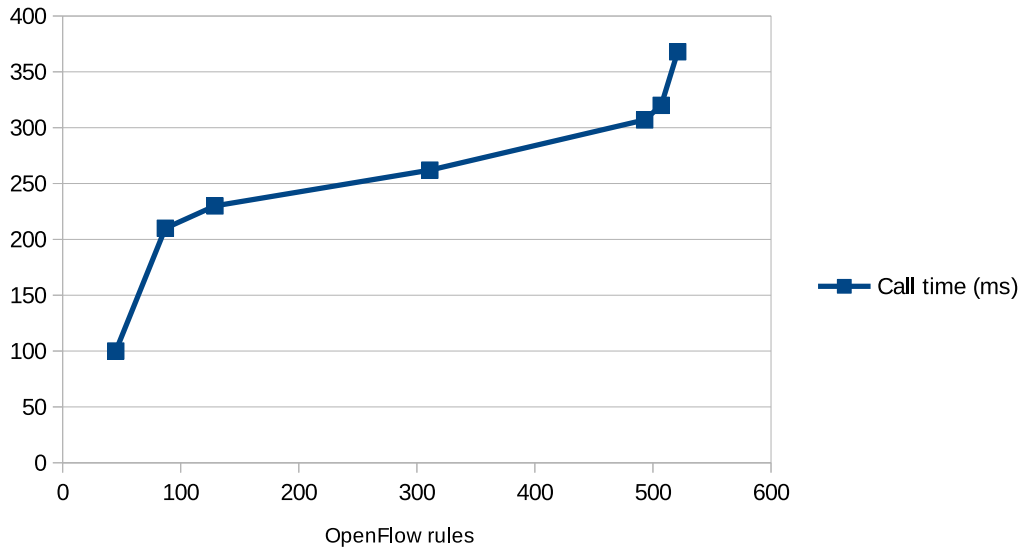

Fig. 5.7: Rules Generation module execution time

realistic rules on the current TouSIX IXP, it is not possible to give a clear conclusion yet. However, by analyzing the tests results, it is identified that most of the call duration are related to database operations. Database operations represent 95% of the total call time. We are currently working on solutions for reducing the time spent waiting a response from the database. The increase of the time for generating more than 500 rules seems to be directly related to this longer and longer database access time.

Statistic manager performance The statistics manager application is a custom implementation including two main functionalities: the management of a time-series database, and a graphing tool to visualize this database. In the following, the performance of the statistic manager are presented. They have been obtained by stressing it using the Locust [60] stress test framework.

The test performed consists in measuring the time for getting and displaying the results of different functions provided by the statistic manager including, for instance, the times required for posting or getting flow statistics, aggregated traffic statistics, etc. The results have been obtained on sequences of 20 requests per seconds in order to really stress the statistics manager, and also for limiting measurement errors. The results are displayed on Figure 5.8. It especially exhibits that on an average of 20 requests per second, the mean response time is increasing. It also shows that with the most time consuming requests in terms of computing resources (when computing global traffic statistics for instance), it can take up to 2.6 seconds to get the results. In addition, it is observed that these huge delays accumulate, leading to non real-time display of the traffic statistics. This is of course a strong limit of the current version of the statistic manager that will be addressed in future work.

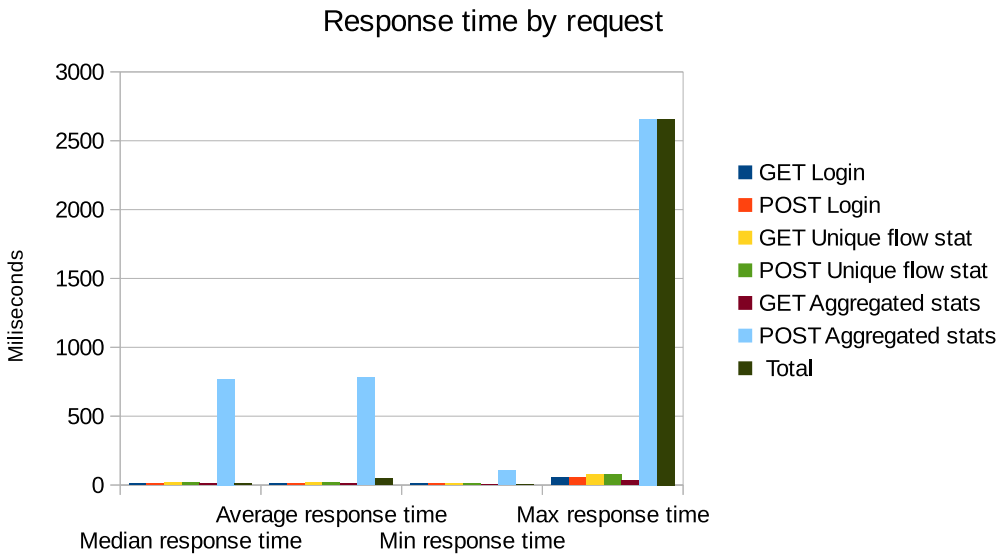

Fig. 5.8: Statistics manager response time

5.6 Conclusion

This chapter illustrated how an IXP can benefit from an SDN technology as OpenFlow for enhancing reliability and manageability using Umbrella approach. This has been performed on the real TouSIX IXP where OpenFlow based devices, tools and applications have been successfully deployed, thus demonstrating the practical applicability and benefits of such a solution. As a result, TouSIX now fully leverages OpenFlow for its day-to-day operations. As future work, we target solutions for improving the scalability of the SDN/OpenFlow solution, as well as introducing ways for securing such an SDN based IXP.

6 Conclusion

This thesis discusses the importance of programmability within the network fabric. The steadily growing importance of Internet eXchange Points in the Internet ecosystem on one side and the programmability introduced by the SDN paradigm on the other, have opened up new possibilities at an inter-domain level.

In this scenario, we fashioned a new SDN-enabled exchange fabric which is reliable, easy to manage but most importantly fully programmable. The Umbrella SDN network design gives a stronger separation of control and data plane functionalities, leading to an enhanced scalability, reliability, and manageability. We proposed a new approach to IXP management that leverages SDN programmability to tackle part of the control directly within the data plane. A significant benefit of Umbrella is that it removes all broadcast traffic. Integrating Umbrella in today's IXP architecture is straightforward as we maintain a strict separation between layer 2 and layer 3 functions.

This thesis equally acknowledges the importance of continuous network monitoring and testing for level production architectures. We studied and architected new prototypes for open-source monitoring and testing (e.g., OSNT and OFLOPS). Most notably, the OSNT architecture enables a flexible combination of multiple packet-processing pipelines using a new virtualization technique. The proposed solution is not bound to a specific hardware (though it has been developed on the NetFPGA platform) and should be able to provide flexibility and versatility across a range of uses. OFLOPS enhances OSNT by providing a tool for the OpenFlow community in order to acquire a better understanding of the performance impact of OpenFlow implementations and motivate a community of rigorous OpenFlow testing.

Finally, this thesis demonstrates the practicality of the Umbrella approach in a real-world scenario. One of the key results of this thesis is indeed TouSIX, the first SDN-enabled european IXP. In the last chapter, we have presented the work undertaken in greater detail to migrate the legacy switching fabric in Umbrella. TouSIX has been running for more than a year and has proved to increase stability, manageability, and monitoring. Thus demonstrating the practical applicability and benefits of such a solution. As a result, TouSIX now fully leverages OpenFlow for its day-to-day operations. For future work, we will target solutions for improving the

scalability of the SDN/OpenFlow solution, as well as introduce ways for securing such an SDN-based IXP.

I believe this is the first step towards a new concept of the network fabric, which is no longer constrained by vendor specific protocols. Open Source and White Box solution used in our approach can reduce operational costs and reduce access fees for all members. I think the simplification of the IXPs fabric core switch with the 3 heuristic steps proposed by Umbrella could drastically reduce the management cost on removing completely any management interface. Programming all forwarding and path decisions at the edge of the fabric reduces the amount of the control demand significantly.

Acronyms

OpenFlow OpenFlow Protocol

OSNT Open Source Network Tester

OFLOPS OpenFLow Operations Per Second

IETF Internet Engineering Task Force

IoT Internet of Things

ICT Information and Communications Technology

OSI Open Systems Interconnection model

Euro-IX Euro-Internet eXchange

MAC Media Access Control

VLAN Virtual Local Area Network

IGP Interior Gateway Protocol

OSPF Open Shortest Path First

IS-IS Intermediate System to Intermediate System

RS Route Server

RIR Regional Internet Registry

IANA Internet Assigned Numbers Authority

LAN Local Area Network

LDP Label Distribution Protocol

MPLS MultiProtocol Label Switching

LSP Label Switched Path

TRILL Transparent Interconnect of Lots of Links

RSTP Rapid Spanning Tree Protocol

IEEE Institute of Electrical and Electronics Engineers

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

SDN Software Defined Networking

BGP Border Gateway Protocol

ISP Internet Service Provider

IXP Internet eXchange Point

ISP Internet Service Provider

AS Autonomous System

IP Internet Protocol

PoP Point of Presence

VPN Virtual Private Network

VPLS Virtual Private LAN Services

IETF Internet Engineering Task Force

UDP User Datagram Protocol

EVPN Ethernet Virtual Private Network

VXLAN Virtual Extensible Local Area Network

ARP Address Resolution Protocol

Bibliography

[1]Niels L.M. Van Adrichem, Benjamin J. Van Asten, and Fernando a. Kuipers. „Fast Recovery in Software-Defined Networks“. In: *2014 Third European Workshop on Software Defined Networks* (2014), pp. 61–66 (cited on page 42).

[2]B. Ager, N. Chatzis, A. Feldmann, et al. „Anatomy of a Large European IXP“. In: *SIGCOMM*. ACM, 2012 (cited on pages 31, 32, 52).

[8]*AMS-IX Annual Reports*. `https://ams-ix.net/about/annual-reports--2`. [Online; accessed 23-Sep-2015] (cited on page 45).

[9]G. Antichi, M. Shahbaz, Y. Geng, et al. „OSNT: Open Source Network Tester“. In: *IEEE Network* Special issue on Open Source for Networking: Tools and Applications (2014) (cited on page 74).

[10]Gianni Antichi, Stefano Giordano, David J Miller, and Andrew W Moore. „Enabling open-source high speed network monitoring on NetFPGA“. In: *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. 2012 (cited on pages 64, 81).

[11]B. Augustin, B. Krishnamurthy, and W. Willinger. „IXPs: Mapped?“ In: *Proceedings of SIGCOMM*. 2009 (cited on page 52).

[12]Bernd Beckert. „Network neutrality from an innovation research perspective“. In: *2011 50th FITCE Congress - "ICT: Bridging an Ever Shifting Digital Divide"* (2011), pp. 1–5 (cited on page 23).

[13]Theophilus Benson, Aditya Akella, and David Maltz. „Unraveling the Complexity of Network Management“. In: *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*. NSDI'09. Boston, Massachusetts: USENIX Association, 2009, pp. 335–348 (cited on page 24).

[15]Michaela Blott, Jonathan Ellithorpe, Nick McKeown, Kees Vissers, and Hongyi Zeng. „FPGA research design platform fuels network advances“. In: *Xilinx Xcell Journal* 73 (2010) (cited on page 70).

[16]Nicola Bonelli, Andrea Di Pietro, Stefano Giordano, and Gregorio Procissi. „Flexible high performance traffic generation on commodity multi–core platforms“. In: *Traffic Monitoring and Analysis*. Springer, 2012 (cited on page 80).

[17]V. Boteanu and H. Bagheri. „Minimizing ARP traffic in the AMS-IX switching platform using OpenFlow“. MA thesis. the Netherlands: Universiteit van Amsterdam, 2013 (cited on page 2).

[18]S. Bryant and P. Pate. *Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture*. RFC 3985. RFC Editor, 2005 (cited on page 45).

[19]K. Butler, T. Farley, P. Mcdaniel, and J. Rexford. *A Survey of BGP Security Issues and Solutions*. Tech. rep. AT&T Labs - Research, Florham Park, NJ, 2004 (cited on page 47).

[20]Juan Camilo Cardona and Rade Stanojevic. „IXP Traffic: A Macroscopic View“. In: *Proceedings of LANC*. 2012 (cited on pages 41, 52).

[21]M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian. „Fabric: A Retrospective on Evolving SDN“. In: *Hot Topics in Software Defined Networks (HotSDN)*. ACM, 2012 (cited on page 46).

[22]Vinton G Cerf and Robert E Kahn. „A Protocol for Pocket Network Intercommunication“. In: C.5 (1974) (cited on page 7).

[23]Nikolaos Chatzis, Georgios Smaragdakis, Anja Feldmann, and Walter Willinger. „There Is More to IXPs than Meets the Eye“. In: *CCR* 43.5 (2013), pp. 19–28 (cited on page 52).

[26]Adam Covington, Glen Gibb, John W Lockwood, and Nick Mckeown. „A packet generator on the NetFPGA platform“. In: *IEEE Symposium on Field Programmable Custom Computing Machines (FCCM)*. 2009 (cited on pages 64, 81).

[28]*Dell Networking S4810*. `http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell_Force10_S4810_Spec_sheet.pdf`. 2014 (cited on page 79).

[29]R Doriguzzi-corin, E Salvadori, and P A Aranda Guti. „NetIDE : removing vendor lock-in in SDN“. In: i (), pp. 11–12 (cited on page 23).

[30]*EU FP7 ALIEN Project page*. `http://www.fp7-alien.eu`. 2014 (cited on page 77).

[32]Nick Feamster, Jennifer Rexford, Scott Shenker, et al. „SDX: A software-defined Internet exchange“. In: *Open Networking Summit* (2013), pp. 2–3 (cited on pages 2, 78).

[34]*FranceIX website*. `https://www.franceix.net/en/events-and-news/news/franceix-outage-notification/`. [Online; accessed 29-Jan-2016] (cited on pages 33, 85).

[35]Ali Ghodsi, Scott Shenker, Teemu Koponen, et al. „Intelligent design enables architectural evolution“. In: *Proceedings of the 10th ACM Workshop on Hot Topics in Networks - HotNets '11* (2011), pp. 1–6 (cited on page 24).

[36]Natasha Gude, Teemu Koponen, Justin Pettit, et al. „NOX: towards an operating system for networks“. In: *SIGCOMM Computer Communication Review* 38.3 (2008), pp. 105–110 (cited on page 26).

[37]Arjun Guha, Mark Reitblatt, and Nate Foster. „Machine-verified Network Controllers“. In: *SIGPLAN Not.* 48.6 (June 2013) (cited on page 74).

[38]A. Gupta, L. Vanbever, M. Hahbaz, et al. „SDX: A Software Defined Internet Exchange“. In: *SIGCOMM*. ACM, 2014 (cited on pages 2, 37, 42, 43).

[39]Arpit Gupta, Robert MacDavid, Rüdiger Birkner, et al. „iSDX: An Industrial-Scale Software Defined Internet Exchange Point“. In: *Networked Systems Design & Implementation (NSDI)*. USENIX Association. 2016.

[40]*Hibernia IX monitoring tool*. http://data.ix.atrato.net/pcap. [Online; accessed 29-Jan-2016] (cited on page 33).

[41]*High-Capacity StrataXGS Trident Ethernet Switch Series with Integrated 10G Serial PHY - BCM56840 Series*. http://www.broadcom.com/products/Switching/Carrier-and-Service-Provider/BCM56840-Series. 2014 (cited on page 79).

[42]N. Hilliard, E. Jasinska, R. Raszuk, and N. Bakker. *Internet Exchange Route Server Operations*. Tech. rep. (cited on pages 22, 31).

[43]M. Hughes, M. Pels, and H. Michl. *Internet Exchange Point Wishlist*. https://www.euro-ix.net/ixps/ixp-wishlist/. [Online; accessed 29-Jan-2016]. 2015 (cited on pages 1, 37).

[47]*iperf, TCP and UDP bandwidth performance measurement tool, http://code.google.com/p/iperf.* (Cited on page 80).

[49]*IXP-Manager source code*. https://github.com/inex/IXP-Manager. [Online; accessed 23-Sep-2015] (cited on page 91).

[50]Hani Jamjoom, Dan Williams, and Upendra Sharma. „Don't call them middleboxes, call them middlepipes". In: *Proceedings of the third workshop on Hot topics in software defined networking (HotSDN '14)* (2014), pp. 19–24 (cited on page 26).

[52]Hyojoon Kim and Nick Feamster. „Improving network management with software defined networking". In: *IEEE Communications Magazine* 51.2 (2013), pp. 114–119 (cited on page 24).

[53]L Kleinrock. *Information flow in large communication nets*. 1961 (cited on page 6).

[54]C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. „Netalyzr: Illuminating The Edge Network". In: *ACM Internet Measurement Conference (IMC)*. 2010 (cited on page 80).

[55]M. Lasserre and V. Kompella. *Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling*. RFC 4762. http://www.rfc-editor.org/rfc/rfc4762.txt. RFC Editor, 2007 (cited on page 20).

[58]Barbara Liskov and Stephen Zilles. „Programming with abstract data types". In: *Proceedings of the ACM SIGPLAN symposium on Very high level languages* (1974), pp. 50–59 (cited on page 24).

[59]John W Lockwood, Nick McKeown, Greg Watson, et al. „NetFPGA–an open platform for gigabit-rate network switching and routing". In: *IEEE International Conference on Microelectronic Systems Education (MSE)*. 2007 (cited on page 70).

[60]*Locust.io website*. http://locust.io/. [Online; accessed 23-Sep-2015] (cited on page 93).

[61]A. Lodhi, N. Larson, A. Dhamdhere, C. Dovrolis, and k.c. Claffy. „Using peeringDB to Understand the Peering Ecosystem". In: *SIGCOMM Computer Communication Review* (2014) (cited on page 48).

[62]M. Mahalingam, D. Dutt, K. Duda, et al. *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. RFC 7348. http://www.rfc-editor.org/rfc/rfc7348.txt. RFC Editor, 2014 (cited on page 21).

[64]Nick McKeown, Tom Anderson, Hari Balakrishnan, et al. „OpenFlow: Enabling Innovation in Campus Networks“. In: *Computer Communication Review* (2008) (cited on pages 26, 32, 42).

[65]Andrew Moore, James Hall, Christian Kreibich, Euan Harris, and Ian Pratt. „Architecture of a network monitor“. In: *Passive & Active Measurement Workshop*. 2003 (cited on pages 71, 80).

[66]T. Narten, E. Nordmark, W. Simpson, and H. Soliman. *NCP/TCP TRANSITION PLAN*. Tech. rep. 2007 (cited on page 7).

[67]T. Narten, E. Nordmark, W. Simpson, and H. Soliman. *Neighbor Discovery for IP version 6 (IPv6)*. Tech. rep. IETF, 2007 (cited on pages 39, 86).

[68]*Netperf, http://www.netperf.org* (cited on page 80).

[69]P. Newman, G. Minshall, and T.L. Lyon. „IP switching-ATM under IP“. In: *IEEE/ACM Transactions on Networking* 6.2 (1998), pp. 117–129 (cited on page 26).

[70]OECD. „Broadband Networks and Open Access“. In: 218 (2013) (cited on page 14).

[71]Open-IX. *IXP Technical Requirements OIX-1*. `http://www.open-ix.org/standards/ixp-technical-requirements`. [Online; accessed 29-Jan-2016] (cited on page 37).

[74]*Open vSwitch website*. `http://openvswitch.org/`. [Online; accessed 23-Sep-2015] (cited on page 86).

[75]*OpenFlow Switch Specification*. `http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf`. [Online; accessed 29-Jan-2016] (cited on page 37).

[76]J Pan, S Paul, and R Jain. „A survey of the research on future internet architectures“. In: *Communications Magazine, IEEE* 49.7 (2011), pp. 26–36 (cited on page 7).

[77]Christos Pappas, Katerina Argyraki, Stefan Bechtold, et al. „Transparency Instead of Neutrality“. In: *HotNets '15 November 16–17 2015, Philadelphia, PA USA* (2015) (cited on page 23).

[79]*PeeringDB website*. `http://www.peeringdb.com`. [Online; accessed 29-Jan-2016] (cited on page 48).

[80]I. Pepelnjak. *Could IXPs Use OpenFlow to Scale?* The Middle East Network Operators Group (MENOG). 2012 (cited on pages 32, 37, 42, 43).

[81]R. Perlman, D. Eastlake, D. Dutt, S. Gai, and A. Ghanwani. *Routing Bridges (RBridges): Base Protocol Specification*. RFC 6325. `http://www.rfc-editor.org/rfc/rfc6325.txt`. RFC Editor, 2011 (cited on page 21).

[82]*Pica8 P3922 Specifications*. `http://www.pica8.com/documents/pica8-datasheet-64x10gbe-p3922-p3930.pdf`. 2014 (cited on page 78).

[83]Matthew Prince. *The DDoS That Almost Broke the Internet*. `https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/`. [Online; accessed 29-Jan-2016] (cited on page 42).

[84]*Python cProfile module*. `https://docs.python.org/3.5/library/profile.html`. [Online; accessed 17-Dec-2015] (cited on page 92).

[85]Barath Raghavan, Martín Casado, Teemu Koponen, et al. „Software-defined internet architecture: decoupling architecture from infrastructure“. In: *Proceedings of the 11th ACM Workshop on Hot Topics in Networks* (2012), pp. 43–48 (cited on page 24).

[87]P. Richter, G. Smaragdakis, A. Feldmann, et al. „Peering at Peerings: On the Role of IXP Route Servers“. In: *Internet Measurement Conference (IMC)*. ACM, 2014 (cited on pages 21, 31, 41).

[88]Luigi Rizzo. „Netmap: a novel framework for fast packet I/O“. In: *USENIX Annual Technical Conference (ATC)*. 2012 (cited on page 80).

[89]Lawrence G. Roberts. „Multiple computer networks and intercomputer communication“. In: *Proceedings of the first ACM symposium on Operating* (1967), pp. 1–6 (cited on page 6).

[90]C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A.W. Moore. „OFLOPS: An Open Framework for OpenFlow Switch Evaluation“. In: *Passive and Active Measurement (PAM)*. 2012 (cited on page 74).

[91]*Ryu REST API definition*. `http://ryu.readthedocs.org/en/latest/app/ofctl_rest.html`. [Online; accessed 23-Sep-2015] (cited on page 90).

[92]*Ryu SDN controller*. `http://osrg.github.io/ryu`. [Online; accessed 29-Jan-2016] (cited on page 43).

[93]*Ryu website*. `http://osrg.github.io/ryu/`. [Online; accessed 23-Sep-2015] (cited on page 86).

[94]A. Sajassi, R. Aggarwal, N. Bitar, et al. *BGP MPLS-Based Ethernet VPN*. RFC 7432. `http://www.rfc-editor.org/rfc/rfc7432.txt`. RFC Editor, 2015 (cited on page 21).

[95]P. Saul. „Direct Digital Synthesis“. In: *Circuits and Systems Tutorials*. 1996 (cited on page 69).

[96]Brandon Schlinker, Kyriakos Zarifis, Italo Cunha, et al. „Try Before you Buy: SDN Emulation with (Real) Interdomain Routing“. In: *Presented as part of the Open Networking Summit 2014 (ONS 2014)*. Santa Clara, CA: USENIX, 2014 (cited on pages 43, 51, 55).

[97]Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. „Enabling Fast Failure Recovery in OpenFlow Networks“. In: *IEEE Design of Reliable Communication Networks* (2011), pp. 164–171 (cited on page 42).

[98]Ieee Computer Society. *IEEE 801.1D - MAC Bridges*. Vol. 2004. June. 2004, pp. 1–281 (cited on page 31).

[99]P. Srivats. „OSTINATO: An open, scalable packet/traffic generator“. In: *FOSS.IN*. 2010 (cited on page 80).

[100]*Tcpreplay, https://github.com/synfinatic/tcpreplay.* (Cited on page 80).

[101]Massimo Ussoli and Gunnar Prytz. „SNTP Time Synchronization Accuray Measurements“. In: *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2013 (cited on page 80).

[102]Kashi Venkatesh Vishwanath and Amin Vahdat. „Swing: Realistic and responsive network traffic generation“. In: *IEEE/ACM Transactions on Networking* 17.3 (2009) (cited on page 80).

[103]Richard Wallis and David Buckingham. *Open Architecture as Communications Policy*. 2013 (cited on page 6).

[104]M. Wessel and N. Sijm. „Effects of IPv4 and IPv6 address resolution on AMS-IX and the ARP Sponge“. MA thesis. the Netherlands: Universiteit van Amsterdam, 2009 (cited on page 32).

[105]Tim Wu. „Network Neutrality , Broadband Discrimination“. In: *Journal on Telecommunications & High Technology Law* 2.2001 (2003), pp. 141–179 (cited on page 23).

Web & Online references:

[@3]AMS-IX. *Allowed Traffic Types on Unicast Peering LANs*. 2016. URL: http://ams-ix.net/technical/specifications-descriptions/allowed-traffic (visited on May 25, 2016) (cited on page 37).

[@4]AMS-IX. *AMS-IX policy statement*. 2015. URL: https://ams-ix.net/about/strategy--2/ams-ix-policy-statement (visited on Mar. 28, 2016) (cited on page 23).

[@5]AMS-IX. *ARP Sponge*. 2016. URL: https://ams-ix.net/technical/specifications-descriptions/controlling-arp-traffic-on-ams-ix-platform (visited on Mar. 28, 2016) (cited on page 35).

[@6]AMS-IX. *Controlling ARP traffic on AMS-IX platform*. 2016. URL: https://ams-ix.net/technical/specifications-descriptions/controlling-arp-traffic-on-ams-ix-platform (visited on May 25, 2016) (cited on page 2).

[@7]AMS-IX. *historical-traffic-data*. 2016. URL: https://ams-ix.net/technical/statistics/historical-traffic-data (visited on Mar. 28, 2016) (cited on pages 1, 11).

[@14]Bill Snyder - Ed Snowden. *Snowden the nsa planted backdoors in cisco products*. 2015. URL: http://www.infoworld.com/article/2608141/internet-privacy/snowden--the-nsa-planted-backdoors-in-cisco-products.html (visited on Mar. 28, 2016) (cited on page 24).

[@24]CIDR-Report. *Plot Unique ASes*. 2016. URL: http://www.cidr-report.org/as2.0/ (visited on Mar. 28, 2016) (cited on pages 1, 10).

[@25]CISCO. *The Internet of Things. How the Next Evolution of the Internet is Changing Everything*. 2011. URL: http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (visited on Mar. 28, 2016) (cited on page 8).

[@27]DE-CIX. *DE-CIX non profit ECO organization*. 2015. URL: https://www.de-cix.net/get-connected/faq/ (visited on Mar. 28, 2016) (cited on page 23).

[@31]EURO-IX. *Internet Exchange MAP EURO-IX*. 2016. URL: https://www.euro-ix.net/ixps/ixp-map/ (visited on Mar. 28, 2016) (cited on page 16).

[@33]Federal Networking Council. *Internet Definition*. 2015. URL: https://www.nitrd.gov/fnc/Internet_res.aspx (visited on Mar. 28, 2016) (cited on page 5).

[@44]IEEE. *802 LAN/MAN architecture Working Group*. 2016. URL: http://www.ieee802.org/1/ (visited on Mar. 28, 2016) (cited on page 17).

[@45]IEEE. *IEEE Rapid Reconfiguration of Spanning Tree*. 2015. URL: http://www.ieee802.org/1/pages/802.1w.html (visited on Mar. 28, 2016) (cited on pages 20, 31).

[@46]International Telecommunication Union. *International Telecommunication Union Statistics*. 2016. URL: https://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2015/ITU_Key_2005-2015_ICT_data.xls (visited on Mar. 28, 2016) (cited on pages 8, 9).

[@48]ITU. *global ICT ITU stats*. 2016. URL: https://www.itu.int/en/ITU-D/Statistics/Pages/publications/wtid.aspx (visited on Mar. 28, 2016) (cited on pages 1, 8).

[@51]Jensen, M. *Promoting the Use of Internet Exchange Points: A Guide to Policy, Management, and Technical Issues*. 2016. URL: http://www.internetsociety.org/sites/default/files/promote-ixp-guide_0.pdf (visited on May 25, 2016) (cited on page 32).

[@56]Licklider. *ARPA notes of the Intergalatic Computer Network*. 1963. URL: http://worrydream.com/refs/Licklider-IntergalacticNetwork.pdf (visited on Mar. 28, 2016) (cited on page 6).

[@57]LINX. *LINX Good for Internet*. 2015. URL: https://www.linx.net/about/good-of-the-internet (visited on Mar. 28, 2016) (cited on page 23).

[@63]Maksym Tulyuk AMS-IX. *Danger of Proxy ARP in IX environment*. 2011. URL: https://ripe63.ripe.net/presentations/130-Proxy_ARP_RIPE_Nov2011.pdf (visited on Mar. 28, 2016) (cited on page 2).

[@72]Open-IX. *Open-IX Framework-V5.2*. 2015. URL: http://www.open-ix.org/wp-content/uploads/2013/12/Open-IX_Framework-V5.2.pdf (visited on Mar. 28, 2016) (cited on page 23).

[@73]Open Networking Foundation. *ONF Overview*. 2016. URL: https://www.opennetworking.org/about/onf-overview (visited on Mar. 28, 2016) (cited on page 26).

[@78]PCH. *Internet Exchange Directory*. 2016. URL: https://prefix.pch.net/applications/ixpdir/ (visited on Mar. 28, 2016) (cited on page 16).

[@86]Remco Van Mook. *The 1000 Dollar IXP*. 2015. URL: https://ripe71.ripe.net/presentations/30-1000-dollar-exchange-ripe71.pdf (visited on Mar. 28, 2016) (cited on page 16).

List of Figures

List of Tables

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.