Technische Universität Berlin
Fakultät für Elektrotechnik und Informatik
Lehrstuhl für Intelligente Netze
und Management Verteilter Systeme

# On Workload-driven Router Designs

vorgelegt von

Nadi Sarrar (Dipl.-Inf.)

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Dissertation

**Promotionsausschuss:**

| | |
|---|---|
| Vorsitzender: | Prof. Dr. Axel Küpper, Technische Universität Berlin, Germany |
| Gutachterin: | Prof. Anja Feldmann, Ph.D., Technische Universität Berlin, Germany |
| Gutachter: | Prof. Dr. Steve Uhlig, Queen Mary, University of London, UK |
| Gutachter: | Prof. Roch A. Guérin, Ph.D., University of Pennsylvania, USA |

Tag der wissenschaftlichen Aussprache: 19. November 2012

Berlin 2012
D 83

Ich versichere an Eides statt, dass ich diese Dissertation selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Datum Nadi Sarrar

## Abstract

Over the past 15 years, the number of networks connected to the Internet, also called Autonomous Systems (ASes), grew at least by a factor of ten. Accordingly, estimations of the global IP traffic further emphasize this trend with a growth by four orders of magnitude. All this puts persistent pressure toward the core infrastructure of the Internet which is required to keep up with the increasing traffic demand. Consequently, Internet routers need to maintain dynamic forwarding information for a large number of networks in a very fast memory, such that a destination lookup operation can be performed on each incoming packet at speed to comply with the low-latency requirements in Internet Service Provider (ISP) networks. The hardly foreseeable, indispensable capacity of this type of specialized memory and the difficulties involved in handling the churn in its contents are primary obstacles in Internet router scalability, notably their price tag and energy consumption.

In this thesis we rely on insights derived from Internet measurements to design a router architecture that relaxes the requirements for fast destination lookup memory. Firstly, a very small fraction of all networks (IP prefixes) capture most of the traffic — an opportunity for traffic offloading. Secondly, the Forwarding Information Base (FIB) can be compressed — an opportunity to reduce FIB memory requirements. Based on these observations we propose a novel IP router design that follows the split architecture paradigm; separating control from data plane functionality. The control plane is a complex, software-centric system that runs routing protocol daemons and acts as the data plane controller. The data plane on the other hand is a specialized, hardware-centric forwarding engine that provides the required primitives for IP routing. A communication channel constitutes a bridge connecting the two entities.

Advances in open-source software routers as well as programmable switches enable us to build an IP router prototype that follows our design principles. The emergent developments in the Software Defined Networking (SDN) community and the OpenFlow protocol in particular match the needs of the communication channel and let us demonstrate the feasibility of our design. To complete the study of our prototype, we implement a framework that allows to spot critical performance impairments of OpenFlow-enabled switches.

In this dissertation we raise the point that efficient and scalable router designs can be found by leveraging knowledge about router workload properties, *i.e.*, Internet traffic. Accordingly, we conduct an extensive study of Internet traffic as seen by a large European Internet Exchange Point (IXP). The business model of an IXP is to provide a switching fabric that allows a large number of ASes to establish peering relationships and to exchange traffic. Member ASes connect their routers to the IXP's switching fabric and announce IP prefix-based forwarding information to one another. The large number of neighboring routers that a single member router peers

with is unique to IXPs. In fact, a large fraction of the AS-to-AS links at our IXP are invisible from the rest of the Internet, which suggests a re-assessment of the mental picture that the research community has about the AS-level structure of the Internet.

Finally, we study the recent growth of IPv6 traffic, the latest version of the Internet Protocol. The World IPv6 Day in June 2011 was a milestone in the adoption of IPv6. We perform measurements of IPv6 traffic around the World IPv6 Day and show, that IPv6 traffic has doubled in volume and its application mix is getting similar to that of IPv4, suggesting a push toward mainstream use of IPv6 in the Internet.

# Zusammenfassung

Die Anzahl der an das Internet angeschlossenen Netzwerke, sogenannte Autonome Systeme (AS), hat sich in den vergangenen 15 Jahren verzehnfacht. Dementsprechend wuchs auch das globale Internet-Datenverkehrsvolumen stark an; Schätzungen zufolge sogar um vier Größenordnungen. Diese Entwicklungen setzen die zu Grunde liegende Infrastruktur des Internets stark unter Druck, da diese mit den ständig wachsenden Anforderungen des auftretenden Datenverkehrs mithalten können muss. Internet-Router verwalten dynamische Pfadinformationen für eine große Menge an Zielnetzwerken in sehr schnellem Speicher, so dass pro eingehendem Datenpaket mit minimaler Verzögerung der zum Ziel führende Pfad ermittelt werden kann. Hierfür werden hochspezialisierte Speichertechnologien eingesetzt. Aufgrund der kaum absehbaren zukünftigen Anforderungen an deren Speicherkapazitäten sowie der Schwierigkeit, die hohen Raten an Pfadänderungen in dem Speicher zeitnah abzubilden, stellt dieser Speicher eine zentrale Hürde im Sinne der Skalierbarkeit, des Preises und des Energieverbrauchs von Internet-Routern dar.

Aufbauend auf Erkenntnissen aus Datenverkehrsmessungen entwickelt die vorliegende Dissertation eine Router-Architektur mit deutlich geringeren Anforderungen an die Kapazität des Speichers für Pfadinformationen. Zum einen ist bekannt, dass der größte Anteil des Datenverkehrs nur wenigen Zielnetzwerken zugeordnet ist; eine Gelegenheit für sogenanntes „Traffic-Offloading“. Zum anderen lässt sich der Inhalt des Pfadspeichers, die „Forwarding Information Base“ (FIB), komprimieren, wodurch sich dessen Speicheranforderungen reduzieren lassen. Auf Basis dieser Beobachtungen wird in dieser Dissertation eine neuartige Router-Architektur vorgestellt, welche dem „Split-Architecture“ Konzept entspricht: Sie trennt Daten- und Kontrollfunktionalitäten („Data Plane“ und „Control Plane“) voneinander. Dabei stellt die Control Plane ein komplexes, softwarezentrisches System dar, welches Routingprotokolle ausführt und das Verhalten der Data Plane kontrolliert. Die Data Plane hingegen fungiert als eine hochperformante, hardwarezentrische Datenvermittlungseinheit, entsprechend der Anforderungen des IP-Routing. Ein Kommunikationskanal verbindet diese beiden Funktionseinheiten.

Durch die technischen Fortschritte in quelloffenen Software-Routern sowie in programmierbaren Switchen wird im Rahmen dieser Dissertation ein Router-Prototyp entsprechend den vorgestellten Designprinzipien entwickelt. Die Entwicklungen im Bereich von Software Defined Networks (SDN), sowie insbesondere dem OpenFlow Protokoll, erfüllen die Anforderungen an den Kommunikationskanal und bieten damit die technische Grundlage des Prototyps. Schließlich wird ein Testsystem entwickelt, welches entscheidende Leistungsaspekte von OpenFlow Switchen herausstellt.

Die aus dieser Dissertation gewonnenen Erkenntnisse verdeutlichen, dass unter Berücksichtigung der Arbeitslast effiziente und skalierbare Router-Architekturen entwickelt werden können. Dementsprechend werden umfangreiche Messungen von

Internet-Datenverkehr eines europäischen Internet Exchange Point (IXP) durchgeführt. Das Geschäftsmodell eines IXP besteht darin, eine Switch-basierte Netzwerkstruktur bereitzustellen, die es einer großen Anzahl an ASen erlaubt, untereinander Verbindungen für einen direkten Datenaustausch aufzubauen. Entsprechend verbindet ein teilnehmendes AS mindestens einen seiner Router mit der IXP Infrastruktur und tauscht dort mit anderen angeschlossenen ASen Routinginformationen aus. Die große Anzahl an derartigen Verbindungen, die ein Router mit Routern anderer ASe aufbaut, ist eine Besonderheit für IXPs. Dementsprechend lässt diese Dissertation eine Vielzahl an direkten Verbindungen zwischen ASen sichtbar werden, die für den Rest des Internets verdeckt sind. Aufgrund dieser Erkenntnis wird das übliche Verständnis der wissenschaftlichen Gesellschaft über die Internetstruktur auf AS-Ebene in Frage gestellt.

Abschliessend untersucht diese Dissertation den jüngsten Anstieg von Datenverkehr über IPv6, der neuen Version des Internet Protokolls. Im Juni 2011 fand mit dem World IPv6 Day ein Meilenstein in der Adaption des IPv6 Protokolls statt. Diese Dissertation führt Messungen im Zeitraum des World IPv6 Day durch und zeigt eine Verdopplung des IPv6-Datenvolumens mit einer IPv4-ähnlichen Zusammensetzung an beitragenden Applikationen, ein Indiz für die nachhaltige Verbreitung von IPv6 im Internet.

## Pre-published Papers

Parts of this thesis are based on pre-published papers co-authored with other researchers. I thank all of my co-authors for their valuable contributions. All co-authors have been acknowledged as scientific collaborators of this work.

### International Journals

Nadi Sarrar, Steve Uhlig, Anja Feldmann, Rob Sherwood, Xin Huang. **Leveraging Zipf's Law for Traffic Offloading.** In *ACM SIGCOMM Computer Communications Review (CCR)*. January 2012.

### International Conferences

Nadi Sarrar, Gregor Maier, Bernhard Ager, Robin Sommer, Steve Uhlig. **Investigating IPv6 Traffic – What happened at the World IPv6 Day?** In Proceedings of *Passive and Active Measurements Conference (PAM)*. Vienna, Austria. March 2012.

Charalampos Rotsos, Nadi Sarrar, Steve Uhlig, Rob Sherwood, Andrew W. Moore. **OFLOPS: An Open Framework for OpenFlow Switch Evaluation.** In Proceedings of *Passive and Active Measurements Conference (PAM)*. Vienna, Austria. March 2012.

Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, Walter Willinger. **Anatomy of a Large European IXP.** In Proceedings of *ACM SIGCOMM*. Helsinki, Finland. August 2012.

### Technical Reports

Nadi Sarrar, Marcin Bienkowski, Stefan Schmid, Steve Uhlig and Robert Wuttke. **Exploiting Locality of Churn for FIB Aggregation.** TU Berlin Technical Report, No. 2012-12. November 2012.

### Workshops

Nadi Sarrar, Anja Feldmann, Steve Uhlig, Rob Sherwood, Xin Huang. **Towards Hardware Accelerated Software Routers.** In Proceedings of *ACM CoNEXT Student Workshop*. Philadelphia, PA, USA. November 2010.

### Posters

Nadi Sarrar, Stefan Schmid, Steve Uhlig, Anja Feldmann. **Impact of FIB Aggregation on Traffic Offloading.** USENIX Symposium on Networked Systems Design and Implementation (NSDI), poster session. San Jose, CA, USA. April 2012.

# Contents

# 1 Introduction

The Internet began as a small academic network connecting a few universities and enabled a small group of researchers to exchange messages over it [79]. Since then, the Internet has evolved to a worldwide mass platform for global communication with billions of people participating [72]. Without any doubts, the Internet has become one of the most critical infrastructures today, for businesses as well as our daily private lives.

There are many stories to be told on why the Internet is as successful as it is. Some of those stories build on our basic needs to socially interact with people, some others see a reason in the desire of businesses to compete on a global market without being physically present everywhere in the world. From a technical perspective, the success story of the Internet, and the fact that it was able to grow to such a massive scale while maintaining broad robustness, stems from the early choice of following a de-centralized design principle [58]. Local failures in general do not disrupt the operations in other parts of the Internet, and there is no single knob that can turn off the entire Internet.

Indeed, the Internet is a network of networks, which are also called Autonomous Systems (ASes). An AS is a distinct administrative domain comprising of a single or multiple sub-networks. Some ASes are Internet Service Providers (ISPs), enabling homes, enterprises, as well as other ASes to connect to the Internet. Other ASes are content providers, such as Google or Facebook, and again others are content deliverers, such as Akamai or Limelight, to name a few. There is a wide range of business types of ASes in the Internet, with each having their own policies, demands, and expectations as a network in the Internet.

In addition, most pairs of ASes do not have a direct link for data transfer between each other. To achieve full Internet-wide connectivity, meaning that each host in any AS can communicate with each other host in any other AS, some better connected ASes need to transit traffic in service of other ASes. As a result, ASes need not only to establish a peering link and exchange traffic between one another. ASes are also required to exchange reachability information such that their peering ASes know which other networks they can reach through them. To automatically exchange such reachability information a suite of routing protocols has been standardized and implemented, with the Border Gateway Protocol (BGP) [122] being the routing protocol used between ASes [58].

Internet routers are the devices that speak BGP and forward data traffic toward their destination. To accomplish this, Internet routers need to maintain reachability information for all sub-networks which are present in the Internet. This database is called the Forwarding Information Base (FIB). The FIB is stored on very fast memory in Internet routers to support rapid destination lookup operations for each incoming data packet. The sheer amount of reachability information which is exchanged using BGP is growing rapidly, as the number of ASes, as well as the number of sub-networks inside of ASes, is increasing at a very high pace [69]. As of mid-2012, the full BGP routing table contains more than 430,000 entries, one for each sub-network in the Internet. As a consequence, a critical lifetime defining factor for any Internet router is the number of forwarding entries that can be stored in its FIB [96].

Today's commercial router vendors do not directly approach the challenge of scaling FIB memory in their router architectures ranging from small enterprise routers to high-end ISP core routers. In those architectures, FIB memory is simply increased with each new router model according to the vendors best estimate of the demand within the time-frame of the desired router lifetime [96]. Whenever the number of Internet sub-networks exceeds this fixed number of available FIB entries, a router cannot be used anymore for routing according to the full BGP routing table, although it might still be capable of handling the traffic workload.

**Leveraging router workload properties.** In this thesis, we revisit the architecture of Internet routers and make the following first key observation: *From an Internet traffic perspective, a router design which is scalable in terms of its FIB size should leverage re-occurring statistical properties in the Internet traffic.* Internet traffic is known to be consistent with Zipf's law at various levels of traffic aggregation [49, 157, 146, 27, 53]. We propose a traffic offloading scheme which, when applied as a design choice to build an Internet router, can reduce FIB memory requirements by about two orders of magnitude. This makes the size of the FIB memory largely irrelevant as a router's lifetime defining factor. Based on real Internet traffic traces, we show that our scheme meets the requirements of an ISP network.

Another technique to reduce FIB memory requirements is FIB aggregation [41], which describes methods to compress the contents of a FIB while maintaining forwarding

equivalence. We study the temporal and spatial locality properties of BGP routing updates and explore their potential to help improving upon online FIB aggregation algorithms which are already known in the literature [144, 161, 90]. We evaluate FIB aggregation techniques in the context of our traffic offloading scheme and explore the additional improvements that can be achieved when implementing FIB aggregation together with traffic offloading.

**Designing a switch-based router.** As a next step, with FIB memory requirements being significantly reduced, we raise the question of whether switches in principle can replace routers. Switches and routers differ mainly in two aspects. Firstly, switches have a much smaller amount of FIB space, which however is sufficient to implement a router based on our scalable router design principle. Secondly, switches offer only a limited set of features in their software compared to routers. We propose to implement the missing features by using programmable switches, for example those supporting the OpenFlow protocol [95] from the Software Defined Networking (SDN) space. Accordingly, we present and implement our SDN Router, which relies solely on commodity hardware (an OpenFlow-enabled switch and a PC), and open-source software supporting routing protocols like BGP [120, 61, 13].

We show that our SDN Router prototype complies with Internet standards by demonstrating that it can establish peering relationships through BGP and other routing protocols with commercial routers. We further confirm that our router meets the requirements of an ISP network. This lets us formulate our second key observation: *A high-performance router can be built following the split architecture paradigm by implementing the software-centric and hardware-centric tasks on separate devices.* The software-centric tasks include speaking routing protocols like BGP and maintaining the FIB contents and are generally referred to as an Internet router's *control plane*. The hardware-centric tasks are mainly focused on the packet forwarding operations including the per-packet destination lookups which involve the FIB and are generally referred to as the *data plane* of an Internet router.

Besides demonstrating a practically feasible way of scaling FIB memory in Internet routers, we believe that our SDN Router design is particularly attractive to the research community. Commercially available routers are highly integrated devices which keep the specifics of their software and hardware components fixed, closed, and largely unaccessible to researchers. This has forced researchers wanting to test the real-world performance and behavior of their own routing protocols to run their experiments in simulators or by using pure software routers. Both such approaches have their limitations in realism as they can not run at the same performance as a commercial hardware-centric Internet router, nor can they simulate the exact behavior of a such a device. In contrast, our SDN Router allows researchers to run experiments on a more realistic basis. The openness of the software components enable researchers to implement new routing protocols, or apply changes to existing ones, and deploy them in a university lab testbed, or even in a production network, to evaluate new ideas in a real environment.

**Internet structure and traffic.** With this thesis we want to raise the point that efficient and scalable router designs can be found by leveraging the re-occurring statistical properties of the router's data plane and control plane workloads. To this end, we believe that one should strive to keep an up-to-date and accurate picture of not only the Internet traffic properties, but also the topology of the Internet, the peering relationships between ASes, as well as the deployment status of the next version of the Internet Protocol (IP), when attempting to design a future-proof Internet router.

Accordingly, we rely on two extensive Internet traffic measurement studies. The first of these studies is based on Internet traffic traces collected at a large Internet Exchange Point (IXP), a central point in the Internet at which hundreds of ASes are present and establish peering relationships [8, 84]. However, the IXP ecosystem is understudied in the literature, the consequence being that most of the many peerings at an IXP are missing in published estimations of the Internet AS-level topology. The AS-level topology of the Internet, which is the map of the peering relationships between any two ASes, is complex and difficult to infer as public routing data is incomplete. Indeed, based on the IXP traffic traces, we uncover a large number of previously unknown peering links between ASes and with that, we revisit the common mental picture of the Internet AS-level structure. This brings us to our third key observation: *IXPs are a central topological component in the AS-level Internet and can be viewed as a microcosm of the Internet ecosystem itself, at which massive peering and traffic exchange is taking place.*

The second of our measurement studies focuses on the advances in the adoption of IPv6 [34], the next incarnation of the Internet protocol. The Internet address space supported by the current version of the Internet protocol, IPv4, is officially exhausted [68]. The size of the IP address space defines the number of addressable hosts in the Internet. IPv4 supports more than four billion host addresses which was considered to be sufficient at the time of its specification. Now that we are running out of IPv4 addresses there is an increasing need for Internet-wide deployment of the IPv6 protocol. IPv6 supports $2^{128}$ addresses, a number so large that it is not even remotely likely that we run into address space limitations again in the future. We study the growth of the IPv6 traffic at a coordinated global IPv6 deployment milestone event in June 2011, the World IPv6 Day [150]. On that day we observe a steep increase in the IPv6 traffic volume which sustained and continued growing months later.

**Summary.** We see a promising potential in workload-driven router designs. Based on our analysis of real Internet traffic and routing protocol traces we propose algorithms for traffic offloading and for FIB aggregation, which stimulate novel router designs, such as our SDN Router. We complement this thesis with passive measurements of IXP traffic to increase the accuracy of our understanding of the Internet topology, as well as IPv6 traffic, to follow up on the deployment of the IPv6 protocol, a soon-to-be major workload component for Internet routers.

## 1.1 Outline

Chapter 2 provides background information about the technical foundations and some of the related literature relevant to this thesis. This includes discussions about routing in the Internet, about the evolution of router and switch architectures, and about the structure of the AS-level Internet.

Chapter 3 presents the design and implementation of the SDN Router while focussing on the implied technical challenges. The next two chapters cover the algorithmic components of the SDN Router and evaluate the proposed ideas based on real Internet traffic data: Chapter 4 presents and evaluates Traffic-aware Flow Offloading (TFO), a traffic offloading algorithm which leverages the Zipf property in Internet traffic; Chapter 5 presents and evaluates Locality-aware FIB Aggregation (LFA), a FIB aggregation algorithm which leverages the spatial and temporal locality properties in BGP routing updates. Chapter 6 completes the feasibility study of the SDN Router by benchmarking current OpenFlow-enabled switches.

Chapter 7 presents the results of an extensive Internet traffic study based on Internet Exchange Point (IXP) data, which uncover a substantial number of previously unseen connections in the Internet AS-level topology. Lastly, Chapter 8 studies the progress of the adoption of the IPv6 protocol and finds plenty of evidence that a wide-spread use of IPv6 is to be expected in the near future.

Chapter 9 summarizes this thesis and discusses perspectives for future research in this area.

## 1.2 Contributions

This thesis makes the following contributions:

**SDN Router:** An SDN-based scalable router architecture built around commodity components and open-source software.

The SDN Router prototype implementation consists of an OpenFlow controller, called RouteVisor, that unites an open-source software router with an OpenFlow-enabled switch. Interoperability tests confirm that this router can successfully peer with commercial equipment and forward traffic.

**Traffic-aware Flow Offloading (TFO):** A traffic offloading algorithm that leverages Internet traffic properties.

The evaluation results of the TFO algorithm based on three real Internet traffic traces from diverse network locations show, that TFO can maintain a high traffic offloading ratio while keeping the costly communication overhead low.

With that, the TFO algorithm allows a complete implementation of the SDN Router.

**Locality-aware FIB Aggregation (LFA):** An online FIB aggregation algorithm that leverages the spatial and temporal locality in BGP routing updates.

The evaluation results of the LFA algorithm suggest that the locality properties in the BGP routing updates can be exploited for improving FIB aggregation algorithms. A further study shows, that FIB aggregation can have a positive impact on the traffic offloading performance of TFO in an SDN Router.

**OFLOPS:** A combined software/hardware-framework for benchmarking OpenFlow-enabled switches.

OFLOPS allows studying and unveiling some critical performance impairments of OpenFlow-enabled switches. The results show, that the packet forwarding latencies and the OpenFlow command completions times differ substantially across devices.

**Analysis of an IXP:** An in-depth analysis of one of the largest IXPs worldwide.

The IXP study is based on nine month's worth of sFlow records collected at an IXP in 2011. This study covers the IXP's ecosystem, the traffic that is exchanged between the IXP's members, and the peering relationships at that IXP. The results suggest a re-assessment of the mental picture that the research community has about the Internet ecosystem and the AS-level structure of the Internet.

**Analysis of the World IPv6 Day:** A close investigation of IPv6 traffic as observed at an IXP and a campus network.

At the World IPv6 Day– a coordinated global test-run of the new version of the Internet protocol – significant changes occurred: Native IPv6 traffic has almost doubled in volume and its application mix is getting closer to the typical application mix of IPv4 traffic. The results suggest that IPv6 is getting closer to becoming mainstream.

# 2
# Background

The routing infrastructure of the Internet is complex in nature. Throughout this thesis we touch many aspects of the evolved large-scale networks and systems which run today's Internet. In this chapter we summarize the necessary background information while putting it into context.

We begin with a walk through the routing control plane of the Internet and identify some of the challenges of the Forwarding Information Bases (FIBs) in Internet routers. We then discuss and compare router and switch architectures and observe similarities.

We continue with a discussion of a new software control interface into networks as proposed by the Software Defined Networking (SDN) concept. We introduce OpenFlow, one instance of an SDN protocol, which adds a programming interface to switches for remote control and monitoring of their data plane.

We then zoom out and take a look at the AS-level structure of the Internet and introduce the different types of peering links between networks in the Internet.

## 2.1 Routing in the Internet

From a coarse-grained routing perspective the Internet is a collection of autonomous systems (ASes), with some such ASes being connected for traffic exchange. From a finer-grained routing viewpoint those ASes each consist of a number of network

segments themselves, to which they provide AS-local as well as Internet-wide reachability. Accordingly, routers need to be concerned about two kinds of routing tasks: *(i)* inside an AS and *(ii)* between ASes, also referred to as intra-AS and inter-AS routing, respectively.

An intra-AS routing protocol typically follows the idea of finding the shortest paths between any two points in a network. There are limited concerns with respect to policy enforcement or confidentiality in terms of the topological details of the network. The main goal of intra-AS routing lies in making most efficient use of the overall available network capacity by minimizing the path lengths between any two points A and B in the network. On the contrary, an inter-AS routing protocol needs to allow the implementation of strict business policies, such that an ISP can dictate their routers to use cheapest paths over shortest paths, to save on expenses for transit traffic. Also, inter-AS routing protocols must not disclose the topology of an ISP's network, as topological details are generally considered a business secret.

The differences in the objectives of intra-AS and inter-AS routing require their own principles to take routing decisions and their own protocols for exchanging IP prefix-based reachability information between routers. Today's most common intra-AS routing protocols are Open Shortest Path First (OSPF) [97] and Intermediate System to Intermediate System (IS-IS) [111]. In the Internet there is only a single inter-AS routing protocol: the Border Gateway Protocol (BGP) [122].

Internet routers typically run multiple routing protocols, *e.g.*, combinations of BGP, OSPF, and/or IS-IS. Each routing protocol maintains a database comprising reachability information with the best routes to each reachable sub-network (IP prefix), called the Routing Information Base (RIB). These RIBs are then merged to compose the Forwarding Information Base (FIB). The FIB contains for each reachable sub-network the next-hop router toward which the corresponding traffic is to be sent.

Today, a complete BGP RIB with all announced sub-networks in the Internet contains more than 430,000 entries [69]. OSPF and IS-IS RIBs typically contain entries only for the network segments within an AS, a much smaller number compared to the size of the Internet-wide BGP RIB. Nevertheless, the resulting FIB contains more than 430,000 entries. This number is expected to further increase as the Internet is still growing and IPv6 is starting to be used, giving rise to two different kinds of concerns: *(i)* FIB size [96], as routers need to have sufficient memory to maintain a full copy of the FIB; and *(ii)* routing dynamics [85, 148, 82], as routing updates need to be incorporated quickly into the FIB.

Given that BGP prefixes are typically globally visible in the Internet, any change or update as well as any link failure can lead to routing updates. Today, the number of routing updates roughly corresponds to a few updates per second with peak rates of up to a few thousands. However, every update might require the re-computation of the RIB and a modification of some of the FIB entries. Therefore, it is crucial for Internet routers to have enough computing and memory resources to be able to keep

up with the rising quantity of routing information as well as their dynamics in the routing control plane of the Internet.

## 2.2 Routers and switches

Now that we have covered the basics of routing in the Internet and pointed out the inherent challenges that are relevant to this thesis in Section 2.1, we now briefly review router and switch architectures to highlight their main differences. In a nutshell, routers and switches differ in their architecture, hardware capabilities, and software features.

### 2.2.1 Router architectures

Internet routers need to support *control plane* (routing) as well as *data plane* (forwarding) operations.

The control plane tasks include running the needed routing protocols (see Section 2.1), maintaining peering relationships with other routers, maintaining routing tables (RIBs) for each routing protocol and peering router, computing a forwarding table (FIB) based on the RIBs, and providing administrative interfaces for router configuration and management. For this, the control plane requires a lot of software and significant computing resources such as processing power and memory capacity.

On the other hand, the focus of the data plane is on high-performance packet processing and forwarding. For example, on a fully loaded 10GBit/s link with packets of 1,000 bytes in size, one packet needs to be handled every 0.8µs. Hence, the data plane operations are challenging as performance is key. In addition, routers typically support a range of different network link technologies, such as Ethernet and optical interconnects such as SONET/SDH [116]. For this, a router needs to manage and translate traffic between a number of network interfaces and buffer packets if an output link is congested. Furthermore, a router needs to manage access to network resources through Access Control Lists (ACLs). However, the main concern of this thesis lies in a router's tasks to perform the destination lookup operation involving the FIB on every incoming packet, as well as to perform the according actions, such as re-writing the layer-2 (MAC) addresses of the packet. All this suggests an implementation of the data plane using dedicated hardware whenever a very high, reliable, and deterministic packet forwarding performance is typically required in enterprise and carrier networks.

When studying the evolution of router architectures one comes across quiet a number of generations of router designs. The first generation of routers was very similar to today's PCs. Shared memory or a bus was used for data transfer to interconnect the Network Interface Cards (NICs) and most of the packet processing tasks were done

by a central processor (CPU) [9]. This design however was finally abandoned mainly because it could not cope with the increasing requirements in throughput capacity which came with the rising data rates in the Internet.

The next generation of routers relied on line cards [9] which augmented the capabilities of NICs by increasing their processing power and memory size. The memory on the line cards was used to store forwarding information, the FIB. This architecture is getting closer to today's router designs, the main difference being that it still relies on a bus or shared memory to move packets from one network interfaces to another.

Finally, the bus for interconnecting network interfaces was replaced by a switching fabric [9] which enabled terabit routers. In contrast to the bus-driven architecture, the switching fabric enabled multiple pairs of network interfaces to simultaneously exchange data, and that at very high speeds without involving the CPU. Practically all commercial routers, enterprise-grade and above, rely on such an architecture [136].

Given the high data rates of current network link technologies, router line cards rely on fast, specialized technologies for maximum FIB lookup performance, for example a combination of SRAM, RLDRAM, TCAM, and specialized ASICs [149]. The size of a routers FIB memory is generally considered a major cost component of a router, and when too small, a lifetime defining factor, as Internet routers need to be able to work with the Internet-wide BGP FIB (see Section 2.1).

### 2.2.2 Switch architectures

Routers and switches conceptually have strong similarities as both their main objectives are to forward packets according to a set of rules at a very high speed. Consequently, switches also rely on a switching fabric for high performance packet forwarding. Switches, originally developed to be pure layer-2 devices, today even have limited support for layer-3 protocols.

One of the main differences between switches and routers can be found in their software. As switches by design operate on layer-2, the control software of a switch is optimized for layer-2 protocols like the Spanning Tree Protocol (STP), or its accelerated variant the Rapid Spanning Tree Protocol (RSTP), to avoid forwarding loops in the switched network. Another popular layer-2 feature is IEEE 802.11q Virtual LAN (VLAN), which allows the creation of multiple *virtual* LANs that transparently share a single physical network.

Because of the fact that switches are layer-2 devices by design, their FIB memory technologies are tailored to layer-2 protocols. Routers, being layer-3 devices that need to map incoming IP packets to their destination sub-network, are required to implement FIB lookups according to the longest prefix matching algorithm. Sub-networks are defined by IP address prefixes of various lengths. The longest prefix matching algorithm returns for a given IP packet the sub-network with the longest

prefix that entirely matches to the first bits of the destination IP address of the packet. This is a complex operation as matching an IP address to one out of many IP prefixes requires comparisons of varying numbers of bits according to the prefix lengths and it must always return only the longest matching prefix.

FIB lookups on switches are simpler as the layer-2 identifiers are all fixed in length (MAC addresses, VLAN identifiers, etc.). Hence, the FIB memory in switches typically is specifically designed for fast fixed-length comparisons and as such it is not capable of performing longest prefix matching. That said, today's switches do offer some layer-3 capabilities, and for those they come with Ternary Content-Addressable Memory (TCAM) which also allows fast prefix lookups. TCAM in switches is typically used for Access Control Lists (ACLs), for traffic shaping, and also for routing. TCAM is also found on router line cards to implement the same functionality, except for routing.

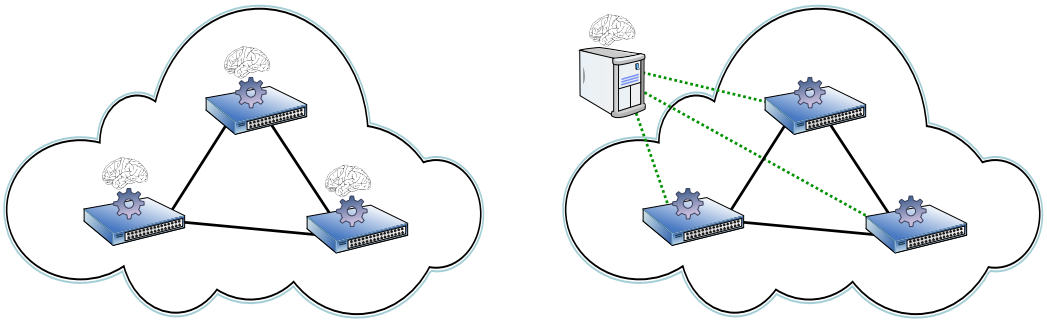
TCAM is a very fast but expensive [141] and power-hungry [81, 156] technology which provides space only for a limited number of FIB entries in today's switches and routers, typically in the order of a few thousands of entries, two orders of magnitude below the required number of entries needed for full Internet-wide routing (see Section 2.1). As we will later see, our SDN Router, with its techniques to significantly reduce FIB memory requirements, can utilize a FIB with just a few thousands of entries, *e.g.*, a switch TCAM, for full Internet-wide routing.

Another important difference between routers and switches lies in the support for different link technologies. While routers support a number of link technologies (see Section 2.2.1), switches are limited to Ethernet. However, we note that Ethernet is becoming the technology of choice even for wide-area links in ISP networks due to its price-performance benefits [51].

## 2.3 Software Defined Networking and OpenFlow

Software Defined Networking [137], or SDN in short, is about making networks programmable. The desire for network programmability is rooted in the academic community as well as in the networking industry [95]. The academic community on the one hand strives toward testing new ideas on real devices and in real networks, which is problematic with the current proprietary equipment whose software cannot be modified. The industrial community of network operators on the other hand sees big advantages in network programmability, as it would enable them to tune the behavior of their networks to their actual needs.

A high-level illustration of the SDN paradigm in comparison to a typical network is shown in Figure 2.1. The network diagram shown in Figure 2.1(a) represents a typical network architecture as of today. The three network devices, which can be either routers or switches, accomplish two main tasks. They first need to implement



(a) A common network consisting of three routers or switches.

(b) An SDN-based network consisting of three programmable switches and a controller running on a PC.

**Figure 2.1:** A common network and an SDN-based network illustrated side-by-side.

the hardware-centric task of forwarding packets, called the *data plane* (symbolized by the gear wheel). They further need to implement the software-centric, complex control logic, called the *control plane* (symbolized by the brain), to instruct the data plane to forward packets according to a set of forwarding rules.

In contrast, Figure 2.1(b) illustrates a network which realizes the concept of SDN. In SDN, the main thing that a packet forwarding device like a switch needs to provide is a high-performance data plane. The complex, software-centric control plane logic, the *brain*, is moved to a separate entity, called the network controller (illustrated by a PC). This controller decides for each traffic flow which path to take through the network. The controller then passes its routing decisions in the form of forwarding rules on to the switches, which in turn forward the incoming traffic based on these forwarding rules. Hence, the controller and the switches need means to communicate, the dotted lines in the figure illustrate the respective communication channels.

The essential element in an SDN-based network architecture is the communication channel. This communication channel is to be understood as an API into the forwarding behavior of switches. As such, this API needs to provide an appropriate set of primitives to give network control applications the necessary set of tools for remotely controlling switches. More precisely, this API should strive for maximum flexibility in order to support a wide range of network applications.

OpenFlow [95] is the prominent SDN communication channel protocol, standardized by the Open Networking Foundation (ONF) [109]. The OpenFlow protocol includes basic operations such as *modify-forwarding-table* and *get-statistics*, which allows the controller to modify the set of forwarding rules of a switch and obtain traffic statistics from a switch, respectively. The set of packets that match a given forwarding rule are collectively referred to as a *flow*. A flow in OpenFlow 1.0 [110] can be defined according to any combination of layer-2 (MAC), layer-3 (IPv4), and layer-4 (TCP, UDP) packet header fields, some of which allow partial wildcarding of bits. The next

milestone in the OpenFlow protocol specification, version 1.2, adds support for IPv6 and MPLS, amongst others, and provides a much more advanced forwarding table abstraction.

In this thesis we use OpenFlow-enabled switches to implement our SDN Router prototype. The switches that we use support the OpenFlow 1.0 protocol which allows a flow to be defined as an IP prefix. Besides one missing feature (TTL decrementation, added since OpenFlow 1.1), the protocol meets all requirements to build a router from a switch and a PC according to our SDN Router design.

## 2.4 Internet AS-level topology

Our mental picture of the AS-level topology that describes the high-level structure of the Internet has changed over the years. The differences in the understanding of the Internet topology stem not only from actual changes in the peering relationships between ASes and from changes in the ASes which serve the main content, but also from improvements in our methodologies of estimating the AS-level topology.

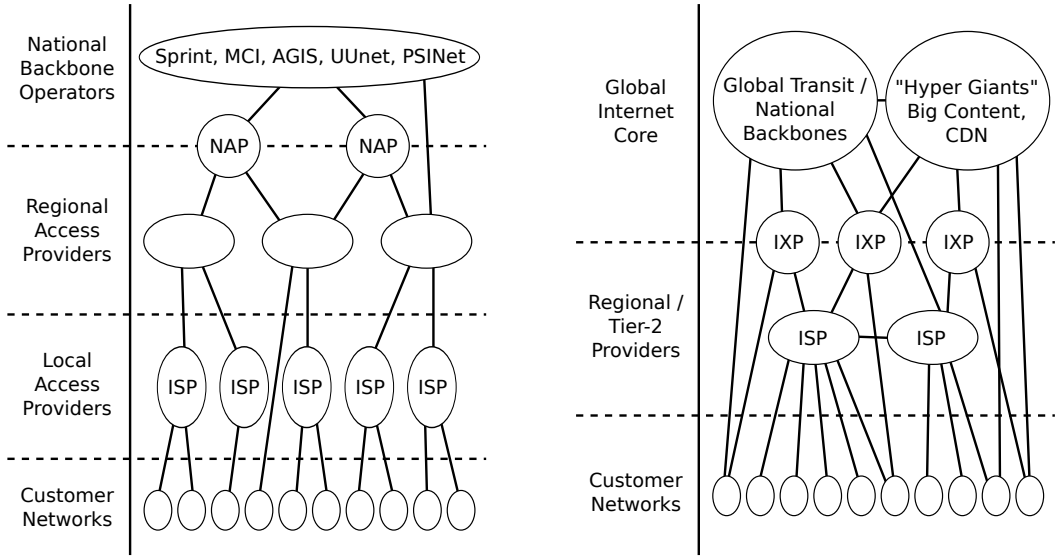
### 2.4.1 Peering links between ASes and their hierarchy

There are two main concepts which need to be understood when discussing the AS-level topology of the Internet: *(i)* The classification of peering links between ASes into peer-peer (P-P) links and customer-provider (C-P) links; and *(ii)* the classification of ASes into their hierarchical groups of tier-1, tier-2, and leaf networks.

Peering links between ASes can either be established between equally treated peers (P-P) or between customer and provider ASes (C-P). P-P links are typically settlement free and established between ASes of roughly the same size[1]. Peers in a P-P link agree to provide free connectivity to their own sub-networks as well as to those of their customer ASes. In contrast, in a C-P relationship the customer obtains full Internet-wide connectivity through the provider. In other words, the provider in a C-P relationship provides transit to its customer, meaning that the provider agrees to accept and forward traffic to all destinations in the Internet. This includes destinations for which the provider itself has to rely on another provider.

This brings us to the idea of hierarchy in the peering map of the Internet. The biggest and best connected ASes are members of the group of tier-1 ASes. All members of the tier-1 group agree to establish P-P links to all other tier-1 ASes. There is no C-P link in the Internet in which a tier-1 AS is the customer. This means that a tier-1

[1]The size of an AS is not clearly defined and subject to negotiations between ASes. It can be any combination of number of connected sub-networks, number of customer ASes, geographical reach, traffic volume, popularity of served content, etc.




(a) Traditional Internet structure [65].

(b) Internet structure as of 2010 [84].

**Figure 2.2:** Evolution of the Internet AS-level structure.

AS has Internet-wide reachability through C-P links, in which the tier-1 AS is the provider, and through P-P links.

Next in the hierarchy is the group of the tier-2 ASes which by definition need at least one upstream provider in a C-P relationship for Internet-wide connectivity. Tier-2 networks also have their own transit customers through C-P relationships in which they appear as providers. In short, tier-2 ASes have providers as well as customers. In order to reduce the amount of transit traffic to their providers as well as the cost which is associated with that, tier-2 networks establish P-P relationships with other tier-2's and leaf ASes. At the bottom of the hierarchy are the leaf networks. A leaf AS has providers (through C-P links) and peers (via P-P links), similar to tier-2's, but no customer ASes, contrary to tier-2's[2].

### 2.4.2 Evolution of the mental model of the Internet structure

Now that we have covered the basics of the tier-level categorization of the ASes in the Internet along with the present differences in the peering relationships between them, we turn to the evolution of the mental picture representing the logical structure of the AS-level Internet. The textbook version of the traditional logical structure of the Internet [65] exhibits a somewhat clean hierarchy, see Figure 2.2(a): A few large

[2]Leaf ASes correspond to the notion of tier-3 ASes – a term used in some of the related literature. The distinction of tier-2's and tier-3's however is not always consistently defined. Hence we opt to use a different term (leaf) and provide a clear and comprehensive definition.

national backbone operators form the *core* of the Internet. Below them are a number of Network Access Points (NAPs), which are peering points for the regional carrier networks and the Internet core ASes. At the next level in the logical hierarchy, one level below the regional access providers, we have the local ISPs. The customer networks are at the bottom end of that hierarchy. Mapping our terminology of tiers onto that structure, the national backbone operators are the tier-1's, the customer networks are the leaf's, and both hierarchical groups of access providers fall into the category of tier-2 networks.

An extensive study of Internet traffic published in 2010 [84] has revealed significant changes to this mental picture, see Figure 2.2(b). First, the logical *core* of the Internet is no longer occupied solely by large tier-1 networks, but also by big Content Delivery Networks (CDNs) and big content providers. From a routing perspective, these *Hyper Giants* actually fall into the category of leaf networks as they do not have any transit customers. However, these networks tend to have a significant number of direct peering links to ISP networks, thus bypassing the Internet core. In addition, they are reportedly responsible for most of the Internet inter-domain traffic, most of which is never seen by the core [84]. The second difference is the introduction of Internet Exchange Points (IXPs) into this picture, replacing the NAPs. As successors of NAPs, these IXPs operate an advanced switched infrastructure to provide a common peering platform for all kinds of ASes. The result being a flattening of the mental model of the logical Internet topology with a larger number of peering relationships and a significant fraction of all Internet traffic bypassing the core.

*The hosts executing the Border Gateway Protocol need not be routers.*

K. Lougheed and Y. Rekhter, A Border Gateway Protocol (BGP), IETF RFC 1105, June 1990.

# 3 SDN Router

In this chapter we present the design of our SDN Router followed by a description of our prototype implementation. The SDN Router concept is core to this thesis as it demonstrates the feasibility of our idea to build a router according to the split architecture paradigm. Our SDN Router prototype uses commodity PC hardware running an open-source software router for the routing control plane. This software router is coupled with a programmable switch to offload most of the traffic to the switch. We perform interoperability tests and confirm that it is standards-compliant as it can co-operate together with current commercial off-the-shelf routers. Our prototype fills the gap between pure software routers and commercial high-end routers. However, as the SDN Router concept is a generic alternative to current router architectures, we expect it to have an impact on future high-end equipment. In Chapter 4 we show that our prototype implementation is capable of handling full IP routing tables and that it meets the traffic requirements of an existing carrier aggregation network.

## 3.1 Introduction

In principle, routers are very simple packet processing devices. The main task of a router is to determine for incoming packets the appropriate output interface based on the content of its forwarding table (FIB), as introduced in Chapter 2.2.1. However, this simplicity is deceiving as *(i)* computing the routing table is a complex and computationally expensive task, *(ii)* performing the longest prefix match at line rate is non-trivial, and *(iii)* achieving multiple terabit throughput is challenging. The number of vendors offering high-performance routers is very limited and they tightly

integrate software and hardware components. Hence, the current building practice of high-end routers couples software and hardware innovation cycles and requires a lot of software which is specific to the hardware vendor.

The design of our SDN Router on the other hand is based on open components and allows researchers (but not only researchers) to implement, test, and deploy in production networks their own routing protocols, customizations to existing protocols, and enhancements in a router's set of features. We leverage the decoupling of the traditional tasks of a router, *i.e.*, maintaining up-to-date routing information as well as forwarding packets. We propose an alternative router design that couples a commodity PC running an open-source software router with lower-cost programmable switching hardware by delegating most of the packet forwarding workload to the switch.

Open-source routing software [120, 43, 73, 4] is already deployed within enterprise, ISP, and IXP networks for specialized tasks. Its quality is approaching what is commonly referred to as *carrier grade*. However, PC based routers with open-source routing software do not offer the throughput, port density, or deterministic packet switching performance needed for ISPs' purposes. On the other hand, we observe that Ethernet switches often provide layer-3 functionalities and contain components, *e.g.*, TCAMs, for performing rapid per-packet destination lookups according to the longest prefix matching algorithm (see Chapter 2.2.2) at line rate and they can offer multiple terabit throughput.

However, such switches usually have very limited FIB memory, typically in the order of a few thousands, which is not enough to store a complete Internet routing table that contains more than 430,000 entries today. We propose to leverage the switch FIB for handling the most important prefixes only, according to their traffic volume. We later show that we can indeed capture most of the traffic with just a few thousand prefixes (see Chapter 4) by benefiting from the re-occurring Zipf-like property in Internet traffic.

In summary, we are using the switch as a flexible forwarding engine for high-performance forwarding for most of the traffic. The PC acts as a route controller and furthermore handles the traffic that is chosen not to be forwarded by the switch.

In this chapter, we present our SDN Router concept (Section 3.2) and describe our prototype implementation (Section 3.3). Our prototype is based on an OpenFlow-enabled switch [95] and a Linux-PC running the open-source routing suite Quagga [120]. We present RouteVisor (Section 3.3.1), an OpenFlow controller, that serves as the glue logic combining the software router Quagga and the switch, and perform interoperability tests with existing routers in Section 3.3.2. We the discuss related work in Section 3.4 and summarize in Section 3.5.

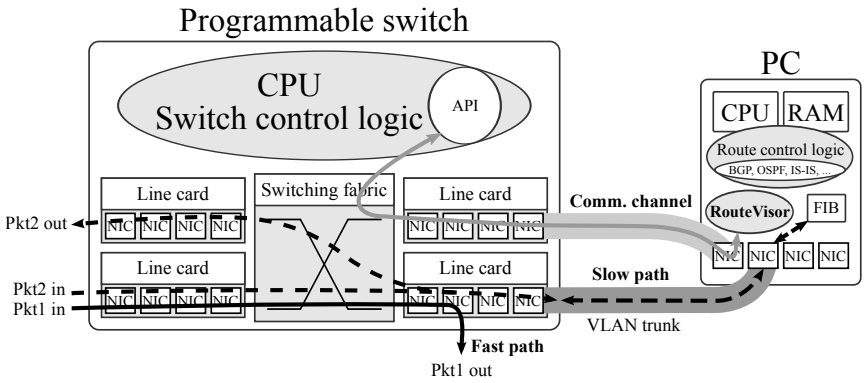



**Figure 3.1:** SDN Router concept with slow path (dashed black arrow), fast path (solid black arrow), and communication channel (grey arrow).

## 3.2 Concept

Our SDN Router concept builds upon the idea of combining open-source routing software running on commodity PC hardware with a programmable Ethernet switch for high-performance packet forwarding.

Figure 3.1 illustrates our proposed router design concept. The PC runs the route controller, which can be any software router. In addition, the PC runs a software agent, called RouteVisor, whose main task it is to transparently glue together the software router and the programmable switch. The programmable switch is the only link to the outside world, *i.e.*, peering routers are connected to switch ports. RouteVisor makes sure that routing protocol messages, *e.g.*, BGP updates, which arrive at the switch, are delivered to the software router. Also, RouteVisor monitors changes to the FIB on the PC which come from the software router and updates the FIB on the programmable switch correspondingly. Hence, the switch can forward packets without any further involvement of the PC, along the *fast path* of the SDN Router.

However, a switch typically does not have enough FIB memory to store all entries needed for forwarding traffic based on the complete Internet BGP routing table. Accordingly, RouteVisor installs, or *offloads*, only those forwarding entries (IP prefixes) which are ranked high in terms of their traffic share. Because this ranking is very dynamic, it is challenging to determine which prefixes to offload when. We will address this challenge and propose an algorithm, called Traffic-aware Flow Offloading (TFO), in Chapter 4.

For packets arriving at the switch that do not match any of the offloaded prefixes, we introduce the *slow path* of SDN Router. Slow path packets are tagged according to their input port, for example realized by using VLAN tags, and then sent to the PC where the complete FIB is available for destination lookups. The PC then modifies the packet's tag according to the determined output interface and sends it back to

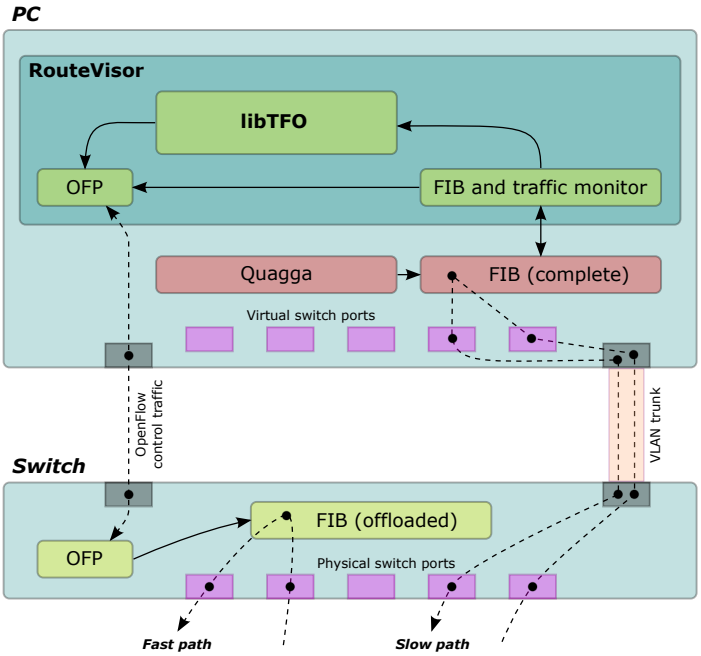



**Figure 3.2:** SDN Router prototype implementation based on a Linux PC running Quagga and RouteVisor (top), and an OpenFlow-enabled switch (bottom).

the switch, which then pops the tag and sends the packet out the output interface corresponding to the tag.

In summary, the SDN Router decouples the routing control plane from the forwarding engine, which is contrary to conventional router architectures. A switch is used for fast path forwarding of most of the traffic, and a PC is used for slow path forwarding of the remaining traffic and for running the route control logic. With the help of a traffic offloading algorithm, the SDN Router aims at offloading most of the traffic off the PC and onto to the fast path – the switch.

## 3.3 Prototype

In this section we describe our SDN Router prototype. This prototype relies on an OpenFlow-enabled switch (see Section 2.3) as the fast path forwarding engine. We use a PC running Linux and the open-source software router Quagga. To connect Quagga with the OpenFlow-enabled switch we implement RouteVisor according to the SDN Router concept.

### 3.3.1 RouteVisor

RouteVisor is an OpenFlow controller on the one hand, and a transparent FIB proxy on the other hand. In Figure 3.2 we provide a diagram of our SDN Router prototype, which illustrates the main software components inside RouteVisor.

To act as an OpenFlow controller, RouteVisor implements an OpenFlow protocol endpoint (*OFP*), which connects to the OFP on the switch. RouteVisor also establishes a VLAN trunk between switch and PC for slow path traffic. For this, RouteVisor creates VLAN network interfaces which are virtual counterparts of the physical switch ports. Quagga is configured to directly operate on these virtual interfaces. RouteVisor monitors all updates by Quagga to the Linux FIB and translates them into OpenFlow FIB update commands, in case they affect offloaded FIB entries. In regular intervals, RouteVisor runs the traffic offloading algorithm TFO which then may trigger updates to the set of offloaded FIB entries. We release a library implementation of TFO, called libTFO, as open-source software[1].

**VLAN glue.** We opt for a design in which we can use any software router without source code modifications, *e.g.*, XORP [61], BIRD [13], or Quagga [120]. To accomplish this, RouteVisor provides the software router with virtual interfaces that correspond to the physical switch interfaces. Contrary to the physical interfaces on the switch, the virtual interfaces represent router ports and therefore have IP addresses (router ports, layer-3) while the physical ones do not (switch ports, layer-2). RouteVisor installs static FIB entries into the switch to direct slow path packets with a corresponding tag through the VLAN trunk to the PC, and to ensure correct forwarding of tagged packets that return from the PC. On the PC, the Linux networking stack forwards packets across the VLAN interfaces.

RouteVisor makes sure that all packets that have to be handled by the software router, including routing updates, are forwarded to the PC. For this, RouteVisor installs a small number of static entries into the switch's FIB. Our RouteVisor implementation installs such entries to match routing protocol messages of BGP, OSPF, and IS-IS. Thus, any adjacent router connected to a switch interface can establish, *e.g.*, a BGP peering session, with Quagga. As a result, to the outside world this switch-PC combination acts as if it was a conventional IP router.

**FIB and traffic monitoring.** RouteVisor monitors changes to the PC's routing table done by Quagga via the Linux Netlink subsystem. Whenever a change to the Linux FIB takes place, RouteVisor is notified by receiving a Netlink message. In addition, RouteVisor maintains packet and bytes counters per FIB entry for both slow path, via counters in the Linux FIB, as well as fast path, through OpenFlow statistics queries. RouteVisor collects and maintains these statistics in regular intervals as

[1] libTFO is available under `http://www.fibium.org/`

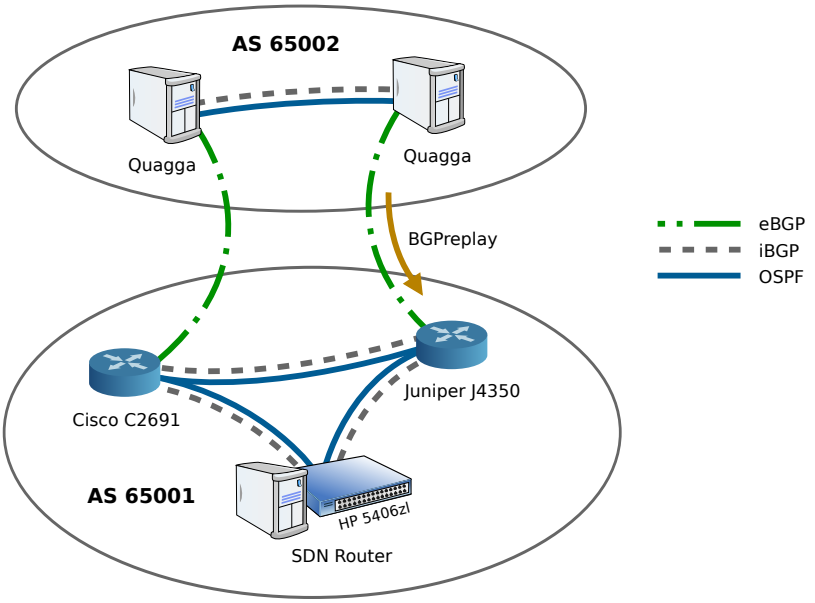



**Figure 3.3:** SDN Router interoperability test network topology.

required by the traffic offloading algorithm, *e.g.*, every 10s in our prototype with TFO.

**OpenFlow switch.** We have successfully tested RouteVisor with OpenFlow-enabled switches by HP (ProCurve 5406zl), NEC (IP8800/S3640), and Quanta (LB4G). For the conformance tests in the next section we rely on the HP switch. The HP switch has FIB memory for about 2,000 prefix-based FIB entries.

### 3.3.2 Interoperability tests

We now demonstrate that our prototype SDN Router can interoperate with Juniper and Cisco routers in a multi-protocol setup. We integrate our prototype into a network consisting of a Juniper router J4350, a Cisco router C2691, and two open-source software routers running on Linux PCs. The topology is shown in Figure 3.3. The two Linux routers belong to AS65002. These have eBGP sessions with the Juniper and Cisco routers. The Juniper and Cisco routers are connected to the RouteVisor-controlled HP switch and constitute an OSPF area within AS65001, which represents an ISP network. We inject BGP updates from one of the open-source routers in AS65002 via an eBGP session into the Juniper router, using BGPreplay [92], a tool that is able to re-inject BGP messages recorded by any of the publicly available BGP monitors, *e.g.*, RIS [125] or Routeviews [129]. All tests ended with the expected FIB contents in all routers and we did not experience any interruptions in any of the BGP and OSPF sessions.

## 3.4 Related work

Making IP forwarding fast and scalable has been and still is one of the major challenges that network equipment vendors face. Route caching was introduced in the late 1980s, for example with Cisco's *fast switching* [25]. These approaches however became unpopular because of the expensive handling of cache misses which became more and more difficult with the increase in data rates. Unlike previous approaches, our proposed scheme proactively populates the FIB cache and cache misses in general do not incur modifications to the cache. Further, unlike previous approaches, our FIB cache is at the level of variable length prefixes rather than on flows or prefixes of fixed length, and our solution makes the correct forwarding decision for all packets including those after routing changes that affect cached FIB entries.

Leveraging fast memory for forwarding is one of the central issues of scalable forwarding [149]. Numerous papers have proposed the use of TCAM for forwarding purposes [132, 81, 156]. Current TCAMs allow very efficient longest prefix matching but have limitations, *e.g.*, in their power consumption and their price. Router vendors nowadays rely on different types of memories on their line cards to ensure a high cost-performance ratio [149]. In SDN Router, we can leverage relatively small TCAMs in switches by using them to handle only the heavy hitters. Chapter 4 studies the associated trade-offs.

Another approach of building Internet routers on top of OpenFlow-enabled switches is RouteFlow [126], a BGP-based routing control platform. In contrast to SDN Router, the authors of RouteFlow do not approach the challenge of FIB size limitations in current switch architectures.

Other proposals to reduce FIB space requirements in routers rely on BGP configuration tricks. For example, ViAggre [11] delegates different subsets of the complete address space to routers, reducing the sizes of individual router FIBs by at least one order of magnitude, at the cost of a certain path stretch. A similar approach is *FIB suppression* as proposed by S-VA [121], where FIB sizes on edge routers are significantly reduced by suppressing certain routing entries from being installed by replacing them with a single default route to core routers. Similarly to these approaches, we make better use of the capabilities of existing network equipment. However, we do not require any re-configurations to routing protocols like BGP, but instead we propose a new way to leverage the capabilities of programmable switches to transform them into routers.

## 3.5 Summary

In this chapter we propose SDN Router, a novel Internet router design which couples a PC, running an open-source software router, with a programmable switch, *e.g.*, and

OpenFlow-enabled one. This concept requires software that virtually glues together the software router and the switch. Accordingly, we design and implement RouteVisor, which acts as an OpenFlow controller and also prepares an environment for the software router to transparently use the switch as its main forwarding engine.

Given that Ethernet switches typically do not have enough FIB memory to store a complete BGP Internet-wide routing table, SDN Router takes the approach of traffic offloading. Only the heavy hitters, *e.g.*, the FIB entries with the highest traffic volumes, are offloaded to the switch and handled via SDN Router's fast path. The remainder is handled by the PC, via the slow path.

The SDN Router concept raises an algorithmic challenge: A traffic offloading algorithm is required that determines which prefixes to offload at what time. We will address this challenge in Chapter 4.

# 4

# Leveraging Zipf's Law for Traffic Offloading

Inherent in the design of our SDN Router we have identified an essential challenge: selecting a set of top prefixes – the *heavy hitters* – over time. In this chapter we focus on this heavy hitter selection problem and begin with the observation, that Internet traffic has Zipf-like properties at multiple aggregation levels. These properties suggest the possibility of offloading most of the traffic from a complex controller (*e.g.*, the PC in our SDN Router) to a simple forwarder (*e.g.*, the switch in our SDN Router), by letting the forwarder handle a very limited set of prefixes; the *heavy hitters*.

As the volume of traffic of individual prefixes is highly dynamic, maintaining a reliable set of heavy hitters over time is challenging. This being the case we identify a trade-off in the heavy hitter selection problem: maximizing the offloading ratio while minimizing the number of changes to the set of heavy hitters over time, for a fixed size of the set. There is a certain cost associated with every such modification, for example in our SDN Router an OpenFlow operation needs to be executed on every change to the set of heavy hitters which can turn into a critical bottleneck (see Chapter 6).

We propose a heavy hitter selection strategy, called Traffic-aware Flow Offloading (TFO), that takes advantage of the properties of heavy hitters at different time scales. Based on real Internet traffic traces, we show that our strategy is able to offload most of the traffic while limiting the rate of modifications to the heavy hitter set, demonstrating the feasibility of our SDN Router design.

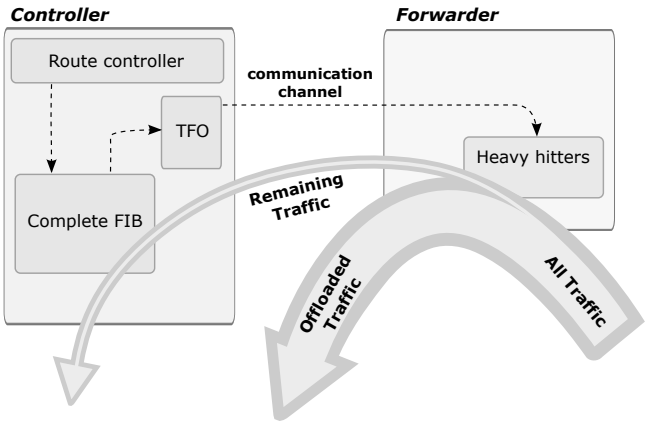



**Figure 4.1:** Traffic offloading system principle.

## 4.1 Introduction

A reoccurring property of Internet traffic at various levels of aggregation is its consistency with Zipf's law [49, 157, 146, 27, 53], *i.e.*, the amount of traffic per flow is consistent with a Zipf distribution. This implies that a small fraction of flows captures most of the traffic. A flow can be as fine as the traffic between a given IP source-destination pair or as coarse as all the traffic sent to a single sub-network, *i.e.*, to an IP prefix. However, heavy flows in the Internet are known to be volatile [147], making their selection for traffic engineering difficult [113].

We propose to utilize the Zipf properties of Internet traffic to offload most of the traffic from a complex controller to a simple forwarder, according to our SDN Router design. In this chapter however we consider a simplified abstraction of the SDN Router, referred to as the *traffic offloading system*. This abstraction contains all the elements of concern to this chapter and may suggest applications of our proposed traffic offloading scheme other than our SDN Router alone.

The principle behind such a traffic offloading system lies in bundling a controller, *e.g.*, a complex and flexible software-centric system, together with a forwarder, *e.g.*, a high-performance hardware-centric packet forwarding device. This approach follows the split architecture idea: Most of the control plane load should be handled by the software-oriented part of the system, while most of the packet forwarding should be performed by a fast and simple device.

Figure 4.1 illustrates the principle controller-forwarder system architecture. The *controller* acts as the routing controller, hence its main responsibility is to take all routing decisions for example through speaking a routing protocol. The controller takes advantage of a *forwarder*, a device optimized for high-performance packet forwarding, by offloading most of the traffic, that is the traffic captured by the heavy hitters, to the forwarder. Through a communication channel between controller and

forwarder, the controller installs and updates the set of heavy hitters in the forwarder to control which flows are offloaded and thus handled by the forwarder. In this chapter we investigate the feasibility of such an offloading system based on real Internet traffic traces.

We first show that in principle, the achievable offloading gain is in the order of more than 90 % when offloading the traffic contributed by 0.2 % of the flows. However, the volatility in heavy hitters incurs significant communication overhead, which is contrary to our design goals of keeping the forwarder simple. To address this challenge, we propose our heavy hitter selection strategy called Traffic-aware Flow Offloading (TFO), which relies on traffic statistics over multiple time scales. Previous work [113, 147] has identified the volatility of heavy hitters in Internet traffic as a challenge for traffic engineering, which also applies to our controller-forwarder context. In addition, we require that the benefit of introducing a forwarder is not counterbalanced by the complexity of the communication between the controller and the forwarder. Therefore, TFO trades off the overhead of modifying the set of heavy hitters against the expected increase in offloading gain.

Our results show that TFO is able to offload most of the traffic to the forwarder while keeping the number of modifications to the set of heavy hitters low, contrary to traditional route caching strategies [77]. This enables us to discuss both the benefits as well as challenges that Internet traffic properties pose for router designs. The contributions of this chapter are the following:

**Traffic offloading:** Based on traffic traces from a large European ISP, a large European IXP, and a transcontinental link we show the potential of traffic offloading and identify the challenges of heavy hitter selection strategies.

**Traffic-aware Flow Offloading (TFO):** We propose TFO, a heavy hitter selection strategy that leverages Zipf and the associated stability properties of heavy hitters across different time scales.

**TFO and traditional caching:** We evaluate TFO and compare it to traditional caching strategies. We observe a cross-over point. TFO outperforms traditional caching when the number of heavy hitters and the communication overhead have to be limited. However, when these constrains are relaxed traditional caching strategies can outperform TFO.

In addition, we release a library implementation of TFO as free software to the community [1], which we have used to implement our SDN Router (see Chapter 3). The focus of this chapter however lies in understanding the potential of traffic offloading, which we believe is crucial not only for our example prototype but also for a wide range of high-capacity, programmable packet forwarding systems.

[1] TFO software implementation: `http://www.fibium.org/`

| | ISP | IXP | Backbone |
|---|---|---|---|
| Network | residential | hundreds of ASes | transcontinental link |
| Speed | 1 Gbps | 1.5 Tbps | 150 Mbps |
| Duration | 2 days | 4 days | 3.5 days |
| Sampling | none | $1{:}2^{14}$ | none |

**Table 4.1:** Summary of our datasets.

The rest of the chapter is structured as follows. In Section 4.2 we revisit properties of heavy hitters including traffic share and churn. In Section 4.3 we first explore the limitations of traditional caching strategies and then propose and evaluate TFO. We discuss related work as well as future perspectives of our work in Section 4.4. We summarize in Section 4.5.

## 4.2 Feasibility study

In this section we perform feasibility experiments based on Internet traffic traces to understand the fundamental challenges and quantify the potential benefits of traffic offloading. We start by revisiting the properties of Internet traffic as seen from three different vantage points in the network. Then, we run evaluations under two optimistic assumptions: (i) future traffic is known and (ii) heavy hitter modifications are instantaneous. The results suggest a need to reduce the churn in the selected heavy hitters in order to limit the communication overhead. They also confirm previous results [147] which show that the variability of heavy hitters differs across ranks. Both observations shall be taken into account when designing a heavy hitter selection strategy such as TFO.

### 4.2.1 Datasets

We base our study on traffic traces from three different network locations (Table 4.1):

**Residential ISP:** The ISP trace relies on passive, anonymized packet-level observations of residential DSL connections collected at an aggregation point within a large European ISP. Overall, the ISP has roughly 10 million (4 %) of the 251 million worldwide broadband subscribers [105]. The monitor operated at a broadband access router connecting customers to the ISP's backbone. The trace started in August 2009 and covers 48 hours.

**Transcontinental link:** The traffic trace from a transcontinental link between Japan and USA also consist of passive, anonymized packet-level observations. The

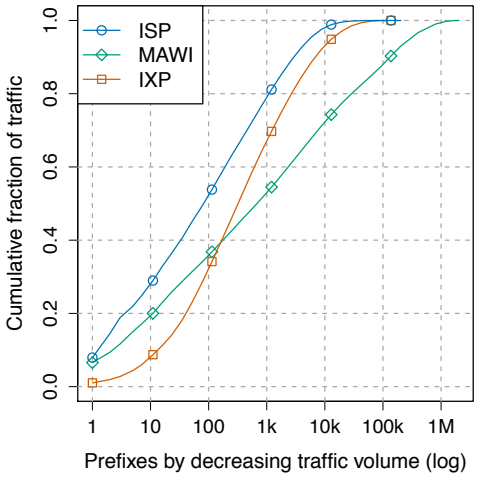



**Figure 4.2:** Distribution of the fractions of traffic per prefix.

trace was collected in April 2010, covers 3.5 days, and is publicly available from the MAWI working group of the WIDE project [24].

**European IXP:** In contrast, the trace from a major European IXP consists of anonymized sFlow records which were captured in April 2011. The sFlow probes were configured to export sampled packets at a sampling ratio of $1{:}2^{14}$. Due to the sampling, some underestimation of small flows is possible, and therefore there may be some bias towards the heavy hitters [28, 46].

While anonymizing the ISP and IXP traces, we map each destination IP address to its longest-matching prefix based on publicly available BGP data. The same technique could not be applied to the trace from the transcontinental link, but fortunately the anonymization uses consistent scrambling of /24 prefixes [24].

These three datasets allow us to look at the problem of offloading from different perspectives: network location, network size, and time. This gives us confidence that our results can be generalized beyond these samples. Moreover, we note that our proposal exploits traffic properties that have already been observed in the past and are not expected to change, as the popularity of Internet flows is expected to retain its consistency with Zipf's law.

### 4.2.2 Consistency with Zipf's law

Since we aim at leveraging the Zipf properties of Internet traffic for traffic offloading, we first confirm that the three traces are consistent with Zipf's law.

Figure 4.2 shows the relative share of the total traffic as a function of the number of top destination prefixes. These results confirm that all three traces exhibit the Zipf

property – a limited number of prefixes account for a majority of the traffic. However, one difference between the traces lies in the number of heavy hitters whose traffic has to be aggregated to capture a desired fraction of the traffic. For example, the most popular 10 destination prefixes of the residential ISP trace and the transcontinental trace already capture more than 20 % of the total traffic, but less than 10 % in the case of the IXP. With 1,000 prefixes one can capture more than 50 % of the traffic. This holds for all three traces. For lower ranks the amount of traffic that each prefix is contributing decreases rapidly. The next 1,000 prefixes combined capture less than 5 % of the traffic. Another difference is that the transcontinental link sees traffic for a larger number of prefixes. However, the least popular half of the prefixes contribute less than 1 % of the traffic. This may be an artifact of the anonymization as described above and the resulting assumption of a flat /24 address space.

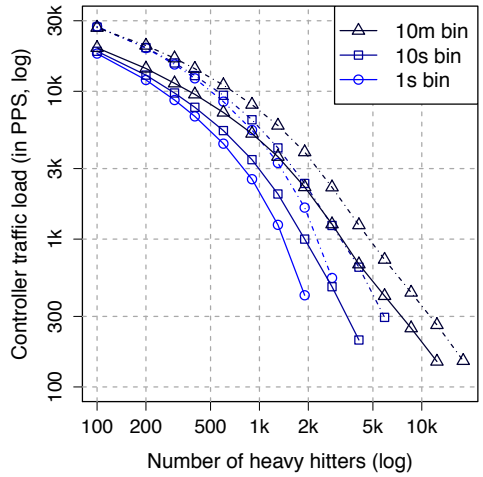
As we will discuss later, the controller typically cannot afford delegating the responsibility for too many flows to the forwarder, because of the associated communication overhead. The most relevant range seems to lie between a few hundred to a few thousand heavy hitters. The above results confirm previous work [77, 135, 147, 75], which has shown that it is possible to offload most of the traffic with a limited number of heavy hitters. However, these works do not study how to derive a heavy hitter selection strategy which trades off the expected benefit of modifying the heavy hitters against the extra work of applying those changes. Instead, previous work has focused on dynamic heavy hitter detection mechanisms [159, 30].

### 4.2.3 Remaining load at the controller

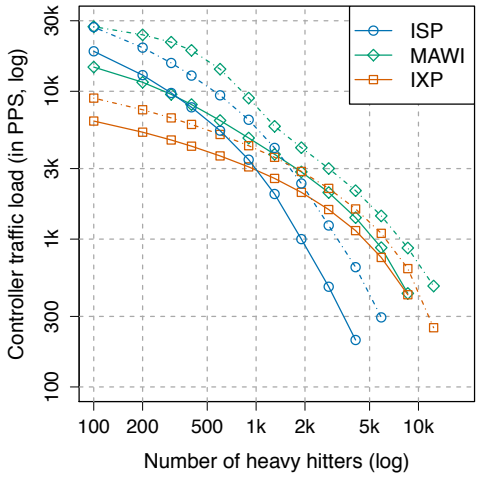
In our context, traffic offloading aims at limiting the rate at which the controller needs to forward packets by offloading most of the traffic load to a specialized forwarding device. By design, the controller is optimized for running complex routing protocols but often it does not satisfy the requirements of packet forwarding. Some known limitations of controllers include the available forwarding bandwidth and delay. Our goal is to limit the amount of traffic the controller has to handle in order to avoid long packet queues and packet loss. As a baseline, we now quantify the amount of traffic the controller has to handle based on our traffic traces.

For this purpose, we rely on a practically infeasible heavy hitter selection strategy called bin-optimal. The bin-optimal strategy assumes that future traffic is known, and that the set of heavy hitters to be used on the forwarder can be updated instantly. The bin-optimal strategy uses fixed time bins. We choose 1s, 10s, and 10 minutes as time bins to capture the varying stability properties of heavy hitters across different time scales. The bin-optimal strategy selects for each time bin the top $n$ heavy hitters, where $n$ is the number of entries that the forwarder keeps.

Figures 4.3(a) and 4.3(b) show the traffic load in packets per second (PPS) which remains at the controller vs. the number of heavy hitters which are delegated to




(a) Impact of bin size on controller load (ISP). Dotted curves represent max values.




(b) Controller load with 10s bins (all traces). Dotted curves represent max values.

**Figure 4.3:** Controller traffic load under the bin-optimal strategy.

the forwarder, on a log-log scale. Overall, we see that even for small values of $n$, in the order of a few thousands, the traffic load at the controller is relatively low. When taking the ISP trace as an example, with $n = 1,000$ the median traffic load at the controller is 3,400 PPS (with a maximum of 6,400 PPS) which corresponds to a relative reduction in load of 92 %. With shorter time bins the offloading gain can be further increased. Figure 4.3(a) shows both the median as well as the maximum traffic rate at the controller for the three considered time bins.

Figure 4.3(b) compares the controller traffic load for all three traces under different numbers of $n$ and fixed 10s time bins. We observe that with all considered traces, the offloading gain increases rapidly with increasing $n$. All three traces incur only up to around 1,000 PPS at the controller when offloading the top 5,000 prefixes.

### 4.2.4 Churn in heavy hitters

Heavy hitters enable significant offloading but are also challenging because of their dynamics [113, 146, 147, 157, 27]. We examine the impact of variability in the heavy hitters when delegating prefixes to the forwarder. The bin-optimal strategy relies on the foreknown heavy hitters for each time bin. In this case, the churn in the heavy hitters stems entirely from the natural traffic variability. We now quantify the corresponding number of changes to the set of heavy hitters between consecutive time bins, as incurred by the bin-optimal strategy.

Figure 4.4 shows the fraction of heavy hitters that are modified for time bins of 10s, as a function of time for the ISP trace. For small values of $n$ such as a few hundreds we

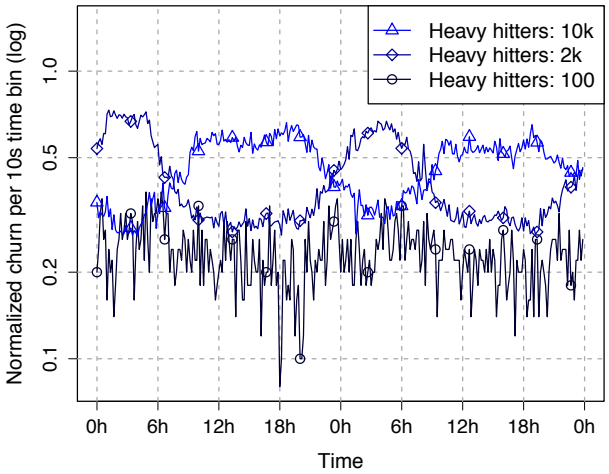


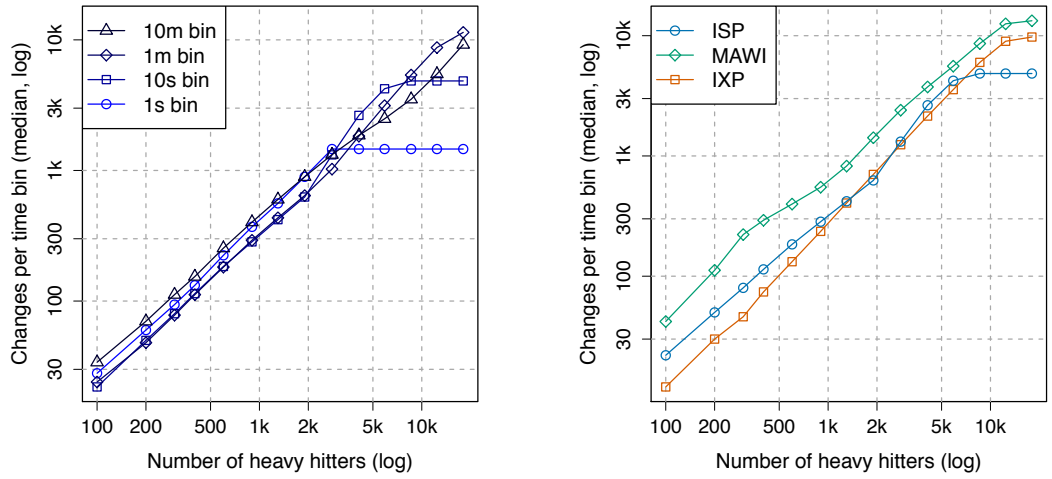

**Figure 4.4:** Churn over time under the bin-optimal strategy (ISP).

observe churn in the order of 15 %. When increasing $n$ up to a few thousands, around half of the heavy hitters are modified. As expected, we notice that changes among the top ranked prefixes occur much less often than among the lower ranked ones. We also point out that the churn is subject to a time of day effect. Interestingly, the direction of the time of day effect inverts as the number of heavy hitters is increased. This is due to the reduced variability and traffic volume during the night.

As shown on Figure 4.5(a) for the ISP trace, relying on larger bins can incur slightly larger numbers of changes from one time bin to the next, as compared to shorter time bins. Moreover, on Figure 4.5(b), we see that even though our network observation points are quite diverse, similar trends apply. We explain the larger churn in case of the MAWI trace again with the assumption of a flat /24 address space.

## 4.3 Traffic-aware Flow Offloading (TFO)

In Section 4.2, based on Internet traffic observations, we have identified the main challenges behind heavy hitter selection strategies for traffic offloading: maintaining a high offloading ratio while limiting the changes to the set of heavy hitters. In this section we first examine how well traditional caching strategies perform in the context of the offloading problem (Section 4.3.1). Traditional caching strategies are not appropriate in our context due to their high churn and the fast reaction times they require. This leads us to develop and evaluate our heavy hitter selection strategy called Traffic-aware Flow Offloading (TFO). Based on our traffic traces, we show that TFO achieves low churn, one order of magnitude less than the bin-optimal strategy from Section 4.2.3, while achieving a similarly high offloading gain.




(a) Churn for different time bins (ISP).

(b) Churn for the three traces.

**Figure 4.5:** Rate of churn under the bin-optimal strategy.

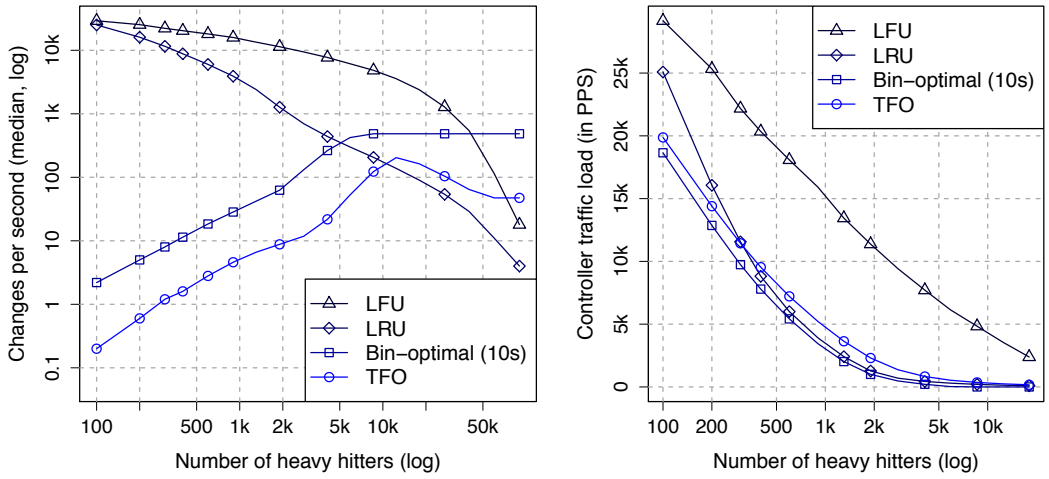
### 4.3.1 Limitations of traditional caching

Traditional caching techniques react on individual misses: When a cache miss occurs, the missing object is fetched and then placed into the cache. During the time interval between the event of a cache miss and the completion of a cache update, incoming packets need to be buffered or will lead to packet re-ordering or loss.

The main difference among traditional caching techniques lies in the strategy to determine which cache entry to evict and replace upon a cache miss. Two common cache replacement strategies are least recently used (LRU) and least frequently used (LFU). We implement both strategies based on the optimistic assumption that cache replacement is instantaneous, to be favorable toward their offloading gain.

The curves for LRU and LFU in Figure 4.6(a) show the amount of heavy hitter changes per second as a function of the number of heavy hitters, $n$. LRU outperforms LFU, consistent with previous work on route caching [77]. Figure 4.6(a) also shows the results for the bin-optimal strategy based on 10s bins. Both LRU and LFU imply a significantly higher churn rate as compared to bin-optimal for up to a few thousand of heavy hitters. For larger values of $n$, traditional caching can outperform bin-optimal in churn rate, this is when $n$ gets close to the total number of active flows. With regards to the remaining traffic load at the controller as shown on Figure 4.6(b), LFU exhibits the worst results. For $n > 200$, all strategies except LFU result in similar remaining loads at the controller.

In summary, these results indicate that the traditional caching strategies LRU and LFU suffer from a major limitation, namely the high rate at which heavy hitters




(a) Churn in comparison.

(b) Controller traffic load.

**Figure 4.6:** Controller traffic load and churn for different strategies (ISP).

need to be changed, especially for values of $n$ ranging up to a few thousands, *i.e.*, the range of $n$ in which offloading is likely to be relevant.

### 4.3.2 The TFO algorithm

In this section, we propose our heavy hitter selection strategy: Traffic-aware Flow Offloading (TFO). The design criteria of TFO stem from our observations in the earlier sections. The goal is to handle most traffic at the forwarder while limiting the number of changes to the heavy hitters. The idea behind TFO is to leverage the stability properties of heavy hitters, which have been shown to depend on their rank [147]. Therefore, we maintain traffic statistics at multiple time scales which enable TFO to take into account short-term as well as long-term trends for heavy hitter selection. To compute a new set of heavy hitters, we further take into account the set of heavy hitters which are currently in place, in order to trade-off the cost of changing the set of heavy hitters with the expected increase in offloading gain.

Figure 4.7 illustrates the four different steps of TFO. We start by taking top heavy hitters from each time scale in a round-robin manner, until we reach a pre-defined value of $n$. The round-robin pre-selection gives equal chances to heavy hitters that are highly ranked at different time scales. Second, we compute the difference between the current heavy hitters at the forwarder and the set of heavy hitters from the first step. This gives us two sets of heavy hitters: those to be installed and the ones to be removed. The last two steps are key to reduce the churn while ensuring fast reaction times. In the third step, we sort these two sets into lists according to their traffic share in inverse orders, with respect to the smallest time scale. Finally, in the fourth

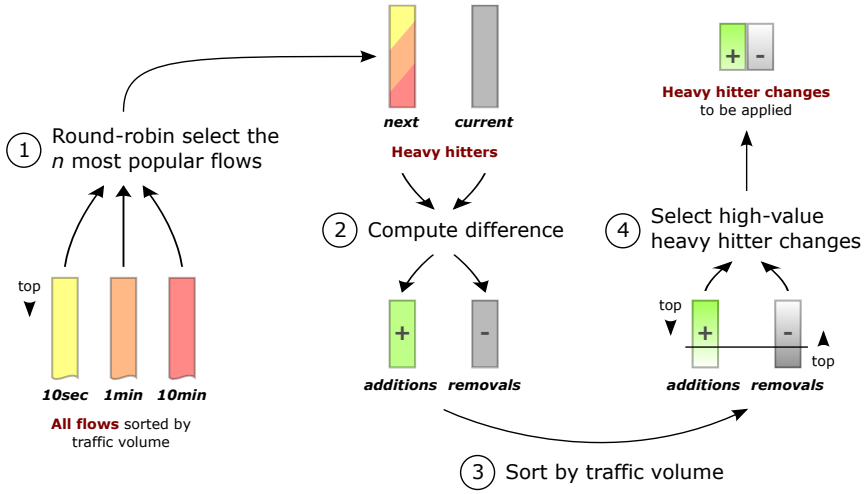
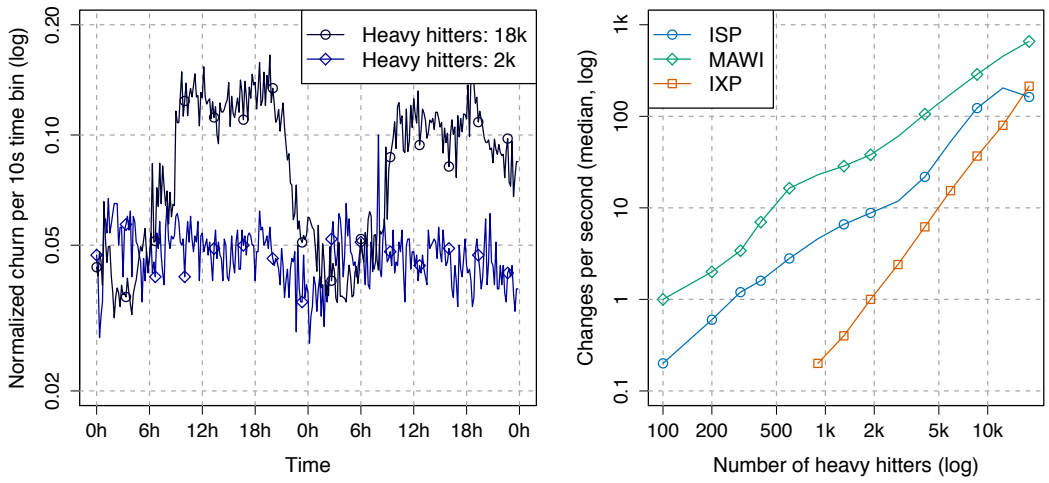



**Figure 4.7:** Functional diagram of Traffic-aware Flow Offloading (TFO).

step, we pick such modifications in which the additions are ranked much higher than the removals, *e.g.*, they differ in traffic share at least by a factor of 2. This fourth step gives priority to the heavy hitters which are already in place, effectively reducing the churn. Consequently, the chosen factor affects the trade-off between offloading gain and communication overhead. In future work, we want to study how to optimally determine this factor. Note, that sudden high-ranked flows can (and should) still enter the set very quickly, *i.e.*, in one round of set computation.

### 4.3.3 Evaluation of TFO

We now evaluate the performance of TFO based on our three traffic traces. Figure 4.6(b) shows, as a function of $n$, the remaining traffic load at the controller for the ISP trace. The offloading effectiveness of TFO is marginally worse compared to the bin-optimal strategy. However, the bin-optimal strategy involves a churn rate which is about an order of magnitude higher as compared to TFO, see Figure 4.6(a). When comparing the evolution of churn over time of the bin-optimal strategy on Figure 4.4, against TFO on Figure 4.8(a), we observe that TFO changes only less than 10 % of the heavy hitters between any two time bins whereas bin-optimal exceeds 70 %, for the ISP trace with $n = 2,000$. When letting $n$ grow to values as large as 18,000, we observe that the rate of churn varies heavily and follows a strong time of day pattern in cases of TFO and bin-optimal. Smaller values of $n$, such as 2,000, are less sensitive to time of day effects and thus show only limited variations. We conclude that an offloading system that aims too high in its offloading gain will face strong churn rate variations. The key lies in understanding the trade-off when choosing $n$: On the one hand, $n$ must be large enough such that the heavy hitters can capture enough traffic to retain a low enough traffic load at the controller. On the




(a) Churn over time (ISP).

(b) Churn for the three traces.

**Figure 4.8:** Rate of churn under TFO.

other hand, one should strive to keep $n$ as small as possible to keep the associated rate of churn and its variation low.

Figure 4.6(a) also compares TFO with the traditional caching strategies LRU and LFU. For $n < 10,000$, TFO outperforms both LRU and LFU in terms of churn rate. However, at around $n = 10,000$ we observe a cross-over point at which LRU starts to outperform TFO. At such large values of $n$, the rate of churn also exhibits strong variations over time.

Figure 4.8(b) presents the median number of changes to the heavy hitters as a function of $n$ for the three traces. We observe an impact of traffic aggregation on the rate of churn: The MAWI trace leads to the highest churn rate followed by the ISP trace, and finally the IXP one with the least amount of churn. Also, the top 500 heavy hitters in the IXP trace are so stable that their median churn is 0 over the four days. This result is important with respect to the scalability of the offloading problem, as our results suggest that the higher the traffic aggregation, the lower the churn.

Figure 4.9 compares the traffic load at the controller for the three different traces when using TFO. The figure shows the median and maximum controller load as a function of $n$, in log-linear scale. The ISP trace gains most with increasing $n$. This is consistent with Figure 4.2, where the top heavy hitters for the ISP capture a larger fraction of the total traffic as compared to the other traces. For $n = 6,000$ we observe an offloading gain of more than 99 %, resulting in a median of only 500 PPS at the controller.

The MAWI trace has the widest CDF curve on Figure 4.2, which leads to the worst offloading gain except for very small $n$. Due to the anonymization technique of the

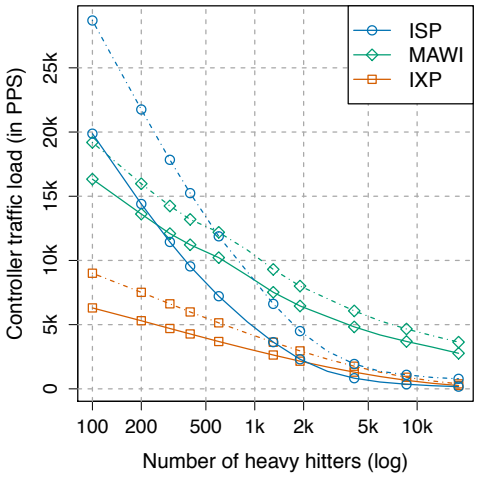



**Figure 4.9:** Controller traffic load under TFO (all traces). Dotted curves represent max values.

MAWI trace, we consider a flat /24 address space which potentially slices heavy hitters into many /24 prefixes. Hence, the MAWI results are penalized in terms of offloading gain per heavy hitter. However, they still result in a very limited packet rate at the controller. For example, with $n = 6,000$, TFO still achieves an offloading gain of 86 %, leaving only 4,200 PPS (median) at the controller.

In contrast to MAWI, the IXP trace presents a very smooth S-shaped CDF curve on Figure 4.2. This stems from the level of aggregation at the observation point, which represents traffic exchanged between hundreds of ASes. Figure 4.9 shows that for most values of $n$, the IXP trace leads to the lowest controller load compared to the other traces. However, as mentioned earlier, due to the low packet sampling rate of this trace we expect a bias in favor of the observed traffic share of the heavy hitters. For $n = 6,000$, TFO achieves an offloading gain of 90 %, leaving less than 1k PPS (median) at the controller.

Note that real traffic traces, especially those as large as the IXP one, contain certain types of attacks such a DDoS, scans, and other types of anomalies. Our results show that despite the presence of such events, neither the churn nor the offloading gain seem affected, even when considering the maximum churn and maximum traffic load at the controller.

## 4.4 Related work

We revisited the Zipf property of Internet traffic and derived a traffic offloading strategy that achieves high offloading gain while limiting the overhead caused by churn in the heavy hitters. With recent work which aims to improve the forwarding

performance of software routers [59, 44] as well as their port density [39], we believe that traffic offloading is a viable alternative to traditional router designs.

Currently, routers maintain large numbers of forwarding entries in specialized memory for fast per-packet lookups (see Section 2.2.1). Routers require ever increasing amounts of such specialized memory, which typically is expensive and/or energy hungry. Furthermore, the rate at which the forwarding entries need to be changed is significant, especially due to routing protocols dynamics [85, 148, 82]. For example, reported BGP update rates are in the order of tens or hundreds per second on average with peaks of several thousands per second, for about 430,000 prefixes. Even with such a limited churn relative to the number of prefixes, the implication of BGP updates on the data plane can be an issue [82, 2]. Compared to today's BGP update rates, the dynamics in heavy hitters is much lower, both in relative and absolute numbers. Furthermore, BGP updates are expected to have limited impact on the heavy hitters as popular prefixes have been shown to be stable [124].

Our work shows that an offloading system has potential to improve router designs, by further decoupling the tasks of the route controller with those of the forwarding engines. The main advantage of the offloading approach is to limit the interactions between the control and data planes to what is strictly necessary from the viewpoint of the traffic. By taking advantage of the forwarding capabilities of programmable forwarding engines such as OpenFlow [95], NetFPGA [91], and ATCA [6], we believe that IP router designs which leverage open-source routing software and traffic offloading, such as our SDN Router (Chapter 3), may challenge existing router designs.

The TFO strategy, when realized in our SDN Router, has an issue in the sense that offloaded prefixes will hide more specifics, if they are not included in the heavy hitter set as well. In cases where such hidden more specifics have different next-hops compared to their parent prefix which hides them, the forwarding correctness constraint is violated. Many options to solve this issue exist. Recent work [53] shows, based on regional ISP traffic traces, that one would have to increase the cache by a factor of, *e.g.*, five, to also include the hidden more specifics. However, simply including all more specifics may not be the most efficient solution to the cache hiding problem. In future work, we plan to design and study algorithms which already take into consideration the number of more specifics of a high-volume prefix when performing the initial heavy hitter selection. That way, we expect to be able to reduce the number of additional more specifics to be added to the heavy hitters. Another option to mitigate the cache hiding issue is to restructure the forwarding table such that there are no, or little, overlapping entries. This can be done, *e.g.*, by controlled prefix expansion [138], or even by transforming the FIB into a collection of non-overlapping prefixes, *e.g.*, into slash-24s only. The results based on the MAWI trace presented in this chapter are not affected by the cache hiding problem as they are based on a flat slash-24 address space, thus they can be taken as an example of how TFO would perform in that case.

In addition, as we plan to apply traffic offloading and TFO to other domains, we intend to incorporate TFO into FlowVisor [134] to complement its slicing policy.

## 4.5 Summary

Internet traffic follows Zipf's law: A small fraction of flows capture most of the traffic. We propose to exploit this property to offload most of the traffic from a complex controller to a simple forwarder. We show that the main challenge lies in achieving a high traffic offloading gain while limiting the churn, and propose a heavy hitter selection strategy called Traffic-aware Flow Offloading (TFO). We perform simulations of TFO on real traffic traces. The results highlight the effectiveness of the strategy used in TFO and suggest that traffic offloading should be considered to build feasible alternatives to current router designs. In future work, we want to study the impact of parameters on TFO such as the chosen time scales. We also plan to evaluate TFO under different traffic loads with flows of granularities other than BGP prefixes. Another future study will focus on possible attacks against the offloading system and adequate defense mechanisms.

# 5

# Exploiting Locality of Churn for FIB Aggregation

Complementary to our traffic offloading concept, we continue with exploring ways to make even more efficient use of the physically available FIB memory through FIB aggregation. Snapshots of the FIB in Internet routers can be compressed (or aggregated) to at least half of their original size, as shown by previous studies.

However, the permanent stream of updates to the FIB due to routing updates complicates FIB aggregation in practice: Keeping an optimally aggregated FIB in face of these routing updates is algorithmically challenging. A sensible trade-off has to be found between the aggregation gain and the number of changes to the aggregated FIB. In this chapter we investigate whether the *spatial* and *temporal locality* properties of BGP updates to the tree-like FIB data structure can be leveraged by online FIB aggregation.

Our contributions include *(i)* an empirical study of the spatial and temporal locality of BGP updates in public Internet routing traces, *(ii)* the specification and simulations of our Locality-aware FIB Aggregation algorithm (LFA), and *(iii)* the impact of FIB aggregation on traffic offloading in our SDN Router. Our results show that even a simple algorithm can effectively exploit the locality of FIB churn to keep low the number of updates to the aggregated FIB, as most FIB updates affect only a small number of regions in the FIB. We further show that by implementing FIB aggregation in the SDN Router, improvements to the traffic offloading ratio can be achieved.

## 5.1 Introduction

At a high level, Internet routers are built around a routing control plane which runs routing protocols and makes routing decisions, and a data plane which forwards packets according to the decisions of the control plane (more details in Section 2.2.1). The crucial link between the two components is the Forwarding Information Base (FIB), containing the forwarding rules. The route controller inserts and deletes FIB entries according to its route computations. The forwarding plane uses the FIB to perform an IP destination lookup on each incoming packet. This requires the FIB to support very fast IP destination lookups so that packets can be forwarded at line rate. In addition, the FIB needs to support frequent updates to the forwarding rules due to the churn in the BGP routing table. Finally, the number of forwarding rules that a FIB needs to keep is growing over time, putting extra pressure on the FIB memory capacity[1]. To fulfill all these requirements, FIB memory used in modern routers is very expensive and power-hungry. It is also considered a main limiting factor in terms of a router's lifetime [96], as mentioned in earlier chapters already.

A natural and local solution to mitigate the problem – before possible long-term solutions are deployed – is the *aggregation* (or *compression*) of the FIB, *i.e.*, the replacement of the existing set of rules by an *equivalent but smaller* set. The aggregation of FIB rules has the appealing property that it is a purely local solution, in the sense that it does not affect neighboring routers and it can be realized entirely in routing control software [161].

While the compression of the FIB is beneficial in terms of memory, it also entails a potential overhead: As the FIB contents of a router change over time – several hundreds of rules are modified each second on average [45] –, FIB aggregation may lead to a situation where already aggregated FIB entries need to be de-aggregated again [144], resulting in additional updates to the aggregated FIB. There is a certain cost associated with each such update as the internal FIB structures have to be updated, delaying the corresponding changes to the forwarding plane. Hence, FIB management strategies, including FIB aggregation algorithms, should aim at limiting as much as possible the number of FIB updates.

In a stream of BGP updates, we typically observe an uneven distribution of updates across BGP prefixes. For example, as shown in Figure 5.1, the distribution of BGP updates to our ISP aggregation router (see Section 4.2.1) is heavily skewed: More than 60 % of all updates are contributed by only 10k prefixes, which is less than 4 % of all prefixes in the FIB. Accordingly, we propose to study *spatial* and *temporal* locality properties in BGP update streams and leverage those in FIB aggregation algorithms to accomplish efficient aggregation while limiting the numbers of updates to the aggregated FIB.

[1]The BGP routing table contains more than 430,000 entries as of mid-2012, up 15% from just a year ago.

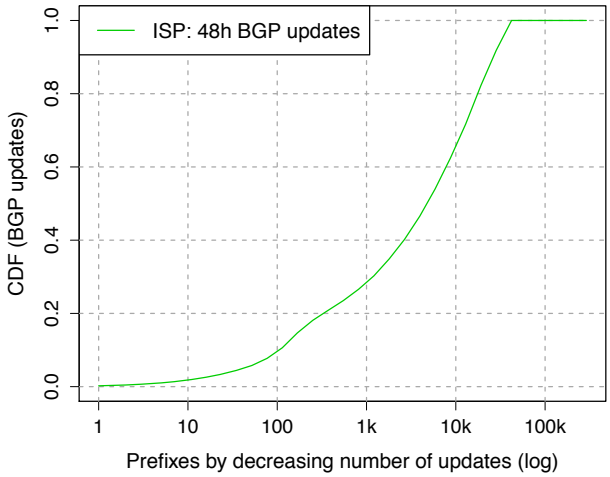



**Figure 5.1:** Distribution of numbers of BGP updates per prefix (ISP trace, see Section 4.2.1).

**Contributions.** We present and evaluate the Locality-aware FIB Aggregation algorithm (LFA). LFA exploits the spatial and temporal locality of BGP churn and aggregates slices of the FIB (STICKS) adaptively to amortize update costs with aggregation gains. We provide an empirical analysis of LFA under publicly available Internet routing data to investigate the trade-offs associated with LFA. We implement FIB aggregation in our SDN Router and show, that improvements to the traffic offloading ratio and the corresponding reduction in controller traffic load can be achieved.

The remainder of this chapter is organized as follows. We start in Section 5.2 with background information about FIB aggregation in general and a description of our assumptions. We discuss the related work in Section 5.3. LFA is introduced in Section 5.4 where we also report on our empirical results on FIB churn locality. We then evaluate the impact of FIB aggregation in our SDN Router in Section 5.5. In Section 5.6 we discuss future work and summarize in Section 5.7.

## 5.2 Background

**Terminology.** An (IP) *address* is a binary string of length $w$ (*e.g.*, $w = 32$ for IPv4 and $w = 128$ for IPv6) or equivalently an integer from $[0, 2^w - 1]$. An (IP) *prefix* is a binary string of length at most $w$; we denote the empty prefix by $\varepsilon$. A prefix *matches* all addresses that have the prefix as their first bits.

We consider an Internet router with a number of network interfaces, or *ports*. A *Forwarding Information Base* (FIB) is a set of *forwarding rules* used by the router

for its packet forwarding operations, where each such rule is a *prefix-port* pair $(p, c)$. A *port* in this context represents all information needed for the router to forward IP packets to a given destination, *e.g.*, the IP address of the next-hop router. For every incoming packet the router performs a destination lookup based on the destination IP address $x$ of the packet. The destination lookup is a *longest prefix match*: Among the forwarding rules $\{(p, c)\}$, the router finds the longest $p$ being a prefix of $x$, and forwards the packet to port $c$. If no rule matches, *i.e.*, no route to the destination is known, the packet is dropped.

For instance, consider a FIB containing four rules $\{(\varepsilon, a), (00, b), (1, c), (11, a)\}$, where $a$, $b$, and $c$ are ports. This FIB could be replaced by an equivalent FIB containing the rules $\{(\varepsilon, a), (00, b), (10, c)\}$. In this compression process, we require *strong forwarding correctness* [161], *i.e.*, we require that the forwarding and dropping behavior remain the same.

We denote the original set of forwarding rules by OT (Original Table). The OT is updated according to the routing protocols by the route controller. Prior to downloading the OT into the FIB of the router we apply FIB aggregation: The forwarding rules of the OT are replaced by an equivalent but smaller set, denoted by AT (Aggregated Table). Hence, the FIB of the router contains the AT.

We assume continuous time; at any time $t$, a single forwarding rule may change its port. The input is a sequence of such changes called *events*. After a change occurs, the route controller must ensure that the AT is equivalent to the OT. To this end, the route controller may apply updates to the forwarding rules in the FIB. The commands may also be issued at any later point in time (*e.g.*, for delayed FIB aggregation).

**Assumptions.** We take a pragmatic standpoint and study algorithms that do not have any knowledge of future prefix changes and that need to decide *online* on where and when to aggregate. Not relying on predictions seems to be a reasonable assumption considering the behavior of the route updates in the current Internet [88].

## 5.3 Related work

There are known fast algorithms for optimal FIB aggregation of table snapshots, for example the Optimal Routing Table Constructor (ORTC) [41] and others [140]. We rely heavily on ORTC in this chapter as it produces aggregated FIBs that are provably optimal. However, as these algorithms do not support efficient handling of incremental updates, a re-computation of the optimally aggregated FIB on each changed forwarding entry is needed. This is computationally expensive and can lead to high churn in the AT. There are several papers that deal with this problem by proposing heuristics that simultaneously try to limit the number of updates to the FIB while maintaining a good compression rate, including SMALTA [144] and

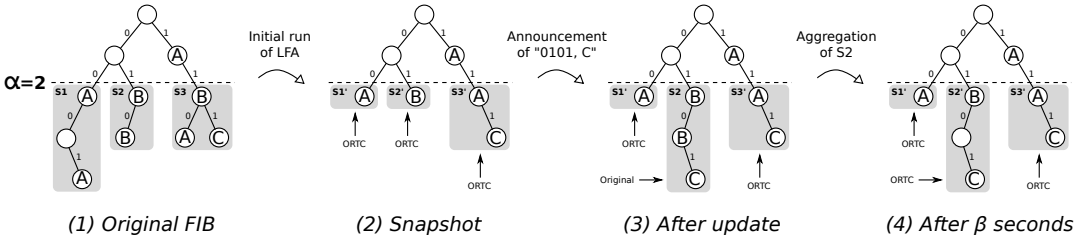



**Figure 5.2:** Locality-aware FIB Aggregation (LFA)

others [90, 161]. However, none of these works consider to leverage BGP churn locality for their benefit, nor do they consider FIB aggregation for the benefit of traffic offloading.

## 5.4 Understanding and leveraging FIB churn locality

In this section we study the locality of FIB churn based on real Internet routing data. From our results, we discuss and quantify the potential benefit of FIB aggregation techniques that treat churny regions of the FIB differently to those with limited churn. We propose an online FIB aggregation algorithm called Locality-aware FIB Aggregation (LFA). LFA aims at aggregating stable parts of the FIB while keeping the less stable ones untouched to limit update overhead. We start by describing the LFA algorithm. Then, we present experimental results based on routing table snapshots that provide a first look at the trade-offs that are associated with LFA. Finally, we evaluate the locality of churn under consideration of the trade-offs and parameters of LFA, based on publicly available streams of BGP updates. Our results indicate that there is substantial locality in routing updates which can be exploited by FIB aggregation algorithms.

### 5.4.1 Locality-aware FIB Aggregation (LFA)

LFA operates as follows. The FIB in its usual trie representation is split into subtrees (STICKS) which are aggregated only when they are considered stable. This is when a STICK has not been affected by updates for a pre-specified time period ($\beta$ seconds), the Optimal Routing Table Constructor (ORTC) [41] is used to optimally aggregate this STICK. Before a STICK is updated due to a routing update, it is reverted to its original (deaggregated) representation before the update is applied. We simulate LFA on real BGP update streams and identify the trade-offs associated with its parameters $\alpha$ (spatial locality) and $\beta$ (temporal locality). In all of our simulations we verify that the AT indeed is equivalent to the OT.

In LFA, the tree is split horizontally into two parts. The upper part, which we call GROUND, contains the less specific prefixes and remains untouched by LFA. Hence, as the GROUND is not subject to aggregation in LFA, routing updates to the GROUND can be applied immediately as they come (one update to the GROUND results in one update to the AT). The lower part contains the more specific prefixes and is aggregated selectively by LFA. The parameter $\alpha$ defines at which depth (prefix length) to draw the line that separates the GROUND from the more specific part of the tree. All prefixes with a prefix length $\geq \alpha$ belong to the more specific part.

The more specific part of the tree is split vertically into subtrees, called STICKS. All the nodes with prefix length $= \alpha$ represent root nodes of individual STICKS. A STICK which has not seen any updates for a pre-defined time period is aggregated using the ORTC algorithm. In LFA, STICKS are aggregated independently from the GROUND: No next-hop information, which can change over time, is being inherited from the GROUND when aggregating a STICK. Also, as original and aggregated STICKS are congruent in their forwarding information, there is no dependency of the GROUND on the present state of a STICK (aggregated or non-aggregated).

The parameter $\beta$ specifies the time in number of seconds after which a STICK is aggregated in the absence of updates. For each STICK a timestamp is maintained that indicates the time of its most recent update. On incoming updates to a STICK we distinguish two cases:

1. STICK *aggregated:* In case the affected STICK is aggregated, the STICK is reverted to its non-aggregated (untouched) version prior to applying the update.
2. STICK *untouched:* Updates are applied as-is to non-aggregated STICKS.

In both cases, the STICK's last update timestamp is set to the time of the update. A priority queue maintains pointers to each untouched STICK, sorted by the time of their most recent update. A timer is implemented for keeping track of the tail of the queue and aggregation is applied to those STICKS that have a last update timestamp $\leq$ current time - $\beta$.

Figure 5.2 illustrates the algorithmic components of LFA for $\alpha = 2$. The trie represents a FIB, trie levels represent prefix length starting at zero, and letters represent ports. Empty nodes do not have a corresponding entry in the FIB. The first figure highlights how $\alpha$ is used to separate the GROUND from the STICKS S1 to S3. Initially, in (2), all STICKS are aggregated using ORTC while reducing the total number of prefixes from 8 to 5. In the figures, we append a prime symbol to the STICK identifiers when they are aggregated, hence we now have the STICKS S1' to S3'.

Next, in Figure 5.2 (3), we consider an update that affects S2'. Prior to applying the update, S2' is reverted to its deaggregated form S2. After that, the update can be applied. In this example the update reflects a prefix announcement which is handled by LFA's insert procedure. Algorithm 1 provides pseudo-code for LFA's insert procedure. We leave out the delete procedure as it is similar to the insert one,

except lines 2 and 10 call *TrieDelete()* instead of *TrieInsert()*[2]. After S2 remains unchanged for $\beta$ seconds, S2 is aggregated again in step (4).

**Algorithm 1** LFA-Insert(P, N)

```
1: if P < α then
2:     TrieInsert(P, N)
3: else
4:     S ← Stick(P)
5:     if IsAggregated(S) then
6:         RevertToOriginal(S)
7:     else
8:         Dequeue(S)
9:     end if
10:    TrieInsert(P, N)
11:    SetTimestamp(S)
12:    Enqueue(S)
13: end if
```

### 5.4.2 Analysis of FIB churn locality

LFA has been especially designed to facilitate studies of the locality of churn in the FIB. More specifically, LFA allows us to (1) quantify the aggregatability of dependency-free[3] regions of the FIB, (2) monitor the locality of churn over time, and (3) study the trade-offs of the parameters $\alpha$ and $\beta$ of LFA.

**Aggregatability of sticks.** We rely on snapshots of real routing tables to study the general aggregatability of STICKS and the dependency on $\alpha$. We obtained the routing table dumps from RouteViews[4] [129]. As the results for the different routing tables we analyzed are similar[5], we present results based on a single routing table snapshot from a large US Internet service provider. This routing table contains about 400,000 entries with more than 900 unique next-hop ASes.

At first, in Figure 5.3, we show the number of STICKS as a function of $\alpha$. The figure compares the maximum possible number of STICKS for a given $\alpha$ with the number of existing STICKS in our snapshot. To get the maximum possible number of STICKS, we assume that all STICKS rooted at $\alpha$ have at least one prefix. Figure 5.3 shows that

[2]We note that real implementations of LFA's insert/delete procedures must be able to instantiate and destroy STICKS, that is when a first prefix of a STICK is announced, or when the only prefix of a STICK is withdrawn, respectively.

[3]A dependency-free region of a FIB is a group of prefixes that does not have more specifics, but less specifics may (and typically do) exist.

[4]Due to limitations in the data we approximate ports by next-hop ASes.

[5]We ran our analyses on about 30 routing table dumps from each year between 2009 to 2012 and observed similar results.

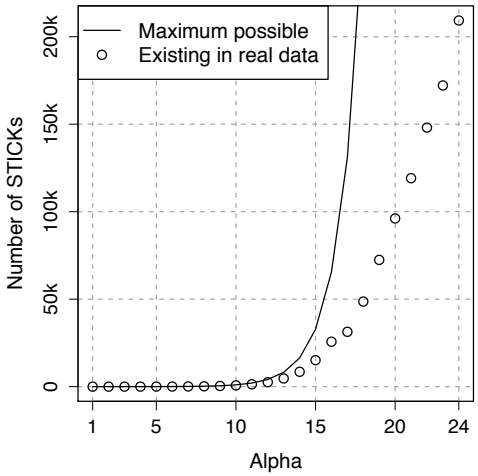


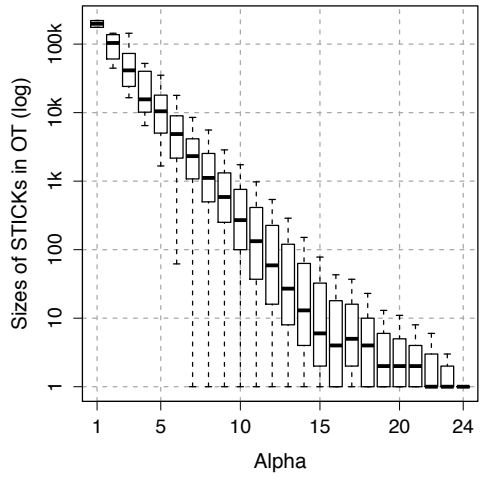

**Figure 5.3:** A first look at the impact of $\alpha$ in LFA:
Number of existing STICKS as a function of $\alpha$.

the number of existing STICKS is substantially smaller than the maximum possible. This means that despite the near exhaustion of the current IPv4 address space, IPv4 FIBs are sparsely populated in terms of their filling of the tree data structure.

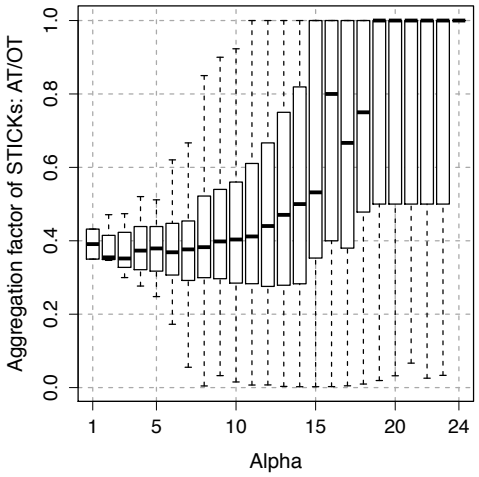
Figure 5.4(a) shows the impact of $\alpha$ on the distribution of OT STICK sizes, *i.e.*, the numbers of prefixes in non-aggregated STICKS. We observe that both the average and the maximum STICK size decreases as $\alpha$ increases. For values of $\alpha$ larger than 7, the minimum STICK size goes to 1, indicating that at least one STICK contains no more than a single prefix. Figure 5.4(b) shows the per-STICK aggregation factor as a function of $\alpha$. For $\alpha \leq 15$, STICKS can be aggregated to half of their original size, while bigger values of $\alpha$ result in worse aggregation factors. We observe a non-monotonic behavior in Figure 5.4(b) for $\alpha \geq 16$. This is a result of the strong dependency of ORTC on the structure of a STICK for the efficiency of its aggregation. This dependency is more visible when STICKS are very small.

We conclude that values of $\alpha \leq 15$ will lead to good aggregation factors without incurring a high overhead for tracking and keeping the state of a large numbers of STICKS, while at the same time achieving median STICK sizes of more than one. It is a necessary (but not sufficient) requirement for a STICK to be larger than one in size in order for it to be effectively aggregatable.

With Figure 5.5, we complete our routing table snapshot analysis. Figure 5.5 shows, as a function of $\alpha$, the total number of prefixes in the AT. We further decompose the AT size into its GROUND and STICK components. For $\alpha \leq 15$, the GROUND contributes only limited numbers of prefixes while the prefixes from the STICK components dominate the total size of the AT, which is more than 60% off of the size of the OT. This is consistent with our results in Figure 5.4(b), in which we show that the aggregation gain suffers when alpha grows beyond 15. Furthermore, we observe




(a) Distribution of STICK sizes in OT as a function of $\alpha$.




(b) Per-STICK aggregation gain as a function of $\alpha$.

**Figure 5.4:** A second look at the impact of $\alpha$ in LFA.

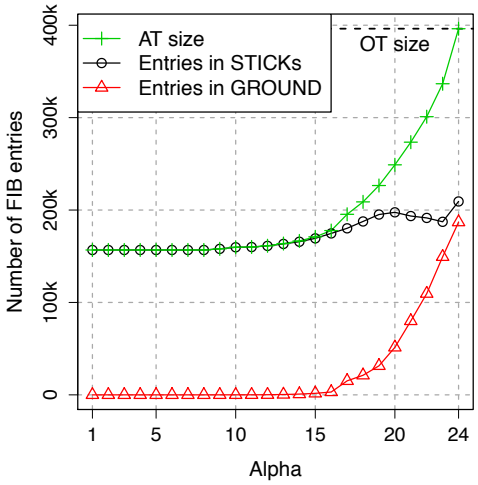
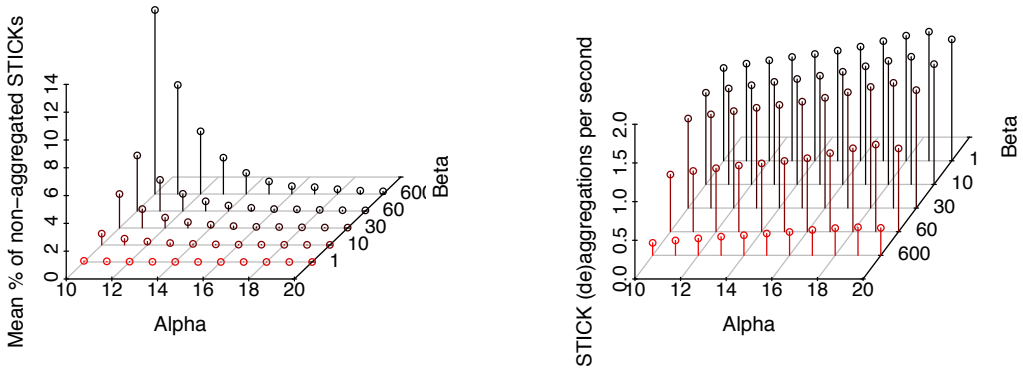



**Figure 5.5:** Size of aggregated STICKS and GROUND as a function of $\alpha$.




(a) Fraction of non-aggregated STICKS.

(b) Number of STICK (de)aggregations.

**Figure 5.6:** LFA trade-offs with $\alpha$ and $\beta$.

a steep increase in the size of GROUND for $\alpha \geq 20$. At the same time, we see limited changes in the STICK sizes. As a result, the total size of the AT grows until it reaches the size of the OT, see the dashed line on Figure 5.5.

In summary, based on our analysis, a sensible region of $\alpha$ in the general case of current IPv4 routing tables appears to lie below 16. The results in Figure 5.5 are particularly encouraging for LFA as they show that even for $\alpha$ up to 18 the total size of the FIB can be reduced by at least 50%. This gives us evidence in the approach of aggregating STICKS individually, as the achieved aggregation factors are similar to those from optimal aggregation of the complete FIB [41, 144].

**Trade-offs over time.** Now that we have expectations from our analysis about the impact of $\alpha$ on the achieved aggregation factors, we now analyze the online performance of LFA under changing $\alpha$ and $\beta$. For that, we rely again on publicly available data from the RouteViews project [129]. We rely on a single dataset taken from a Canadian ISP router that contains more than 400,000 routing table entries. We obtain the routing table snapshot along with a stream of more than 400,000 BGP updates which cover a period of seven hours. This router has almost 200 unique next-hop ASes. We verified that the results presented are similar to those from different routers on different days. In the following results, we consider values of $\alpha$ ranging from 10 to 20, and values of $\beta$ of 1, 10, 30, 60, and 600 seconds. We chose these values of $\beta$ because they capture the scales at which BGP routing events take place [45].

In Figure 5.6(a), we show the fraction of STICKS that are non-aggregated over time. This is a particularly important metric to consider as it provides some intuition about the locality of routing table updates. Non-aggregated STICKS represent those that

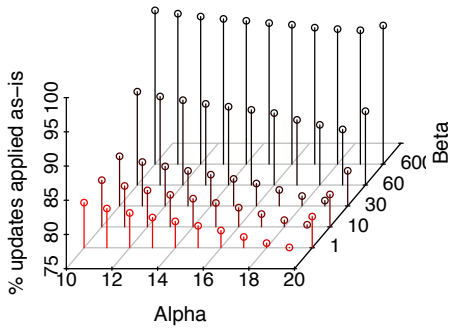
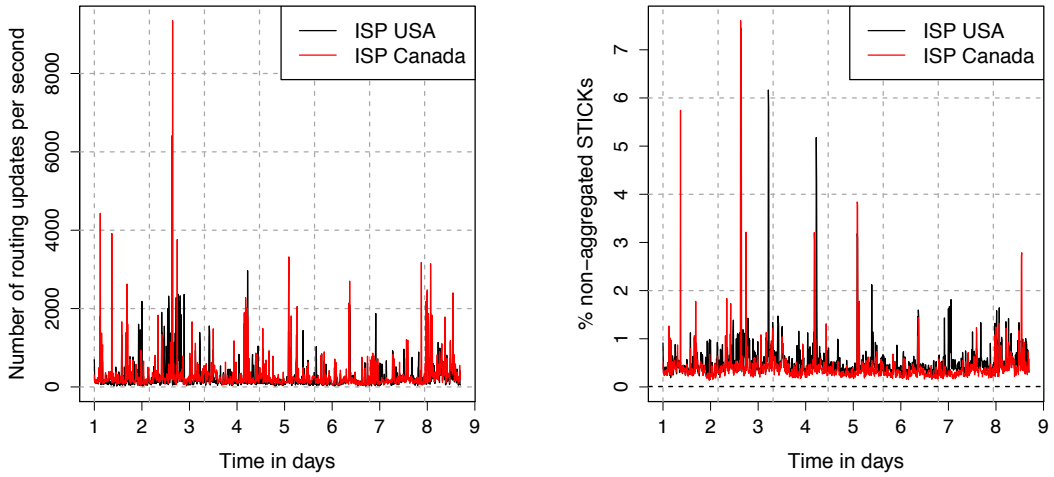



**Figure 5.7:** LFA trade-offs with $\alpha$ and $\beta$: fraction of routing updates that are applied as-is, which is when the update affects the GROUND or a non-aggregated STICK.

have seen updates within the last $\beta$ seconds. The results in Figure 5.6(a) show, that for $\alpha \geq 14$ and $\beta \leq 60s$ the fraction of non-aggregated STICKS is very low: On average, less than 0.4% of the STICKS are not aggregated.

Another metric to consider is the number of (de)aggregations of STICKS over time. This metric will tell us how often updates hit aggregated STICKS, requiring to deaggregate them before applying the update, and how often STICKS are aggregated after a stable period of $\beta$ seconds. In Figure 5.6(b) we show the average number of STICK (de)aggregations per second as a function of $\alpha$ and $\beta$. For improved visual presentation we reverted the ordering of values on the y-axis. The results show that even for a value of $\beta$ as small as $1s$ the average number of STICK (de)aggregations per second does not exceed 3. We also observe that this metric strongly depends on $\beta$ as the results show a steep increase when considering $\beta$ from $600s$ to $1s$.

Finally we study the impact of $\alpha$ and $\beta$ on the fraction of routing table updates which can be applied immediately at no extra cost due to deaggregations of STICKS. This includes all routing table updates that affect either the GROUND, or non-aggregated STICKS. Figure 5.7 shows the fraction of such routing table updates as a function of the parameters $\alpha$ and $\beta$. We observe that as $\beta$ decreases, this fraction also decreases. This is expected since smaller values of $\beta$ limit the ability of LFA to leverage update locality over time. On the other hand, the behavior of $\alpha$ is non-trivial. As $\alpha$ increases, the GROUND increases, while non-aggregated STICKS decrease (Figure 5.6(a)). The net effect we observe is a decrease of the number of updates that can be applied as-is. This happens because the number of updates to the GROUND increases very slowly with $\alpha$, while non-aggregated STICKS decreases much faster with $\alpha$. The reason for this behavior is that smaller STICKS have a higher likelihood of being aggregated, as they are less likely to be affected by routing updates.




(a) Number of routing table updates per second. This plot shows the maximum of every 10 minute time bin.

(b) Fraction of non-aggregated STICKS per second with $\alpha = 15$ and $\beta = 60s$. This plot shows the maximum of every 10 minute time bin.

**Figure 5.8:** LFA performance over time.

**A sensible trade-off.** We now combine the insights from our earlier results and extract the most sensible trade-off in the selection of $\alpha$ and $\beta$. Our results suggest that $\alpha$ should not be larger than 15 to achieve good aggregation gains. The results from our online experiments suggest that $\alpha$ should be $\geq 14$ to maintain a low number of non-aggregated STICKS for $\beta \leq 60s$. For $\alpha = 14$, Figure 5.6(a) suggests that $\beta$ should be no larger than $60s$, while Figures 5.6(b) and 5.7 show benefits in choosing a large value of $\beta$. In summary, our analyses indicate that the most appropriate values are $\alpha = 15$ and $\beta = 60s$.

**Performance over time.** To better understand the performance of LFA with $\alpha = 15$ and $\beta = 60s$, we now perform experiments based on more than one week worth of routing table updates. The results are shown in Figures 5.8 and 5.9 for two ISP routers, one from Canada and one from the USA. We plot the workload in Figure 5.8(a) as the time-series of the number of BGP updates per second. We show the maximum value for every 10 minute time interval to stress how bursty BGP updates can be. We notice several routing events which cause more than 2,000 routing table updates per second. In Figure 5.8(b) we plot the corresponding fraction of non-aggregated STICKS over time. Again, to give importance to the high (bad) values, we show the maximum out of every 10 minute time bin. The auto-correlation (not shown) between the original time-series used in Figures 5.8(a) and 5.8(b) exhibits the impact of $\beta$: We observe a strong correlation within time lags of 60, while larger time lags show a much smaller correlation. Finally, we show in Figure 5.9 the CDF of the fractions of non-aggregated STICKS in one second time intervals. Contrary to Figures 5.8(a) and 5.8(b) that show maximum values over 10 minute bins, Figure 5.9

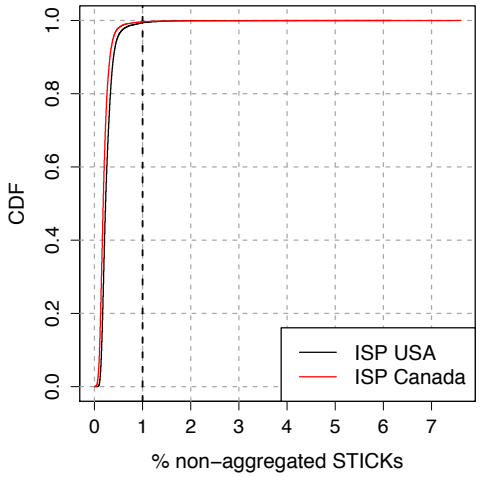



**Figure 5.9:** LFA: distribution of fractions of non-aggregated STICKS per second.

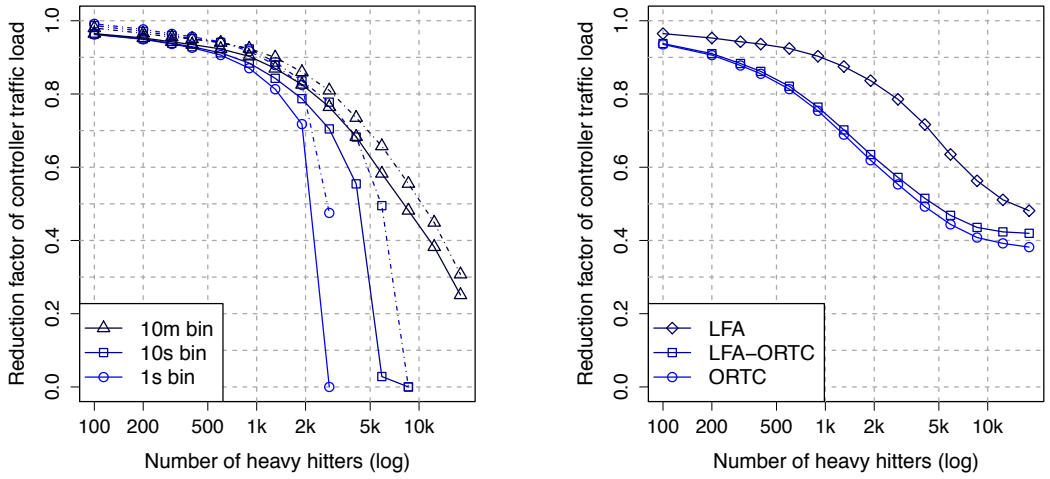
provides a representative perspective on the ability of LFA to keep most of the FIB compressed over time. In more than 99% of the one second time intervals for both routers, less than 1% of the STICKS are non-aggregated. LFA is therefore able to leverage the locality in how the updates affect the FIB structure, by keeping most of it compressed.

**Putting it all together.** Our results show that there is strong locality in the routing table updates with respect to their spatial and temporal properties. This locality can be exploited by FIB aggregation algorithms such as LFA, even under the bursts of routing table updates generated by BGP.

## 5.5 Impact of FIB aggregation on traffic offloading

We now apply FIB aggregation techniques to our SDN Router. More specifically, we analyze the characteristics of FIB aggregation and its impact on the data plane in the context of a traffic offloading system which aims at offloading most of the traffic by focussing on the few most heavily used entries in the FIB of an Internet router.

A straight-forward way to implement FIB aggregation into the SDN Router is to run LFA prior to TFO (see Section 4.3.2), without limiting the choice of algorithms to neither LFA nor TFO. This means the prefixes in the forwarding table are aggregated by a FIB aggregation algorithm, such as LFA, before the heavy hitters are determined by a prefix selection algorithm, such as TFO. As aggregated prefixes span larger address ranges compared to the original ones, we expect them to also capture more traffic. As a result, with aggregated prefixes we can utilize more efficiently the available entries in the set of heavy hitters in terms of the fraction of traffic captured




(a) Impact of LFA on controller load under bin-optimal. Dotted curves represent max values.

(b) Impact of FIB aggregation on controller load under TFO.

**Figure 5.10:** Performance of FIB aggregation in our SDN Router.

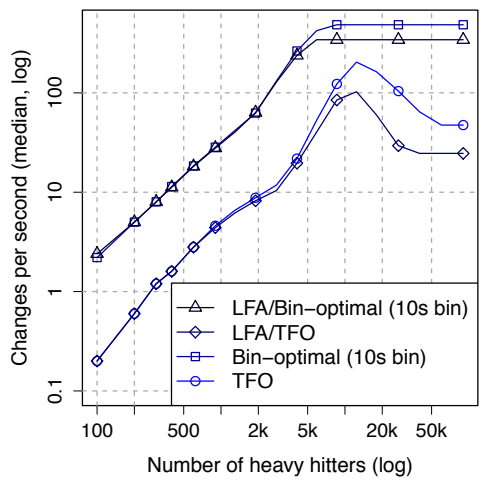
by them. Indeed, in this section we show for a variety of FIB aggregation scenarios, that the offloading ratio achieved by TFO can be increased with the help of FIB aggregation.

### 5.5.1 Offloading ratio under LFA

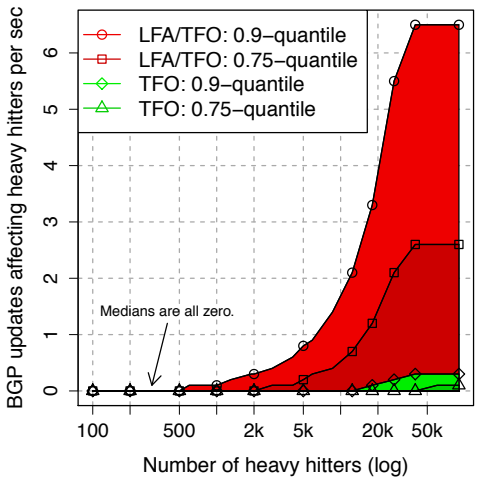
Figure 5.10 shows the benefit that FIB aggregation can have when applied to a traffic offloading system like our SDN Router. The experiments are all based on the ISP datasets; see Section 4.2.1 for a description of the ISP traces, and Section 4.3.3 for an in-depth evaluation of TFO without FIB aggregation. Figure 5.10(a) illustrates the impact of LFA on the controller load, under the idealistic bin-optimal traffic offloading strategy (Section 4.2.3). Recall, that bin-optimal relies on future traffic knowledge. We observe that the number of offloaded prefixes matters significantly: When offloading 2,000 prefixes we can achieve a reduction in controller traffic load of about 20 %. However, with 6,000 heavy hitters and 10s bins we can reduce the controller traffic load already by more than 95 %.

Figure 5.10(b) shows the impact of FIB aggregation on the traffic load at the controller under TFO. For these experiments, we rely on three different FIB aggregation scenarios. *LFA* uses the parameters $\alpha = 15$ and $\beta = 60s$ according to our trade-off experiments in Section 5.4.2. *LFA-ORTC* reflects a configuration of LFA that is approaching pure ORTC, with $\alpha = 10$ and $\beta = 1s$. *ORTC* represents pure ORTC applied to a single forwarding table snapshot from the beginning of the trace[6].

[6]In a pure ORTC scenario, applying a single routing update requires a full re-computation of the




(a) Traffic offloading churn in the heavy hitters.




(b) Rate of BGP updates that affect the heavy hitters.

**Figure 5.11:** Updates to the heavy hitters under FIB aggregation.

The results in Figure 5.10(b) show, that TFO with LFA achieves similar reductions in controller load when compared to bin-optimal with LFA and 10s bins (Figure 5.10(a)) for up to 5,000 offloaded FIB entries. With 2,000 and 6,000 offloaded FIB entries, we observe reductions in controller load of more than 15 % and more than 35 %, respectively. In the same range of numbers of offloaded FIB entries, ORTC achieves reductions of additional 15 to 20 %. The curve for LFA-ORTC is very close to the pure ORTC one, but always slightly worse. This suggests that although we rely on just a single FIB snapshot for the ORTC experiments, we do not seem to have missed any important prefix announcements. Furthermore, this suggests that our LFA-ORTC experiment can indeed serve as a close approximation of pure ORTC in terms of its benefits in offloading ratio.

### 5.5.2 Churn under LFA

We now study the impact of LFA on the frequency of changes to the offloaded FIB entries, also referred to as *churn*. In Figure 5.11(a) we plot the churn rate as a function of the number of heavy hitters. Under the bin-optimal strategy, we observe a slight increase in churn rate when LFA is used. However, the results based on the TFO algorithm are very promising: The number of changes per second to the heavy hitters that TFO requires is strictly lower with LFA, than without LFA.

aggregated FIB, a computationally expensive operation since the algorithm requires three passes through the entire FIB. Our experiment approximates the effectiveness of ORTC by relying on a single, ORTC-compressed FIB snapshot taken from the beginning of the two-day trace.

A different type of churn is caused by BGP updates. Whenever a BGP route change affects an offloaded FIB entry, a corresponding change has to be applied immediately. While keeping in mind that FIB aggregation algorithms typically require multiple updates to the aggregated FIB on a single routing update, we now study the impact of routing updates on the offloaded FIB entries. For this, we plot in Figure 5.11(b) the rate of routing updates that affect the offloaded FIB entries as a function of the number of heavy hitters. In these experiments, the results for TFO reflect the number of BGP updates affecting the heavy hitters, whereas the results for LFA/TFO reflect the number of updates to the offloaded AT entries by LFA. We observe a significant increase in the number of updates that target the offloaded entries when LFA is used. However, the medians are all zero and in 90 % of all seconds in the trace, LFA induces less than seven updates.

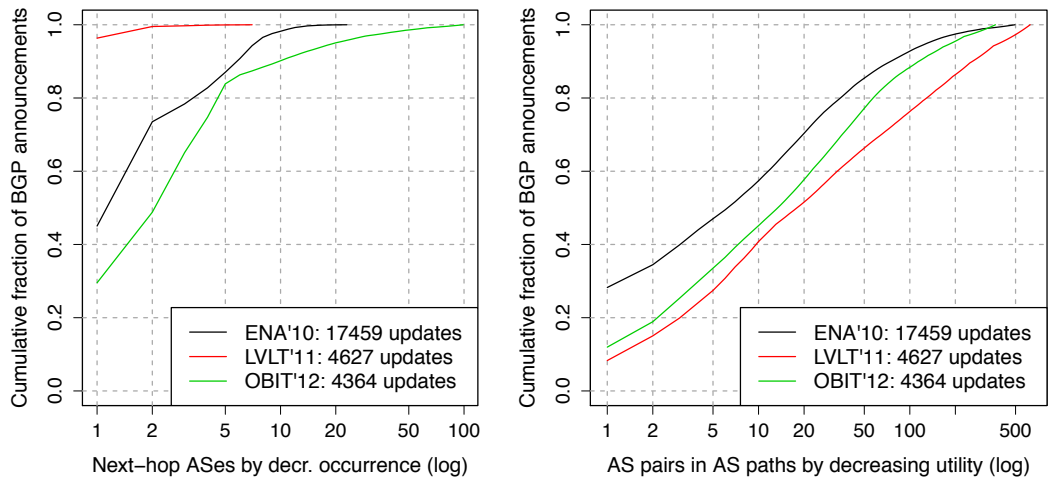
We conclude that LFA, although not optimized for this use-case, can improve a traffic offloading system, like the SDN Router with TFO, in both of its main objectives; *(i)* in the traffic offloading ratio, and *(ii)* in the number of updates to the offloaded FIB entries. In Section 9.2 we propose directions for future research which combine LFA and TFO into a single algorithm for improving FIB aggregation in the specific traffic offloading scenario.

## 5.6 Food for thought: causes of locality in routing updates

In this chapter, we show that BGP update streams have strong spatial and temporal locality properties which can be leveraged by FIB aggregation to reduce churn. However, we do not study the underlying routing events which cause the locality in BGP updates. In this section we discuss some of our intuitions behind this work and perform a few first experiments that motivate further research.

One way to study the temporal as well as spatial locality in updates is by looking at the next-hop ASes that cause most updates in a 1s burst. We perform this experiment for different routers in different years on many 1s update bursts each. In Figure 5.12(a) we observe skewed distributions for three exemplary update bursts: A very small number of next-hop ASes contributes most updates of a burst, *e.g.*, two next-hop ASes account for at least 50 % of the updates. This suggests that coinciding routing events behind a small number of next-hop ASes can cause update bursts.

We now briefly study whether we can identify specific AS-AS links as main causes of BGP update bursts. BGP updates contain the *AS-Path* attribute, which is a list of ASes representing the AS-level route to the destination. For each consecutive AS-AS pair taken from the AS-Path attributes of a burst, we check for their occurrence in all AS-Path's of that burst. In Figure 5.12(b) we plot the fraction of BGP announcements that each AS-AS pair contributes in addition, *i.e.*, their marginal utility. We again observe a highly skewed distribution, in which 20 AS-AS pairs already cover




(a) Distribution of next-hop ASes in different 1s bursts of BGP announcements.

(b) Cumulative marginal utility of unique AS-AS pairs in the AS-PATH attributes in different 1s bursts of BGP announcements.

**Figure 5.12:** Indications of spatial and temporal locality in BGP updates.

more than 50 % of all updates in a burst. As we will see in Chapter 7, there are multiple tens of thousands of AS-AS links in the Internet. If now only 10 to 20 of those links experience a routing event simultaneously, they can already trigger half of a BGP update burst, according to our results.

Accordingly, we propose to further study the actual causes of the temporal locality properties in BGP update streams. Which routing events trigger large numbers of updates in a short amount of time? But also, we want to study the causes of the spatial locality in BGP update bursts and propose to take into account the address space allocation practices to understand, why routing events on AS links can cause large numbers of updates, which are localized in terms of their location in the tree-based FIB data structure.

## 5.7 Summary

In this chapter we study the spatial and temporal locality of routing table updates in the tree-like FIB data structure based on publicly available BGP traces. We propose an online FIB aggregation algorithm, called Locality-aware FIB Aggregation (LFA), that leverages the locality properties in routing table updates. We evaluate the ability of LFA to keep the FIB compressed under the constant stream of BGP routing updates and showed that it is able to keep most of the FIB compressed most of the time. We implement FIB aggregation in our SDN Router and show, that FIB aggregation can improve the traffic offloading ratio.

# 6

# An Open Framework for OpenFlow Switch Evaluation

In this chapter we go back to one key component of our SDN Router prototype; the OpenFlow-enabled switch. When building a network based on OpenFlow in general, or when using OpenFlow-enabled switches for implementing an SDN Router in particular, one must take into account the performance characteristics of the used OpenFlow switch implementations in order to ensure that they match with the requirements of the network control application.

Accordingly, we present OFLOPS, an open source software framework that permits the development of tests for OpenFlow-enabled switches. These tests can evaluate the capabilities and bottlenecks between the forwarding engine of the switch and the remote control application. For high benchmarking precision and flexibility in test development, OFLOPS combines hardware instrumentation with an extensible software framework.

We use OFLOPS to evaluate current OpenFlow-enabled switches and make the following observations: *(i)* The per-packet forwarding latency depends on the active set of OpenFlow actions as well as the specific firmware implementation of the switch. *(ii)* Current OpenFlow implementations differ substantially in the rates at which flow entries can be updated and monitored. *(iii)* For accurate measurements of OpenFlow command completion times one has to instrument both the data plane as well as the control plane.

## 6.1 Introduction

OpenFlow, a protocol instance of Software Defined Networking (SDN), gives access deep within the network forwarding plane while providing a common and simple API for controlling switches (see Section 2.3). However, switch-side implementation details are left to the discretion of each vendor. This leads to an expectation of diverse strengths and weaknesses across the existing OpenFlow implementations, which motivates us to study these differences. To do that, we build and release to the public a comprehensive OpenFlow switch testing framework.

We present OFLOPS[1] (short for *OpenFlow Operations Per Second*), a tool that enables the rapid development of use-case tests for both hardware and software OpenFlow switch implementations. Using OFLOPS, developers can simulate specific usage scenarios and understand the impact of the particular OpenFlow switch implementations on the achieved network performance. We use OFLOPS to test publicly-available OpenFlow software implementations as well as several OpenFlow-enabled commercial hardware switches, and report our findings about their varying performance characteristics. To better understand the behavior of the tested OpenFlow implementations, OFLOPS combines measurements from the OpenFlow control channel with data-plane measurements. To ensure sub-millisecond-level accuracy of our measurements, we bundle the OFLOPS software with specialized hardware in the form of the NetFPGA platform [91]. Note that if the tests do not require sub-millisecond-level accuracy, commodity hardware can be used instead of the NetFPGA [5].

The rest of this chapter is organized as follows. We present the design of the OFLOPS framework in Section 6.2. We describe the measurement setup in Section 6.3 and report on our measurement results in Section 6.4. We discuss related work in Section 6.5 and summarize in Section 6.6.

## 6.2 OFLOPS architecture

Measuring OpenFlow switch implementations is a challenging task in terms of characterization accuracy, noise suppression and precision. Performance characterization is not trivial as most OpenFlow-enabled switches provide rich functionality but do not disclose implementation details. In order to understand the performance impact of an experiment, multiple input measurements must be monitored concurrently. Furthermore, measurement noise minimization can only be achieved through proper design of the measurement platform. Current controller designs, like [55], target production networks and thus are optimized for throughput maximization and programmability, but incur high measurement inaccuracy. Finally, high precision measurements after

[1]OFLOPS (GPL licenced): `http://www.openflow.org/wk/index.php/Oflops`

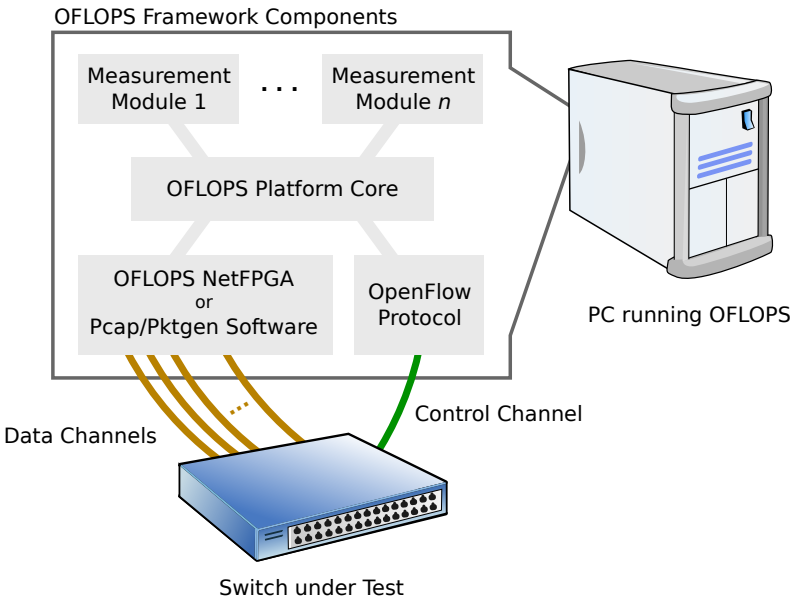



**Figure 6.1:** OFLOPS design schematic.

a point are subject to loss due to unobserved parameters of the measurement host, such as OS scheduling and clock drift.

The OFLOPS design philosophy is to enable seamless interaction with an OpenFlow-enabled switch over multiple data channels without introducing significant additional processing delays. The platform provides a unified system that allows developers to control and receive information from multiple control sources; data and control channels as well as SNMP to provide specific switch-state information. For the development of measurement experiments with OFLOPS, the framework provides a rich, event-driven API, that allows developers to handle events programmatically in order to implement and measure custom OpenFlow controller interactions. The current version is written predominantly in C. Experiments are compiled as shared libraries and loaded at run-time as specified in a simple configuration file, which also defines parameters to the experiments. A schematic of the platform is presented in Figure 6.1. Details of the OFLOPS programming model can be found in the API manual[2].

The platform is implemented as a multi-threaded application, to take advantage of modern multi-core environments. To reduce latency, our design avoids concurrent access controls: We leave any concurrency-control complexity to individual module implementations. OFLOPS consists of the following five threads, each one serving specific type of events:

**(1) Data Packet Generation** controls the data plane traffic generators.

[2]OFLOPS manual: `http://www.openflow.org/wk/images/3/3e/Manual.pdf`

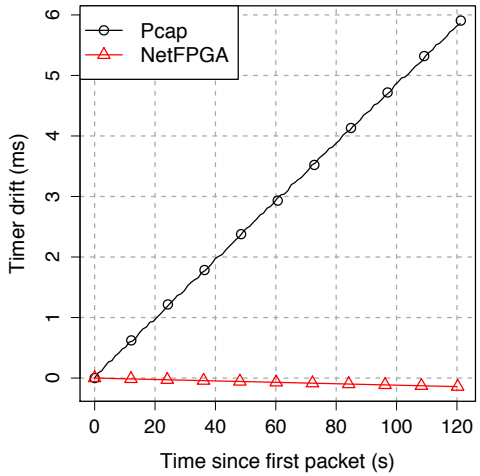



**Figure 6.2:** Evaluating timestamping precision using a DAG card.

**(2) Data Packet Capture** captures and pushes data plane traffic to the modules.

**(3) Control Channel** translates OpenFlow messages to control events.

**(4) SNMP Channel** performs asynchronous SNMP polling.

**(5) Time Manager** manages timed events scheduled by the measurement modules.

OFLOPS provides the ability to control concurrently multiple data channels to the switch. By embedding the data channel within the platform, we can measure the impact of the OpenFlow-based control plane on the data plane. To enable our framework to run on a variety of heterogeneous systems, we integrate support for multiple packet generation and packet capturing mechanisms. For the packet generation functionality, OFLOPS currently supports three mechanisms; *(i)* user-space, *(ii)* kernel-space via the pktgen module [108], and *(iii)* hardware-accelerated through an extension of the design of the NetFPGA Stanford Packet Generator [31]. For the packet capturing and timestamping, the platform supports both the pcap library as well as our NetFPGA design. Each approach provides different levels of timestamping precision and a different performance in generating packets.

A comparison of the precision of the traffic capturing mechanisms is presented in Figure 6.2. In this experiment we use probes of small packets at a constant rate at 100 Mbps for a two minute period. The probe is duplicated, using an optical wiretap with negligible delay, and sent simultaneously to OFLOPS and to a DAG card [33]. In the figure, we plot the relative difference of the timestamps from both of the OFLOPS timestamping mechanisms (pcap and NetFPGA) and the DAG card, for each packet. From the figure, we see that the pcap timestamps drift by 6 milliseconds after 2 minutes. On the other hand, the NetFPGA timestamping mechanism has a smaller drift at the level of a few microseconds during the same period.

| Switch | CPU | Flow table size |
|---|---|---|
| Switch1 | PowerPC 500MHz | 3072 mixed flows |
| Switch2 | PowerPC 666MHz | 1500 mixed flows |
| Switch3 | PowerPC 828MHz | 2048 mixed flows |
| OpenVSwitch | Xeon 3.6GHz | 1M mixed flows |
| NetFPGA | DualCore 2.4GHz | 32K exact & 100 wildcard |

**Table 6.1:** OpenFlow switch details.

## 6.3 Measurement setup

The number of OpenFlow-enabled switches has increased recently, with switch and router vendors providing experimental OpenFlow support through prototype and evaluation firmware. At the end of 2009, the OpenFlow protocol specification was released in its first stable version 1.0 [110], the first recommended version to be implemented by vendors for production systems. Consequently, vendors did proceed on maturing their prototype implementations, offering production-ready OpenFlow-enabled switches today. Using OFLOPS, we evaluate OpenFlow-enabled switches from three different switch vendors. Vendor 1 has production-ready OpenFlow support, whereas vendors 2 and 3 at this point only provide an experimental OpenFlow switch firmware. The set of switches that we consider in this work provides a representative but not exhaustive sample of available OpenFlow-enabled top-of-rack-grade switching hardware. Details with regards to the switch CPU and the size of the flow table (*i.e.*, FIB) of the switches are provided in Table 6.1. In this table, we distinguish between three types of flow entries; *(i)* wildcard flows, which can have *don't-care*-bits, *e.g.*, for matching on IP prefixes; *(ii)* exact-match flows, with all bits in a packet header field defined; and *(iii)* mixed flows, which can be either of the two.

OpenFlow is not limited to hardware. The OpenFlow protocol reference implementation is a software switch, OpenVSwitch [117]. OpenVSwitch provides a replacement for the poor-performing Linux bridge, a crucial component for operating system virtualization. In addition, several hardware switch vendors use OpenVSwitch as the basis for the development of their own OpenFlow-enabled firmware. Thus, the mature software implementation of the OpenFlow protocol is ported to commercial hardware, making certain implementation bugs less likely to (re)appear. In this chapter, we study OpenVSwitch alongside our performance and scalability study of hardware switches. Finally, in our comparison we include the OpenFlow switch design for the NetFPGA platform [99]; a working implementation of the OpenFlow protocol, limited though in capabilities due to hardware platform limitations.

In order to conduct our measurements, we run OFLOPS on a dual-core 2.4GHz Xeon server equipped with a NetFPGA card. For all of our experiments we utilize the NetFPGA-based packet generation and packet capturing mechanisms. 1Gbps control and data channels are connected directly to the tested switches. We measure

| Mod. type | Switch1 med/sd/loss % | OpenVSwitch med/sd/loss % | Switch2 med/sd/loss % | Switch3 med/sd/loss % | NetFPGA med/sd/loss % |
|---|---|---|---|---|---|
| Forward | 4/0/0 | 35/13/0 | 6/0/0 | 5/0/0 | 3/0/0 |
| MAC addr. | 4/0/0 | 35/13/0 | 302/727/88 | -/-/100 | 3/0/0 |
| IP addr. | 3/0/0 | 36/13/0 | 302/615/88 | -/-/ 100 | 3/0/0 |
| IP ToS | 3/0/0 | 36/16/0 | 6/0/0 | -/-/100 | 3/0/0 |
| L4 port | 3/0/0 | 35/15/0 | 302/611/88 | -/-/100 | 3/0/0 |
| VLAN pcp | 3/0/0 | 36/20/0 | 6/0/0 | 5/0/0 | 3/0/0 |
| VLAN id | 4/0/0 | 35/17/0 | 301/610/88 | 5/0/0 | 3/0/0 |
| VLAN rem. | 4/0/0 | 35/15/0 | 335/626/88 | 5/0/0 | 3/0/0 |

**Table 6.2:** Packet header modification times in $\mu$s and packet loss. Processing times $>10\mu$s indicate handling in software.

the processing delay incurred by the NetFPGA-based hardware design to be a near-constant 900 nsec, independent of the probing rate.

## 6.4 Results

In this section we present a set of tests performed by OFLOPS to measure the behavior and performance of OpenFlow-enabled switches, by simulating a set of simple usage scenarios. These tests target *(i)* the OpenFlow packet processing actions, *(ii)* the flow table update rate along with its impact on the data plane, *(iii)* the monitoring capabilities provided by OpenFlow, and *(iv)* the impact of interactions between different OpenFlow operations.

### 6.4.1 Packet modifications

The OpenFlow 1.0 specification [110] defines ten packet modification actions which can be applied on incoming packets. Available actions include the modification of MAC, IP, and VLAN values, as well as transport-layer fields; flows can contain any combination of those actions. The left column of Table 6.2 lists the packet fields that can be modified by an OpenFlow-enabled switch. These actions are commonly used by network devices such as IP routers (*e.g.*, rewriting of source and destination MAC addresses) and NAT gateways (rewriting of IP addresses and ports). Existing network equipment is tailored to perform a subset of these operations, usually in hardware to sustain line rate. On the other hand, how these operations are to be used is yet to be defined for new network primitives and applications, such as network virtualization, mobility support, or flow-based traffic engineering.

To measure the time taken by an OpenFlow implementation to modify a packet header field, we generate a stream of 100 bytes-sized UDP packets via the NetFPGA

card, at a constant rate of 100Mbps (approx. 125k packets per second). This rate is high enough to give statistically significant results in a short period of time. The flow table is initialized with a flow that matches the probe traffic and applies a specific action on all probe packets. The flow processing delay is then calculated using the two NetFPGA-provided packet timestamps of *(i)* when the packet is sent from the NetFPGA to the switch under test, and *(ii)* when the packet is received back by the NetFPGA from the tested switch.

Accordingly, we evaluate the flow processing times with different packet field modifications and report the results in Table 6.2. The table provides the median flow processing times along with its standard deviation and percentage of lost packets.

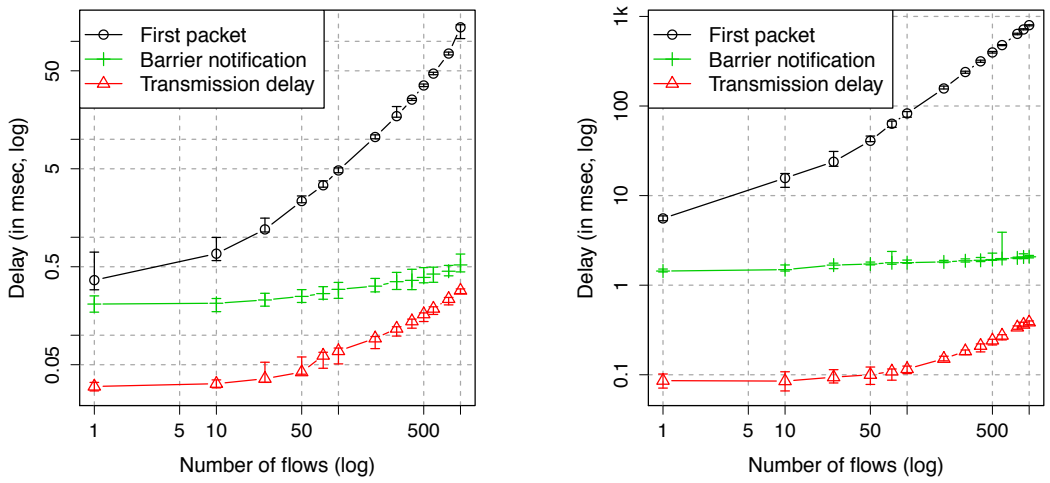
We observe significant differences in the performance of the hardware switches due in part to the way each handles packet modifications. Switch1, with its production-grade implementation, handles all modifications in hardware; this explains its low flow processing time of only 3 to 4 microseconds. On the other hand, Switch2 and Switch3 each run experimental firmware providing only partial hardware support for OpenFlow actions. Switch2 uses the switch CPU to perform some of the available field modifications, resulting in two orders of magnitude higher flow processing times and variance. Switch3 follows a different approach: All packets of flows with actions not supported in hardware are silently discarded. The performance of the OpenVSwitch software implementation lies between Switch1 and the other hardware switches. OpenVSwitch fully implements all OpenFlow actions. However, hardware switches outperform OpenVSwitch when the flow actions are supported in hardware.

We conduct a series of further experiments with variable numbers of packet modifications as flow actions and probe rates (results not shown). We observe, that the combined processing time of a set of packet modifications is equal to the highest processing time across all individual actions in the set. For higher probe rates, we report that all software-implemented actions increase latency and packet loss, while hardware-supported actions are not affected.

### 6.4.2 Flow table updates

The flow table is a central component of an OpenFlow-enabled switch and is the equivalent of a Forwarding Information Base (FIB) in routers. Given the importance of fast FIB updates in commercial routers, *e.g.*, to reduce the impact of routing dynamics onto the data plane, the FIB update processing time of commercial routers provide useful reference points and lower bounds for the time to update a flow entry on an OpenFlow switch. The time to install a new entry on commercial routers has been reported in the range of a few hundreds of microseconds [133].

OpenFlow provides a mechanism to define barriers between sets of commands; the `barrier` command. According to the OpenFlow specification, the barrier command




(a) OpenVSwitch (log-log scale)

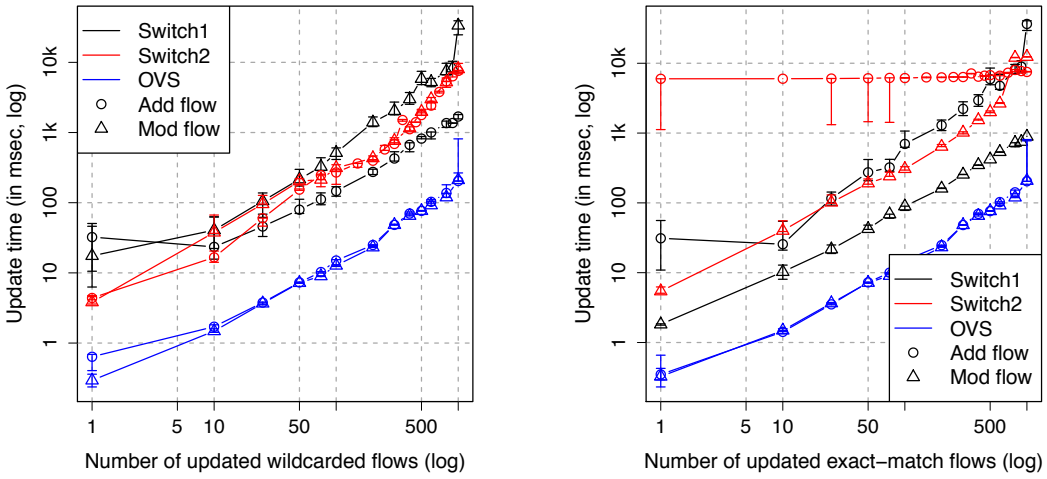
(b) Switch1 (log-log scale)

**Figure 6.3:** Flow entry insertion delay as obtained by using the `barrier` notification and as observed at the data plane.

requests a notification about when a set of OpenFlow operations – all that were issued prior to the barrier – has been completed. Further, the switch has to complete the set of operations issued prior to the barrier before executing any further operation. If the OpenFlow implementations comply with the specification, we expect to receive a barrier notification for a flow modification once the flow table of the switch has been updated, implying that the change can be seen from the data plane.

We check the behavior of the tested OpenFlow implementations with respect to the time it takes to install a new flow table entry, measured with the help of the `barrier` command, and observe variations among them. For OpenVSwitch and Switch1, Figure 6.3 shows the time it takes to install a set of entries into the flow table. The NetFPGA-based switch results (not reported) are similar to those of Switch1, while Switch2 and Switch3 are not reported as the `barrier` command is not supported by their firmware. For this experiment, OFLOPS relies on a stream of packets of 100 bytes at a constant rate of 10Mbps on the data plane that targets the newly installed flows in a round-robin manner. The probe achieves sufficiently low inter-packet periods in order to measure accurately the flow insertion time, without any lost packet.

In Figure 6.3, we show three different times. The *barrier notification* is derived by measuring the time between when the first insertion command is sent by the OFLOPS controller and the time the barrier notification is received by the PC. The *transmission delay* is the time between the first and last flow insertion command. With *first packet* we report the time between when the first insertion command is issued to the switch, and when the first packet matching the last of the (newly)




(a) Wildcarded flow entries.

(b) Exact-match flow entries.

**Figure 6.4:** Flow table update times as observed from the data plane.

inserted flow entries is observed on the data plane. For each configuration, we run the experiment 100 times and Figure 6.3 shows the median result as well as the $90^{th}$ percentiles.

From Figure 6.3, we observe that even though the *transmission delay* for sending flow insertion commands increases with their number, this time is negligible when compared with data plane measurements (*first packet*). Notably, the *barrier notification* measurements are almost constant, increasing only as the *transmission delay* increases (difficult to discern on the log-log plot) and, critically, the barrier notification is received *before* any respective packet is seen on the data plane (*first packet*). This implies that the way the `barrier` command is implemented in our switch implementations does not reflect the time when the actual flow table has been updated.

In these results we demonstrate how OFLOPS can compute per-flow overheads. We observe that the flow insertion time for Switch1 starts at 5ms for a single entry, but converges toward an approximate overhead of 1ms per inserted entry as the number of insertions grows.

### Flow table update times compared

We now distinguish between flow insertions and the modification of existing flows. With OpenFlow, a flow entry can perform exact packet matches or use wildcards to match a range of values, so we further distinguish between exact-match and wildcarded flow entries.

Figure 6.4 shows the flow update time as a function of the number of updated entries, according to the *first packet* metric as introduced earlier. These results show, that for software switches that keep all flow entries in memory, the type of entry or the kind of update does not make a difference in the flow update time.

We observe in Figure 6.4(a), that both Switch1 and Switch2 sometimes take more time to modify existing wildcarded flow entries compared to adding new wildcarded flow entries: For Switch1, this is the case for more than 10 new entries, while for Switch2 this occurs when inserting 10 to 50 new entries. After discussing this issue with the vendor of Switch2, we came to the following conclusion: As the number of TCAM entries increases, updates become more complex as they typically requires re-ordering of existing entries.

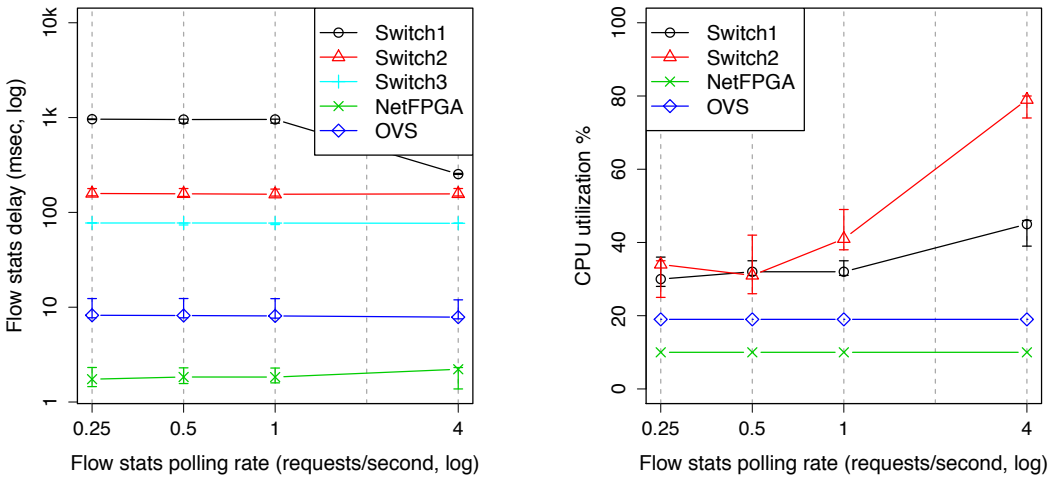
In the case of exact-match flows, the results for Switch1 in Figure 6.4(b) are close to the wildcarded ones, but twisted: It takes more time for Switch1 to insert a new flow table entry than to modify an existing one. We observe a surprising behavior with Switch2: For any number of exact-match flows inserted in our tests, we observe a near constant processing time for Switch2. This may be caused by switch firmware-internal queueing and batching of incoming flow insertion requests. However, updating already existing exact-match flows on Switch2 is similar to updating wildcarded flows. This suggests that the switch firmware implements exact-match flow insertions differently than flow modifications and wildcarded flow insertions.

Clearly, the results depend both on the flow entry type and the switch implementation. For example, exact match entries may be handled through a hardware or software hash table. Whereas wildcarded entries, requiring support for variable length table lookups, must be handled by specialized memory modules, such as TCAM. With the many possible ways of implementation, the flow insertion times reported in Figure 6.4 are not generalizable, but rather depend on the type of flow entry and the particular switch implementation.

### 6.4.3 Traffic statistics

The use of OpenFlow as a monitoring platform has already been suggested for the applications of traffic matrix computation [142, 10] and identifying large traffic aggregates [76]. To obtain direct information about the state of the traffic received by an OpenFlow switch, the OpenFlow protocol provides a mechanism to query traffic statistics, either on a per-flow basis or across aggregates matching multiple flows. The flow statistics give packet as well as byte counts.

We now test the performance of the traffic statistics reporting mechanism of our OpenFlow implementations under test. Using OFLOPS, we install flow entries that match packets sent on the data path. Simultaneously, we start sending flow statistics requests to the switch. Throughout the experiment we record *(i)* the delta time between a flow statistics request and response, *(ii)* the number of packets into which




(a) Reply time.

(b) CPU utilization.

**Figure 6.5:** Time to receive a flow statistic (median) and corresponding CPU utilization.

the switch packs the data of the statistics response, and *(iii)* the departure and arrival timestamps of the data plane probe packets.

Figure 6.5(a) reports the time it takes to receive a flow statistics reply for each switch, as a function of the request rate. Figure 6.5(b) reports the switch-CPU utilization as a function of the flow statistics inter-request time. CPU utilization statistics are retrieved using SNMP. Switch3 is excluded for lack of SNMP support. Despite the rate of statistics requests being modest, we observe surprisingly high CPU utilization results for even a few queries per second being sent.

From the flow statistics reply times, we observe that all switches have (near-)constant response delays, *i.e.*, the delay itself relates to the type of switch. As expected, software switches have faster response times than hardware switches. This reflects the availability of the information in system memory without the need to poll (multiple) hardware counters, and the fact that they run on more powerful CPU and system components compared to the comparatively limited embedded systems in a hardware switch (see Table 6.1). Despite the (near-)consistent response times for Switch1 and Switch2, we observe their CPU utilization increasing proportionally with the rate of requests. We see indications for different implementations of the statistics handling in Switch1 and Switch2: Switch2 has a high CPU utilization, answering flow-stats requests as fast as possible, while Switch1 applies a pacing mechanism on its replies, avoiding over-loading of its CPU. Specifically, at low polling rates, Switch1 splits its answer across multiple TCP segments: each segment containing statistics for a single flow. As the probing rate increases, the switch aggregates multiple flows into a single TCP segment. Finally, neither software nor NetFPGA switches see an impact of the flow-stats rate on their CPU, thanks to their significantly more powerful PC CPUs

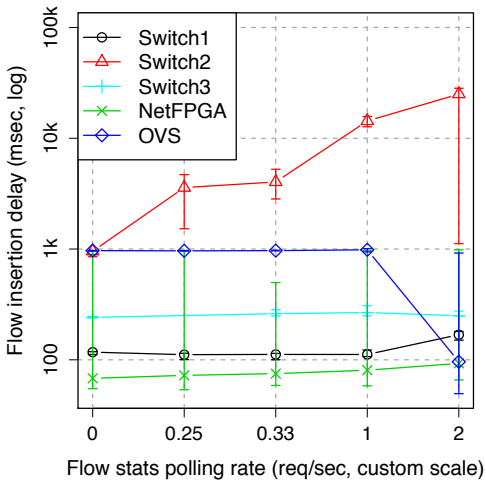



**Figure 6.6:** Delay when updating the flow table while polling for traffic statistics.

(Table 6.1).

### 6.4.4 Performance interactions

An advanced feature of the OpenFlow protocol is its ability to provide applications with, *e.g.*, new flow arrival notifications from the network, while simultaneously providing fine-grained control of the forwarding process. This permits applications to adapt in real time to the requirements and load of the network [60, 154]. Under certain OpenFlow usage scenarios, *e.g.*, the simultaneous querying of traffic statistics and modification of the flow table, which is the case in our SDN Router, we now perform measurements to understand the performance impact that different commands may have on each other.

Through this scenario, we extend Section 6.4.2 to show how the mechanisms of traffic statistics extraction and flow table manipulation interact. Specifically, we initialize the flow table with 1024 exact match flows and measure the delay to update a subset of 100 flows. Simultaneously, the measurement module polls the switch for full table statistics at a constant rate. The experiment uses a constant rate 10Mbps packet probe to monitor the data path, and polls every 10 seconds for SNMP CPU utilization values.

In this experiment, we control the rate at which to poll for statistics, and we keep track of the time it takes for a modified set of flows to become active at the data plane. For each statistics polling rate, we repeat the experiment 50 times, plotting the median and the $90^{th}$ percentile in Figure 6.6. We observe that, for low polling rates, implementations have a near-constant insertion delay, comparable to the results of Section 6.4.2. For higher probing rates on the other hand, Switch1 and Switch3 do

not differ much in their behavior. In contrast, Switch2 exhibits a noteworthy increase in the insertion delay, which can be explained by the increase in CPU utilization incurred by handling requests for flow statistics (Figure 6.5(b)). Finally, OpenVSwitch exhibits a marginal decrease in the median insertion delay and at the same time an increase in its variance. We believe this behavior is caused by OpenVSwitch-internal scheduling and batching of incoming OpenFlow commands, which can vary with the particular arrival-pattern of the coinciding commands.

## 6.5 Related work

OpenFlow is increasingly adopted, both by hardware vendors as well as by the research community, examples are our SDN Router (Chapter 3), or [60, 155, 134]. Yet, there have been few performance studies: To our knowledge, OFLOPS is the first attempt to develop a framework that is able to provide detailed measurements for the OpenFlow implementations. Curtis et al. [32] discuss some design limitations of the protocol when deployed in large network environments. We consider OFLOPS, alongside [52], as one of a new generation of measurement systems that, like the intelligent traffic and router evaluators [74, 3], go beyond simple packet capturing.

## 6.6 Summary

This chapter presents OFLOPS, a modular tool to test the capabilities and performance characteristics of OpenFlow-enabled software and hardware switches. OFLOPS is an extensible software framework which leverages advanced hardware devices (NetFPGA) for measurement accuracy and performance. We use OFLOPS to evaluate five different OpenFlow switch implementations in terms of their key OpenFlow-related performance aspects, *e.g.*, updating flow entries and querying for statistics.

We take advantage of the ability of OFLOPS to perform combined data and control plane measurements to quantify accurately the completion times of OpenFlow commands from a data plane perspective. We identify considerable variations among the tested OpenFlow switches. In addition, we observe that across all tested switches which have support for the `barrier` command – OpenFlow's command completion notifier – the functionality is not correctly implemented. Finally, we report that the flow table statistics monitoring capabilities of existing hardware switches have limitations in their ability to sustain even modest rates of requests. Also, these monitoring operations can lower the performance of other OpenFlow commands, such as flow table insertions, when they coincide.

# 7

# Anatomy of a Large European IXP

We have shown that a well-founded understanding of the traffic in the Internet as well as the dynamics in the Internet's routing control plane are of considerable value for router designs. This motivates us to perform an extensive series of passive measurements to study the diversity in traffic characteristics that routers are facing and their peering relationships at one of the largest Internet Exchange Points (IXPs) worldwide.

The largest IXPs carry on a daily basis traffic volumes in the petabyte range, similar to what some of the largest global ISPs reportedly handle. This little-known fact is due to a few hundreds of member ASes exchanging traffic with one another over the IXP's infrastructure. This chapter reports on a first-of-its-kind and in-depth analysis of one of the largest IXPs worldwide based on nine month's worth of sFlow records collected at that IXP in 2011.

A main finding of our study is that the number of actual peering links at this single IXP exceeds the number of total AS links of the peer-peer type in the entire Internet known as of 2010! To explain such a surprisingly rich peering fabric, we examine in detail this IXP's ecosystem and highlight the diversity of networks that are members at this IXP and connect there with other member ASes for reasons that are similarly diverse, but can be partially inferred from their business types and observed traffic patterns. In the process, we investigate this IXP's traffic matrix and illustrate what its temporal and structural properties can tell us about the member ASes that generated the traffic in the first place. While our results suggest that these large IXPs can be viewed as a microcosm of the Internet ecosystem itself, they also argue for a re-assessment of the mental picture that our community has about this ecosystem.

## 7.1 Introduction

The basic role of Internet Exchange Points (IXPs) dates back to the establishment of Network Access Points (NAPs) as part of the decommissioning of the National Science Foundation Network (NSFNET) around 1994/95, a carefully orchestrated plan for transitioning the NSFNET backbone service to private industry. The vehicle that evolved in support of this transition was a set of four NAPs (*i.e.*, MAE-East, Sprint NAP, PacBell NAP, and Ameritech NAP) that acted as connection points for the commercial carriers that were vying for offering backbone services (*e.g.*, MCInet, Sprintlink, AGIS) and ensured that the network would remain connected at the top level once the NSFNET was retired.

Over the past 15 years, as the Internet grew by leaps and bounds by any imaginable metric, the original four NAPs were replaced by a steadily increasing number of modern IXPs. Originally providing largely just the bare necessities for supporting easy interconnection between their member ASes (*e.g.*, physical space, caches, cabling, power, A/C, or secure access), IXPs themselves have evolved over time. Numbering now more than 300 worldwide [47], many of these IXPs are offering an array of different services that rely on advances in networking technology (*e.g.*, VLANs or MPLS), exploit existing routing protocols in innovative ways (*e.g.*, use of BGP for prefix-specific peering), or provide the economic incentives for an ever-increasing number of networks to join as paying members (*e.g.*, remote peering offerings, support for IXP resellers).

In fact, large IXPs such as AMS-IX, situated in Amsterdam, and DE-CIX, in Frankfurt, offer high-end Service Level Agreements (SLAs) to their members that cover not only the initial provisioning and daily availability of a member's port(s) but also the level of performance of key service parameters. Such innovation on parts of the IXPs has enabled them to compete more directly with the traditional carriers and has led to today's environment where some of the largest IXPs worldwide (*e.g.*, AMS-IX, DE-CIX, LINX, MSK-IX) reportedly carry on a daily basis similar amounts of traffic as some large ISPs (*e.g.*, AT&T, Deutsche Telekom[1]). The traffic volumes at those IXPs are generated by some 300-500 networks that cover the whole spectrum of players in today's Internet marketplace. While there may be regional differences in how extensive in coverage or aggressive in the uptake of new members IXPs are, the critical role they have played in the Internet ecosystem has until recently gone largely unnoticed by the research community whose focus has traditionally been on large carriers and large content.

This chapter reports on a first-of-its-kind measurement-based study of a large IXP and complements a body of existing literature that has focused squarely on large carriers or large content. To this end, we analyze a unique dataset consisting of nine

[1] AT&T reports carrying 28.9 petabytes of data traffic on an average business day [7], Deutsche Telekom reports 422 petabytes per month corresponding to 14 petabytes per day on average [35].

months' worth of anonymized sFlow records that were collected at one of the largest IXPs in Europe, and worldwide, in 2011. We present our dataset, describe our data analysis and illustrate how our observations and findings contribute to an improved understanding of

- the AS-level Internet; that is, the structure and dynamics of the Internet as a network of networks or ASes;
- the Internet peering ecosystem; that is, the practices and economic incentives that drive the market for Internet interconnection and peering between ASes; and
- the Internet inter-domain traffic; that is, the quantity and quality of the traffic exchanged among ASes.

Specifically, the main contributions of our work fall into three categories. First, we show that this large IXP exhibits a surprisingly rich peering fabric in support of the many business objectives of its members. In particular, in terms of AS links of the peer-peer type that are typically established among member AS pairs we show that this IXP has close to 400 members which have established some 67 % (or more than 50,000) of all possible such peerings and use them for exchanging some 10 PB of IP traffic daily. *To put this number in perspective, note that as of 2010, the number of inferred AS links of the peer-peer type in the Internet was reported to be around 40,000 – less than what we observe at this particular IXP alone!*

To explain this startling difference between the number of peerings observed at this IXP (*i.e.*, ground truth) and the number of known peer-peer AS links Internet-wide, we show which portion of the IXP's actual peering matrix is and is not visible when relying on the publicly available BGP data that has formed the basis for much of the past and recent work on inferring the Internet's AS-level connectivity. To further highlight this issue, we illustrate why a large portion of this IXP's actual peering matrix remains invisible even to measurement efforts that go beyond the current state-of-the-art, either with respect to BGP-derived data or traceroute-based measurements, or a combination of the two. By combining our finding of an enormously rich peering fabric among the members of this IXP with an accurate picture of their upstream connectivity (*i.e.*, customer-provider relationships) that is reportedly quite accurate [21, 107], we are able to reconcile the traditionally-assumed hierarchical structure of the Internet with recent claims about a flattening of that structure. Indeed, while the traditional tier-structure of the Internet is still recognizable and can be largely recovered, the observed rich peering fabric at this IXP enables connectivity among networks of all different types and is essentially agnostic of any tier structure. Thus, at least as far as connectivity for the part of the Internet that involves this IXP is concerned, the observed IXP-related peerings provide a myriad of shortcuts and essentially complement any perceived or real hierarchical structure. Note that this realization of a much more elaborate interconnect structure than previously assumed says nothing about how these existing peering links are used to carry traffic.

To address the issue of how much and what type of traffic is traversing this IXP's infrastructure via public peering links, our second main contribution consists of an in-depth analysis of the available sFlow records. By examining the members of this large IXP and with which other members they peer and exchange traffic with, we highlight that large IXPs are a microcosm of the Internet as a whole in terms of types of networks, business relationships, or traffic. We observe various types of networks, from tier-1 to regional and local ISPs, large/medium/small content, host and service providers, content distribution networks (CDN), and a spectrum of academic and enterprise networks. With which other networks these networks establish peering connections at this IXP and exchange what type of traffic has nothing to do with their standing within the traditionally-assumed tier structure, but is largely dictated by economic considerations and business objectives and reflects a wealth of reasons and incentives for why the different types of networks make use of the various service offerings at this IXP.

Our third contribution is motivated by the existing large body of literature on traffic engineering for ISPs that relies critically on understanding how routing policies internal and external to the ISP affect the traffic flow over the ISP's infrastructure and ultimately result in what is commonly referred to as the ISP's intra-domain traffic matrix. In contrast to large ISPs, an IXP's infrastructure as well as the logical connectivity and routing at an IXP are significantly less complex and make it relatively easy to compute an IXP's traffic matrix and use it as a key input to IXP-specific methods in support of an efficient and effective management and operation of its infrastructure in a dynamic IXP marketplace. Despite the significant amount of traffic that the largest IXPs carry, especially when compared to the largest global ISPs, we are not aware of any research paper that provides an even remotely realistic picture of the traffic traversing an IXP. To fill this void, this chapter is the first to obtain and characterize the traffic matrix of one of the largest IXPs, with a particular focus on traffic variability and dependency over time and in space (*i.e.*, across ASes), and application mix. Knowledge of an IXP's traffic matrix and its properties in conjunction with emerging new routing strategies at IXPs is critical for exploring what-if scenarios that an IXP may want to run before announcing new services, encouraging members to send more traffic through the IXP, or for deciding when to upgrade its infrastructure and how.

The remaining part of the chapter is structured as follows. In Section 7.2, we provide information about the large European IXP that we study and discuss the available sFlow records. Using this data, we examine in Section 7.3 the IXP's peering fabric and contrast our findings with results that rely exclusively on data that have been collected without the active participation of the IXP. Studying the members at this IXP as well as how and why they connect to other member ASes, we provide in Section 7.4 a detailed account of this IXP's ecosystem. In Section 7.5, we focus on the IXP's traffic matrix and report on some of its key characteristics. We elaborate on some of the implications of our findings and the new challenges they pose in

Section 7.6, discuss related work in Section 7.7, and conclude in Section 7.8 with a summary of our main findings.

## 7.2 A large European IXP

In this section, we provide a high-level overview of the infrastructure and operations of a large European IXP. We discuss the data that we obtained from this IXP, and present some basic facts about the member ASes of this IXP and about its overall traffic.

### 7.2.1 IXP overview

The main business model of an IXP is to operate and manage a physical infrastructure in support of public and private Internet interconnection. In this chapter, we focus on the public part of an IXP's infrastructure where the IXP's revenues derive mainly from selling network interfaces or ports to customer networks (*i.e.*, ASes) and supporting different types of interconnection arrangements. Such customer networks are referred to as member ASes. A member AS has the advantage to gain network connectivity to all other members of the IXP. However, interconnection arrangements reflect bi-lateral agreements[2] between a pair of member ASes, and these networks may want to impose certain conditions to ensure that they connect only to certain other networks or connect with them in ways that reflect their business model and support their market strategies.

What makes the IXP substrate of the Internet (*i.e.*, the IXPs, their member ASes, and the peerings among these member ASes at those IXPs) such a vibrant marketplace is that the incentives for networks to become members at such public peering platforms are as diverse as the growing number of increasingly diverse ASes. For example, a CDN interested in optimizing its performance while keeping its cost low might want to choose an open peering policy to encourage direct and settlement-free traffic exchange at an IXP with as many networks as possible. On the other hand, large ISPs are likely to be interested in establishing peering relationships with other ISPs of about the same size. To achieve this objective, they may want to base their peering decision on a selective peering policy that allows them to deny peering with small ISPs, thus retaining them as paying customers in customer-provider type interconnection arrangements that are more lucrative. Transit networks have yet different objectives for using an IXP – they look at an IXP as a point of sale of their upstream connectivity offerings. In general, the larger the number of member ASes at an IXP, the more attractive that IXP is as a peering platform. This explains to a large degree the

[2]For the purposes of this chapter, we view multi-lateral peering agreements as a collection of bi-lateral peering agreements.

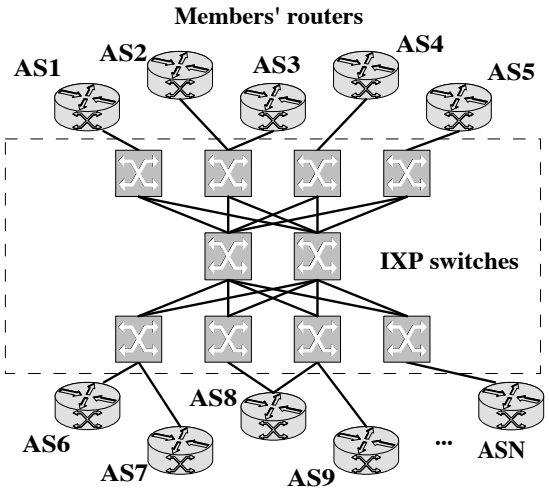



**Figure 7.1:** A typical IXP architecture.

high level of innovation that the IXP marketplace has experienced in the process of becoming a vital component of the Internet ecosystem.

### 7.2.2 IXP infrastructure and data

Figure 7.1 illustrates a high-level overview of the architecture of our IXP. Although complex to maintain and scale, the infrastructure of this large IXP is typical of large IXPs in general, and the IXP's operation can be described in simple terms. The IXP provides a layer-2 switching fabric and each of the member ASes connects its access router to that switching fabric. When a pair of member ASes decides to peer at the IXP, they establish a BGP session between their access routers which, in turn, enables the exchange of IP traffic over this peering link across the IXP's infrastructure.

The volume and properties of the traffic exchanged at an IXP depend on the number of member ASes, the location and scope of the activities of the IXP, the IXP's service offerings, and if the IXP operates for profit or as a non-profit organization [47]. In this chapter, we consider the traffic that is exchanged over the public peering fabric supported by the switching infrastructure of the IXP. In particular, for this study, we rely on nine months' worth of continuous sFlow [118] records that were collected in 2011 at the IXP's infrastructure using a random sampling of 1 out of 16k packets. Our sFlow records capture the first 128 bytes of each sampled packet, thus giving us access to the IP and TCP headers. The sFlow capturing process includes an anonymization step in which IP addresses are scrambled while maintaining prefix consistency [48].

The efforts we made to assess the quality of the available sFlow records included checking for sampling bias and identifying and filtering out less than 1 % of the to-

| | Apr 25<br>May 1 | Aug 22<br>Aug 28 | Oct 10<br>Oct 16 | Nov 28<br>Dec 4 |
|---|---|---|---|---|
| Identified member ASes | 358 | 375 | 383 | 396 |
| Router IPs | 426 | 445 | 455 | 474 |
| MAC addresses | 428 | 448 | 458 | 474 |
| Tier-1 | 13 | 13 | 13 | 13 |
| Tier-2 | 281 | 292 | 297 | 306 |
| Leaf | 64 | 70 | 73 | 77 |
| Countries of member ASes | 43 | 44 | 45 | 47 |
| Continents of member ASes | 3 | 3 | 3 | 3 |
| Average packet rate (Mpps) | 142 | 150 | 166 | 174 |
| Average bandwidth (Gbps) | 838 | 863 | 954 | 992 |
| Daily avg volume (PB) | 9.0 | 9.3 | 10.3 | 10.7 |

**Table 7.1:** Overview of the IXP sFlow datasets.

tal traffic that was immaterial for our study. For example, since sFlow sampling is performed simultaneously and independently by multiple switches within the IXP's infrastructure, there may exist a bias toward such flows that traverse multiple sampling points. When counting the number of different sFlow probes that capture packets exchanged between the same pair of member router interfaces (MAC addresses), we found that more than 99 % of these flows were only sampled by a single probe, providing hard evidence that our data is not corrupted by this sampling bias. As for immaterial traffic, we filtered out all traffic contributed by the IXP's management machines (*e.g.*, route servers) as well as broadcast and multicast traffic, except for ARP packets. Finally, we also eliminated all IPv6 traffic as it constitutes less than 1 % of the overall traffic (in bytes or packets) at this IXP.

### 7.2.3 IXPs: a moving target

Studying one of the largest IXPs means chasing a moving target. Large IXPs present a changing environment, with a number of different dynamic factors acting on different time scales. Over large time scales (*i.e.*, annual or monthly), there are changes due to new IXP policies. On more medium time scales (*i.e.*, weekly), there is churn in IXP membership (*e.g.*, new members join, but there are also potential departures from the IXP associated with mergers and acquisitions), number of switch ports, and peerings (*e.g.*, new peerings are established, de-peerings, or peering changes such as switching from a public peering arrangement to a private peering). On small time scales (*e.g.*, daily or hourly and below), traffic variations are the main cause for changing IXP conditions.

To address this aspect, instead of analyzing the entire nine months of essentially uninterrupted sFlow measurements from our IXP, we selected four one week-long periods during late April, late August, mid-October, and late November/early December of

2011. We selected weekly periods based on the fact that the AS membership at our IXP was by and large stable during the course of a week. At the same time, choosing four one week-long periods from the nine months long sFlow measurements results in four snapshots that – as seen from Table 7.1 – capture some of the churn that our IXP faces on the medium to large time scales. In particular, we note a steady increase in the number of members of our IXP and in the traffic volume they generated during the nine months long measurement period. How these and other changes manifest themselves in the IXP's peering fabric and its use is the theme of the next sections.

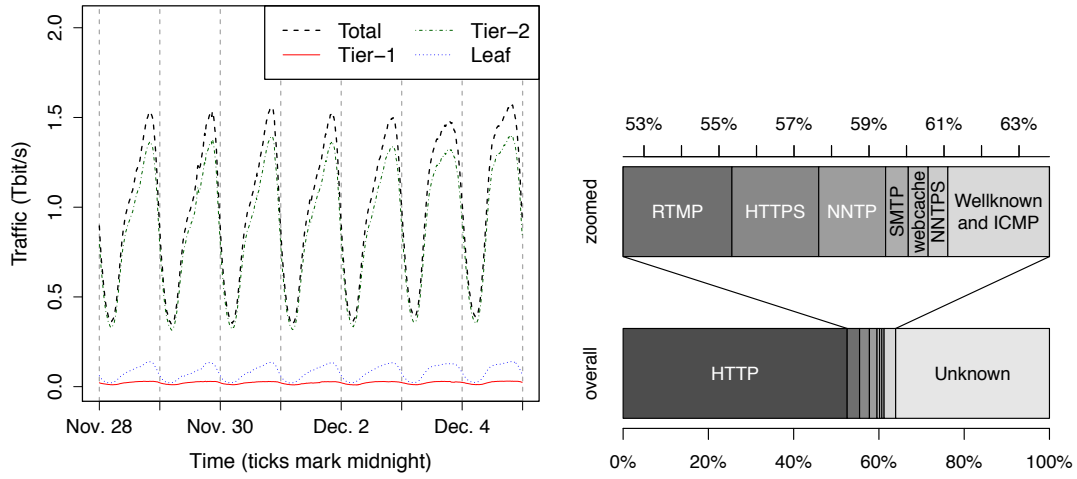
In the rest of this chapter, we use the Nov/Dec data to illustrate our main findings. Where appropriate, we also include the results for the data of the other three weeks. Overall, we find that their analysis is consistent with the results we report here for the Nov/Dec data. In addition, we also spot-checked our results against a number of additional one week-long data and found no inconsistencies.

### 7.2.4 Membership and traffic statistics

To identify the active member ASes at our IXP during a given time period, we had to determine between which member ASes an observed IP packet is being forwarded. To this end, we relied on layer-2 information (*i.e.*, MAC addresses) since the IP addresses in the header of the observed packets were those of the communication endpoints, not the routers on the path. We mapped MAC addresses to router IP addresses and their respective AS numbers by combining link-layer information from sampled ARP packets with routing data obtained from a publicly available looking glass at the IXP. This allowed us to identify 98 % of the members' routers. In the end, we succeeded in determining for more than 99 % of all observed sFlow packet samples their respective originating and receiving member ASes and as a result, we were able to identify for each week more than 350 AS members, each using between one and three logical router interfaces (see the first three rows in Table 7.1). The remaining less than 1 % of the exchanged traffic volume consists of IPv6 traffic and traffic that we could not associate with any member AS.

Being able to identify the IXP's members for a given time period, we examined next the member ASes of the IXP in each of the four weeks in more detail and report in rows 4 to 8 in Table 7.1 overall information about their tier level, country and continent. We considered the networks listed in Renesys "baker's dozen" [123] to be tier-1 ISPs and used the AS rank data provided by CAIDA [15] to classify the remaining members as tier-2 or leaf networks[3]. To this end, we classified a member AS as a tier-2 network iff it has both provider as well as customer ASes and as a leaf network iff it has only provider ASes. Based on this straightforward classification scheme that is largely agnostic to network specifics such as business, size, or traffic, we observed that irrespective of the considered time period, the vast majority of

[3]In this chapter, we decided against using the terms tier-3 and stub because of the different possible interpretations of their meaning.




(a) Traffic volume generated by different tiers.

(b) Application mix.

**Figure 7.2:** IXP traffic statistics for the Nov/Dec week.

members of the IXP are tier-2 networks. At the same time, all the tier-1 ISPs in Renesys "baker's dozen" list are members of our IXP. To determine the country and continent of each member AS, we relied on the country code field that can be found in the AS's whois data. While the IXP members are from more than 40 countries in three continents, most of the member ASes are part of the European Internet scene. To highlight the geographic concentration even more, a majority of those European member ASes offer services in the same country in which our IXP is situated.

Figure 7.2(a) shows that for the Nov/Dec week, the total daily traffic volume generated by the member ASes and exchanged over the IXP's public switching infrastructure was in the petabyte range and followed a pronounced time-of-day pattern that is well-synchronized with the daily business or user activities in the country where our IXP is situated. In agreement with the observed tier-membership of the IXP's member ASes, the tier-2 networks were responsible for most of the total traffic volume. Rows 9, 10, and 11 in Table 7.1 provide information about the total traffic volume for each of the four weeks considered, namely the average packets per second, average bits per second, and average daily volume.

Not surprisingly, when breaking down the total traffic volume that traverses the IXP by member AS, we observe a skewed distribution, irrespective of whether we consider sent or received bytes. Indeed, less than 3 % of the member ASes were responsible for about 30 % of the total traffic and less than 30 % of the member ASes were responsible for close to 90 % of the traffic in the Nov/Dec week. Especially noteworthy is that while all tier-1 ISPs are members at this IXP and, with one exception, do exchange traffic over the IXP's public peering infrastructure, contrary to other parts of the Internet, they contribute relatively little to the total traffic volume, presumably because their high volume traffic travels over private peering

links supported by the non-public part of this IXP's infrastructure. At the same time, the observed link load of public peerings involving tier-1 ISPs at this IXP was typically higher than average and causes some of those tier-1 ISPs to be included in the 30 % of members that generated close to 90 % of the total traffic volume.

Lastly, an added benefit of working with sFlow records is that it enabled us to examine the IXP traffic by applications. To this end, we separated the ICMP from the TCP and UDP packet samples by looking at their protocol field, and (when possible) associated the TCP and UDP packet samples with an application by looking at their source and destination port numbers and relying on the publicly available lists of port numbers used by the most popular applications. Figure 7.2(b) shows the application mix for the Nov/Dec week. While this straightforward approach cannot account for roughly 35 % of the bytes, we clearly see that HTTP is the most dominant application, accounting for more than 50 % of the bytes. This observation is consistent with recent reports from inter-AS traffic studies [84] and measurements of residential networks [93]. The next most popular applications are RTMP, HTTPS, and NNTP, the last of which has been reported to be used as a file sharing alternative [78].

## 7.3 An IXP-centric view of AS-level connectivity

The logical construct known as the AS-level Internet where nodes represent ASes and links denote AS relationships has no room for directly accounting for physical and geographically well-defined components of the Internet's infrastructure such as IXPs. As a result, an IXP's public peering fabric; *i.e.*, the set of (bi-directional) AS relationships of the peer-peer type (P-P links, discussion in Section 7.3.3) that exist among pairs of member ASes of the IXP and express routing policies in support of settlement-free traffic exchange among pairs of members over the IXP's public infrastructure is not directly discernible and has received little attention in the past [8]. In this section, we rely on sFlow records from our IXP to obtain the ground truth of this IXP's public peering fabric. We call the compact description that summarizes which member AS is publicly peering with which other member ASes at this IXP the IXP's peering matrix. We then contrast this actual peering matrix with its counterparts derived from analyzing various BGP and traceroute datasets that have been used in the past for inferring AS-level connectivity and generating inferred AS-level maps of the Internet.

### 7.3.1 Peering fabric seen from within the IXP

According to a commonly-used definition, two ASes are connected (at a particular time) in the logical AS graph if they can exchange routing information directly, *i.e.*, without the help of an intermediary AS that provides transit, presumably for the purpose of exchanging IP traffic. In the case of our IXP where we know its

topology (mapping of MAC and IP addresses to member ASes) and have access to its sFlow records, we use a more pragmatic definition and say that there exists a P-P link between a pair of member ASes if – during a given period of time – we see IP traffic being exchanged between these two member ASes over the IXP's public infrastructure. This pragmatic definition expresses our intention to focus on those P-P links of the IXP's peering fabric that matter; that is, carry actual IP traffic, *e.g.*, BGP packets only in the case of backup links or IP packets generated by genuine application-level traffic. We call the thus-defined peering matrix the "ground truth" for our IXP as it provides the most useful and complete information about the actual status of the peerings between its member ASes.

After filtering the Nov/Dec sFlow records as described in Section 7.2 and analyzing the resulting traffic, we found that out of a total of $396 \times 395 \,/\, 2 = 78{,}210$ (bi-directional) P-P links that the 396 IXP member ASes could potentially establish at the IXP in that time period, more than 50,000 P-P links were actually established and were used to exchange IP traffic. This corresponds to a "peering rate" at our IXP or a "fill degree" of this IXP's (symmetric) peering matrix of about 67 %, meaning that on average, each member AS exchanges IP traffic over the IXP's public infrastructure with some 270 other member ASes. In total, the observed ground truth of this IXP's peering fabric with its more than 50,000 active P-P links is responsible for about 10 PB of traffic that traverses this IXP's public infrastructure daily. Next, we examine how well this IXP's actual peering matrix can be replicated when instead of relying on IXP-provided sFlow records, we are limited by measurements that do not involve the IXP and are obtained from outside the IXP.

### 7.3.2 Peering fabric seen from outside the IXP

In the past, BGP routing information (*i.e.*, control plane data) as well as traceroute measurements (*i.e.*, data plane information) have been widely used to analyze the structure and evolution of the AS-level Internet. Access to our IXP's actual peering fabric gives us a unique opportunity to evaluate how the various inferred peering matrices for this IXP that result from relying on these different IXP-external datasets compare to the IXP's ground truth.

In terms of BGP routing information, we relied on two well-known sources, *i.e.*, Route-Views (RV) [129] and RIPE NCC (RIPE) [125], and on a non-public dataset (NP). For RV and RIPE, we relied on all their available route collectors, and used both BGP table dumps and updates from the same period when the Nov/Dec sFlow records were collected. NP consists of BGP dumps collected from about 70 routers worldwide which receive BGP information from 724 different ASes also covering the full week. Table 7.2 provides details about the total number of ASes from which the various datasets obtained BGP data and shows that despite varying significantly in magnitude, the three datasets are by and large complementary and contain routing information from almost 1,000 different ASes.

| Dataset | Unique LGs / ASN | Visible links | only in this dataset |
|---|---|---|---|
| RV | 78 | 5,336 | 1,084 |
| RIPE | 319 | 10,913 | 5,460 |
| NP | 723 | 3,419 | 684 |
| RV+RIPE+NP | 997 | 13,051 | 10,472 |
| LG | 821 / 148 | 4,892 | 2,313 |
| RV+RIPE+NP+LG | 1,070 | 15,364 | 15,364 |

**Table 7.2:** Overview of routing and looking glass datasets for November. The numbers show P-P links.

With respect to traceroute measurements, we used a dataset that resulted from a re-run of the targeted traceroute experiment described in [8]. This experiment was especially designed with the goal of discovering P-P links at IXPs and relied critically on the availability of publicly available traceroute-enabled looking glass (LG) servers throughout the Internet. The re-run was performed during Nov/Dec of 2011 using an updated list of available LG servers. The dataset we considered is derived from all traceroute probes launched as part of this recent campaign and consists of all inferred P-P links that involve our IXP and have an associated high confidence level of representing actual P-P links at our IXP (see [8] for details).

To systematically examine which P-P links at our IXP can and which cannot be discovered with the help of which IXP-external datasets, we classify these links into three categories. A **visible P-P link** is a P-P link that is observed both in the IXP-provided sFlow records and the IXP-external datasets (*e.g.*, BGP or traceroute data). A P-P link is called an **invisible P-P link** if it is visible from the IXP-provided traffic data (*i.e.*, IP packets traverse the link), but not visible from the IXP-external datasets. Lastly, a **cannot-tell P-P link** is a P-P link that is visible in BGP data but no traffic exchange is observed between the two member ASes in question from our IXP-provided data. This scenario is typical for private peering arrangements supported by the IXP's non-public infrastructure, but could also arise in those rare situations where a peering is not established at the IXP, or simply not visible in the traffic due to packet sampling. Note that the visible and invisible P-P links add up to the more than 50,000 P-P links that constitute the ground truth of our IXP's peering fabric. Furthermore, since the cannot-tell P-P links cannot be seen from the IXP-provided data, they are not a subset of either the visible or invisible peerings.

Using each of the IXP-external datasets, separately and in different combinations, Table 7.2 gives *(i)* the total number of visible P-P links that can be seen from the different IXP-external data and *(ii)* the number of unique visible P-P links; that is, those P-P links that can only be seen from exactly one of the IXP-external datasets. When compared to the ground truth, we see that each of the IXP-external datasets misses the vast majority of the observed links, and even when pooling all this available

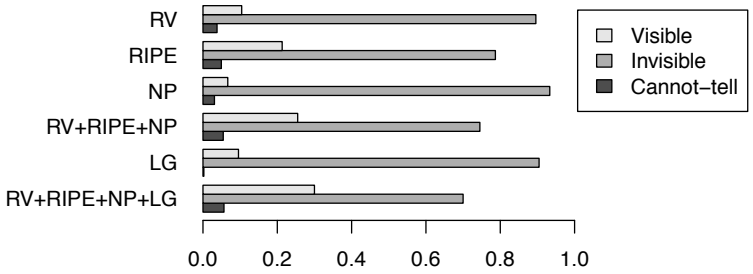



**Figure 7.3:** Peering links and visibility in control/data plane (normalized by number of detected P-P links).

control- and data plane information, we can still only account for a limited fraction of this IXP's actual peering fabric. A more detailed account of our findings is provided in Figure 7.3 and illustrates the breakdown of the P-P links into the three different categories of P-P links introduced above. We observe that even when relying on all the available datasets, about 70 % of the P-P links at this IXP remain invisible.

### 7.3.3 Types of AS relationships at the IXP

Irrespective of the environment of a peering link, which might be an IXP or any other network location, two parties of a bi-lateral peering agreement are free to (mis)use it as they see fit, *e.g.*, as a customer-provider (C-P) link. However, it is commonly assumed that most links at IXPs are of the peer-peer (P-P) type [151, 104]. This has been confirmed by the IXP operator and is supported by our findings in Sections 7.4.2 and 7.4.4.

Nevertheless, we now label the IXP peering links according to external information about their type of peering agreement and check the validity of the assumption that IXP links are mostly of P-P type. For that, we rely on the Cyclops [22] dataset, which provides a comprehensive and regularly updated Internet topology map with link types. From all IXP peering links that we observe, approximately 50,000 in December 2011, about 19 % are present in the Cyclops data, most of which are P-P links (18 %), and only very few are C-P (less than 1 %).

Although this is not to be taken as sufficient evidence about the actual link types at our IXP, it does add confidence to our P-P link assumption. These results are also consistent with our previous key observation that about 70 % of the IXP links are invisible from the outside.

### 7.3.4 Some food for thought

A survey of the recent literature on measuring the AS-level Internet shows that as of late 2009, the total number of P-P links in the entire Internet was estimated to

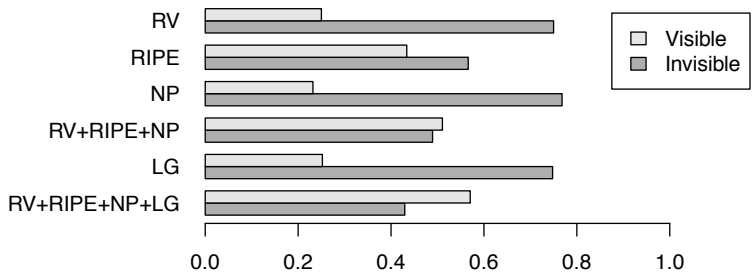



**Figure 7.4:** Peering traffic and visibility in control/data plane (normalized by total traffic volume).

be in the 35,000-45,000 range. The low end of this range results from adding to the 15,000-20,000 P-P links reported in [38] the roughly 20,000 new P-P links that were discovered in [8] and passed very strict validation criteria. The high end of this range is reported by Chen et al. [21] who used less stringent criteria for validating newly detected P-P links. In stark contrast to these recent estimates, our above analysis shows that the more than 50,000 P-P links that we encountered in this single large European IXP exceed the total number of P-P links assumed to exist Internet-wide. In view of arguments that suggest that many of these P-P links at IXPs are not critical in topology inference [160] or for understanding the evolution of the Internet, for example due to their possible role as backup links [38], we use again our IXP as an example. We show in Figure 7.4 the fraction of the total traffic traversing the IXP infrastructure that would not be accounted for if we only knew about the visible links; that is, the P-P links whose existence at this IXP can be inferred from the various BGP or traceroute data. Figure 7.4 shows that when using these IXP-external datasets individually to infer the visible links, each of them misses between 56–78 % of the total traffic (in bytes or packets) handled by this IXP. Even when pooling all the IXP-external datasets, close to half of the total traffic would be missed, due to the large number of P-P links that are not seen. Therefore, trying to gain insight into the economic incentives and business reasons of the various member ASes of this IXP for establishing the encountered peering fabric would be very hard knowing the visible peerings only.

Table 7.2 and Figure 7.3 show why efforts to unveil the peering fabric at this IXP (or others) by using publicly available or even privately collected BGP data or relying on measurements obtained from carefully designed traceroute experiments are essentially doomed. The main reason is the well-known problem of vantage points [128]. On the one hand, as shown in [107], the locations within the AS-level Internet of the monitors traditionally used to collect the widely-used BGP data provide a relatively accurate picture of the Internet AS-level connectivity as far as AS links of the customer-provider type are concerned, reportedly missing less than 11,000 out of a total of about 94,000 of such links Internet-wide [21]. On the other hand, these monitors have hardly any visibility into the Internet's IXP substrate consisting of the various

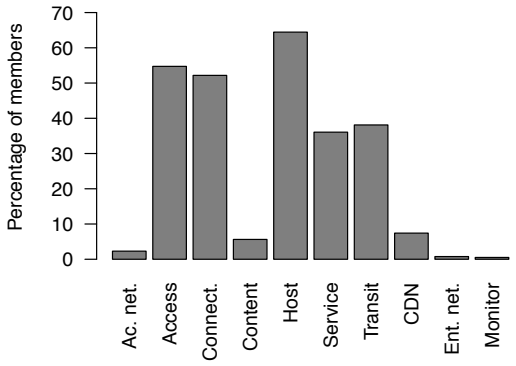



**Figure 7.5:** Diversity in members: AS classification by business types.

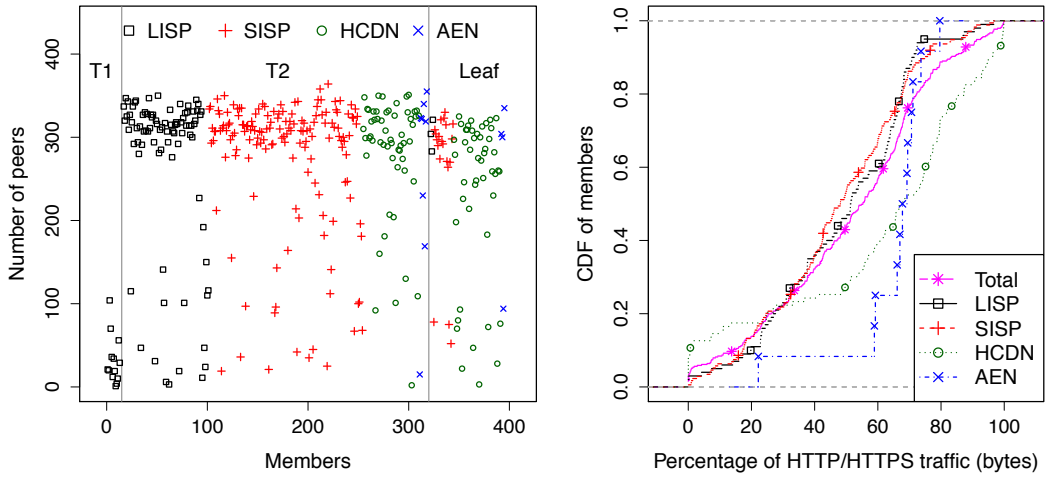
IXPs, their member ASes and the P-P links among them at those IXPs [106] and thus miss the majority of those links. At the same time, even when trying to use traceroute measurements and launch probes from LG servers close to an IXP to a target in an AS "on the other side" of the IXP, due to AS-specific routing policies, there is no guarantee that the probes traverse the IXP and improve the discovery of P-P links at the IXP.

## 7.4 Diversity of the IXP ecosystem

In this section, we take a closer look at our IXP's ecosystem; that is, its member ASes, the rich peering fabric we described in Section 7.3, and various aspects of the traffic that is exchanged among the IXP's member ASes over this peering fabric.

### 7.4.1 Member ASes

We have already noted that the traditional classification of networks into tiers says little about the nature and nothing about the business types of the networks that are members at this IXP (see Section 7.2). Unfortunately, there is no readily available dataset which lets us determine the business type(s) of each member AS. Therefore, we manually examined the information available on each of the member ASes' web sites and present in Figure 7.5 their business type(s). Clearly, the business model of the member ASes differ significantly, and it is not uncommon to encounter member ASes that are in multiple business types. Focusing on their main business type, we further classified each of the member ASes as a *large ISP* (LISP), *small ISP* (SISP), *hosting/service and content distribution network* (HCDN), and an *academic and enterprise network* (AEN). A large ISP is providing transit, connectivity, eyeball access and additional services such as hosting or even content distribution. A small ISP is an access provider and may also provide transit services. Hosting and service




(a) Scatter-plot of num. of peers per member.

(b) Fractions of web-traffic across members.

**Figure 7.6:** Diversity in members: number of peerings and application mix exemplified by web-traffic.

providers are hosting content, either indirectly through providing web-space or rack-space to actual creators of content, or as content owners. Some of them also provide special services such as DNS. The AEN category comprises all networks that are solely used to connect enterprises and universities.

### 7.4.2 Peering

For each of our IXP's member ASes, Figure 7.6(a) shows the number of its P-P links; that is, the number of other members with which it peers at this IXP. The member ASes are ordered (x-axis) according to our classification as introduced in Section 7.2 (*i.e.*, tier-1, tier-2, and leaf networks), with no particular order within the resulting groups. Figure 7.6(a) reveals an enormous diversity with respect to the number of peers – some member ASes peer only with a few other members, while others peer with almost all of them. In particular, we see that the tier-1 ISPs have a small number of P-P links, typically peering with less than 25 % of all member ASes. This observation is consistent with their stated intention of offering only restrictive peering (*e.g.*, on peeringDB [115], on the IXPs web site, or on the companies' web sites). Tier-1 ISPs apparently use the IXP, among other reasons, to augment their existing peerings, but need to do this with care because most other member ASes are either transit customers or potential transit customer for them.

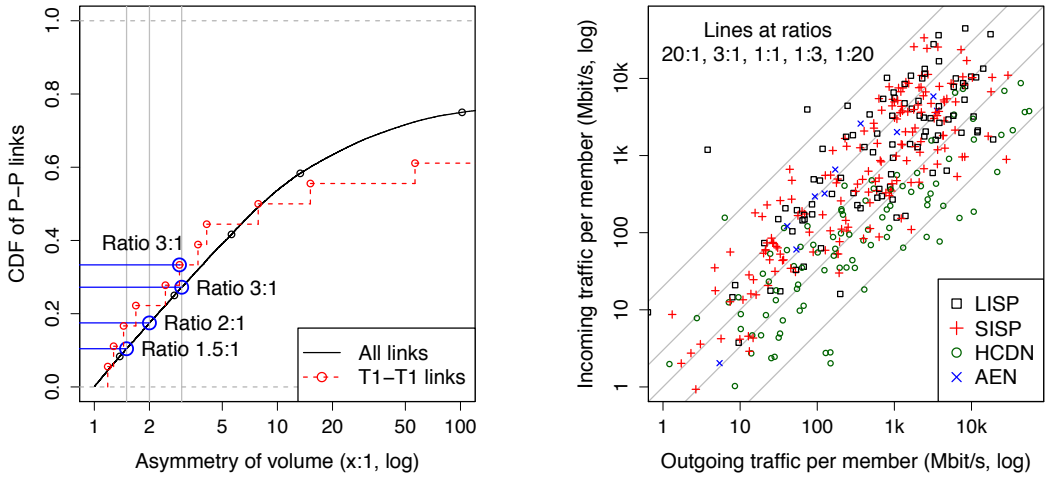
Non-tier-1 members typically peer with a large number of other members, with about 71 % of the tier-2 and leaf member ASes peering with at least 70 % of the other members. Among the tier-2 and leaf networks (more than 96 % of the members),

there are less than 10 % which peer with less than 25 % of all members. Tier-2 and leaf member ASes usually have an open peering policy, meaning that when asked what they look for in a peering, their answers are mainly performance and reducing transit costs. Some prefer selective over open peering policies, especially if setting some standards for a potential peer's network in terms of criteria such as traffic exchange ratio, geographic scope, backbone capacity, or traffic volume is in their interest. We find that most member ASes at this IXP use open peering policies and peer massively with other members. However, we also encounter ASes with open peering policies that do not peer with that many other members. Possible reasons for their low peering rate are *when* they joined the IXP, *if* they offer desirable content or *if* they provide Internet access to a significant number of eyeballs.

Figure 7.6(a) also shows a classification of the member ASes in the four business categories defined above: LISP, SISP, HCDN, and AEN. Based on this classification, we find that in the LISP group, the member ASes with a small number of peerings are the tier-1 ISPs and those ISPs with a selective peering policy. In the HCDN group, the networks with a few peerings include some of the large players, but also small hosting providers (*e.g.*, for banks or online games). The picture is less clear for the SISP group. In general, the observed large number of member ASes that have a large number of peers at this IXP is testimony for the ease with which member ASes can peer at this (and other) IXP. In fact, the findings of a recent survey [151] provide compelling reasons – some 99 % of the surveyed peerings were a result of "handshake" agreements (with symmetric terms) rather than formal contracts, and an apparent prevalence of multi-lateral peering agreements; that is, the exchange of customer routes within groups of more than two parties.

### 7.4.3 Traffic

The contributions to the IXP's overall traffic by the individual member ASes is highly skewed, with the top 30 % of member ASes contributing close to 90 % of the overall IXP traffic. Examining in more detail the traffic volume that each member AS contributes to the IXP's overall traffic, we first investigate what role the traffic exchange ratio plays in establishing P-P links. To this end, we consider the traffic asymmetry across all peerings between any two member ASes and show in Figure 7.7(a) the empirical cumulative probability distribution of this asymmetry. For improved readability we only show the part of the curve for ratios up to 100:1 (75 % of all peerings). The figure reveals a high variability in terms of exchanged traffic between the two member ASes of a peering. Indeed, only 27 % of the links have a traffic ratio of up to 3:1 (see support lines), where a 3:1 ratio is often stated as a typical requirement in common formal peering agreements [103]. Moreover, for 8 % of the peerings the ratio exceeds 100:1, and for another 17 % we observe traffic in only one direction. Figure 7.7(a) also depicts the empirical cumulative probability distribution for the




(a) Traffic asymmetry across P-P links.

(b) Traffic asymmetry (out/in) per member.

**Figure 7.7:** Diversity in traffic asymmetry.

P-P links at this IXP involving only tier-1 ISPs and shows that these peerings are less asymmetric, with more than 33 % of them having a ratio below 3:1.

Figure 7.7(b) shows the traffic asymmetry of the member ASes (*i.e.*, the ratio of outgoing bytes vs. incoming bytes of a given member AS). The traffic of 52 % of the member ASes is more or less symmetric and within the range of 1:3 to 3:1. However, a significant number of member ASes fall in the 3:20 to 20:3 range[4]. In agreement with expectations, HCDNs have more outgoing than incoming traffic, while the opposite is true for LISPs and SISPs. However, there are various exceptions to this rule, and we find HCDNs with significantly more incoming than outgoing traffic and LISPs and SISPs where the opposite holds true. Note that despite the significant diversity in the ratio of incoming and outgoing traffic, more than half of the member ASes that send most of the traffic also receive most of the traffic. Indeed, there is a 50 % overlap among the top 50 member ASes according to bytes sent and the top 50 member ASes according to bytes received.

We can also examine how similar or dissimilar the overall application mix (see Section 7.2) is across all the IXP member ASes. For example, when computing for each member AS the fraction of HTTP/HTTPS traffic relative to the total number of bytes sent and received, we find in Figure 7.6(b) that this application mix differs significantly across the member ASes and follows almost a uniform distribution, indicating that without additional information, it would be difficult to predict which percentage of a member AS's traffic is HTTP. However, as soon as we include for

[4]To illustrate, if we had a member AS that would only deliver content using 1,500 byte-sized packets, the ratio could be as bad as 1:58, assuming on average one ACK of 52 bytes for every two data packets of 1,500 bytes and no overhead for the TCP connection establishment.

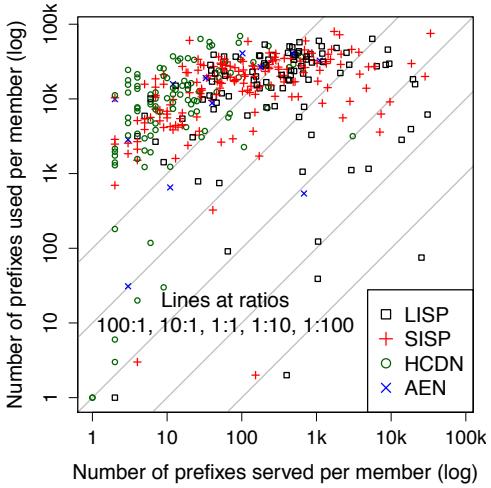



**Figure 7.8:** Diversity in use of prefixes: prefix ratio per member AS.

example information about the member AS's business type, we observe that as expected, hosting providers and CDNs tend to send a larger fraction of HTTP traffic. However, rather unexpectedly, we also see more than 10 % of the hosting providers and CDNs with only marginal fractions of HTTP traffic. Closer inspection shows that these member ASes are primarily service providers that do not provide web content.

### 7.4.4 Prefixes

We next consider the prefix exchange ratio. For this purpose, we say that a prefix is *served* by a member AS if the member AS receives traffic for that prefix. Vice verse, we say that a prefix is *used* by a member AS if output traffic of its access router is destined toward that prefix. Figure 7.8 depicts a scatter-plot of the ratio of the number of prefixes used vs. the number of prefixes served by each member AS and provides clear evidence that the vast majority of the member ASes of our IXP use more than 10-times the number of prefixes they serve. Specifically, we see that hosting providers and CDNs have a tendency to serve a smaller number of prefixes but to use some two orders of magnitude more prefixes. Focusing on the ISPs, we can identify two groups. The first, larger group, serves a diverse but limited set of prefixes, from a few tens to a few thousands. The second, smaller group, serves and uses a large number of prefixes, some tens of thousands. Members that serve such large numbers of prefixes are likely acting as transit networks for other member ASes. However, we again observe exceptions to these general observations in almost all categories.

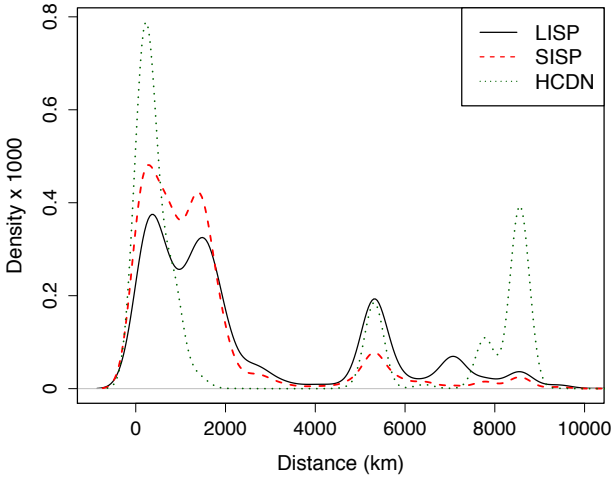



**Figure 7.9:** Geographic distances of IP endpoints to IXP.

### 7.4.5 Geographical aspects

Conventional wisdom about IXPs states that ASes join regional IXPs to exchange local traffic. To examine this general belief, we used the MaxMind GeoLite City database [94] to identify the geographic coordinates of both source and destination IP addresses for each sampled packet. Despite known inaccuracies of this geolocation database, using it for the needs of this study is appropriate, as we are only interested in approximate distances at the country level [119]. Contrary to our expectations, we found that only 10 % of the traffic is exchanged within the country in which the IXP is situated, while another 26 % originates from that country, and another 3 % is destined for that country. However, when relaxing the geographic constraint and considering a geographic region within a radius of 2,000 km from our IXP, we confirm the local nature of IXP traffic – almost 80 % and 72 % of the traffic terminates and originates, respectively, within this relatively close proximity.

To better understand the geographic reach of our IXP, Figure 7.9 depicts the density of the distances of traffic originated by member ASes in the LISP, SISP, and HCDN groups, weighted by byte volume. The distances shown in this figure are measured from the IP source address to the IXP, *i.e.*, they represent the geographic range from which the IXP attracts traffic. We find that HCDNs have the largest fractions of very short distance traffic and at the same time the largest fraction of very long distance traffic – 37 % of traffic volume with a distance larger than 5,000 km suggests the presence of significant intercontinental traffic. This indicates either mismatches in the IP address location [119] and/or that some of the traffic is indeed being served from remote locations. Likewise, member ASes in the LISP group show strong presence at around 5,000 km, which is mainly contributed by a small number of large international ISPs. SISPs and AENs (not shown) typically send traffic from closer to the IXP than members in the other business groups.

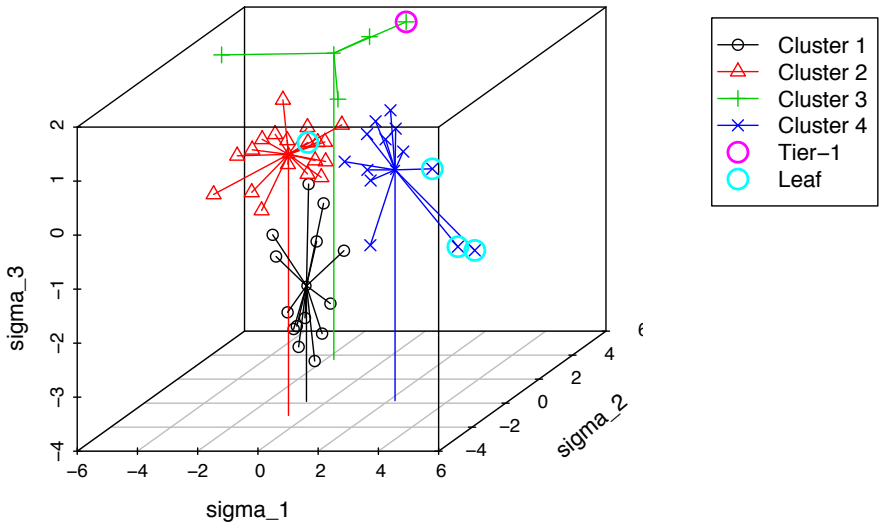



**Figure 7.10:** SVD-based 3D projection based on 12 features of the top 50 member ASes by bytes sent.

### 7.4.6 Tiers without tears

The above analysis highlights the diversity of the member ASes in terms of business types, the number of peerings, as well as their traffic characteristics. We have already seen indications that the traditional classification of networks by tiers cannot account for this observed diversity, mainly because it is agnostic to features of the member ASes such as their business type, traffic, peerings, prefixes, and geographic properties. Clearly, these and other features have the potential of painting a much more interesting and relevant picture of networks compared to what is possible knowing only the presence or absence of provider and customer networks.

In the rest of this section, we explore the possibility of combining some of these features and identify meaningful clusters. To this end, we consider 12 features in an attempt to characterize the member ASes' peerings and traffic characteristics: number of bytes sent, number of bytes received, number of peers, number of ASes and prefixes they send traffic to and receive traffic from, percentage of HTTP traffic that they send and receive, and 25-percentile of the distances from the traffic source to the destination, as well as from the IXP location to the destination for outbound traffic, and from the traffic source to the IXP location for inbound traffic.

We consider the Singular Value Decomposition [80] (SVD) of the $396 \times 12$ matrix and look at its projection into the 3D space defined by the three eigenvectors corresponding to the three largest singular values. Intuitively, the SVD produces a set of combined features (*i.e.*, linear combinations of the original variables) so that the variability of the values of the first few combined features is maximized. Figure 7.10

shows the resulting 3D figure as a scatter-plot. To keep the number of points reasonable, we sub-selected the top 50 member ASes according to sent bytes. Similar plots result if we sub-select according to number of bytes received or if we increase the number of member ASes to the top-100. In generating Figure 7.10, we repeatedly clustered the selected member ASes using the k-means clustering algorithm with random starting points into four clusters. Increasing or decreasing the number of clusters considered (*i.e.*, value k) had no major impact on the nature of the results.

The output of this combined SVD/clustering method typically consists of one small cluster (˜5 member ASes), two medium clusters (˜10 member ASes) and one large cluster (˜20 member ASes). When examining the clusters in more detail, we find that (i) the small cluster contains only large content and service providers, (ii) one of the medium cluster contains mainly small to medium ISPs that provide access to residential and enterprise customers and are all located in a country different from the IXP, (iii) the second medium cluster has mainly large ISPs and backbone networks that provide transit, and (iv) the big cluster has mainly data centers, big hosting providers, CDNs and some ISPs that provide web and server hosting. To highlight one such example clustering, we use in Figure 7.10 different symbols and colors and connect the clusters with a spider and mark their centers by support lines. We view Figure 7.10 as evidence that it is possible to classify an IXP's member ASes in ways that are practical and meaningful and respect the real-world diversity among networks that are critical elements of an IXP's ecosystem. Importantly, annotating the individual points in Figure 7.10 with tier-information shows why conventional tier classification is uninformative and of little help when trying to understand the Internet ecosystem locally (*i.e.*, at this IXP) or globally.

## 7.5 IXP traffic matrix

Many IXPs report up-to-date traffic statistics on their web sites, and some of the largest European IXPs show daily traffic volumes for their public switching infrastructure that have been consistently in the petabyte range for some time. In Section 7.2, we confirm this publicly available but little-known fact for our IXP. A breakdown of this overall traffic by member ASes can be compactly described by an IXP's traffic matrix that specifies for example the hourly or daily traffic volumes exchanged between all member ASes that have a P-P link at this IXP. Despite the many similarities with an ISP's intra-AS traffic matrix, we are not aware of any published research paper that has considered IXP-specific traffic matrices and their properties. This is largely a reflection of the attention that large ISPs (and big content) have received from researchers and an indication that IXPs have been viewed as relatively uninteresting in terms of their topology, traffic, and routing. As is the case with ISP-provided measurements for computing or inferring an ISP's intra-domain traffic matrix, access to IXP-provided data for studying IXP-specific traffic matrices is

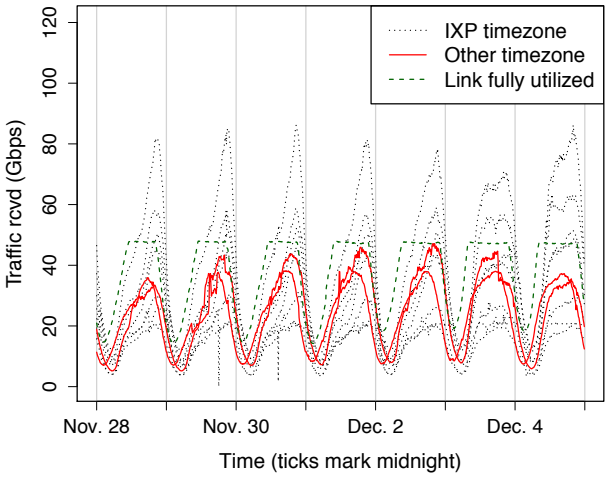



**Figure 7.11:** Daily pattern of top-10 tier-2 members.

similarly critical, and we rely in the following on our IXP-provided sFlow measurements. We report below on a first-of-its-kind analysis of the actual traffic matrix of one of the largest IXPs worldwide, examine in detail properties such as the diurnal pattern, sparsity, and (approximately) low rank, and discuss possible applications of our findings in support of managing and operating a large IXP's infrastructure.

### 7.5.1 Temporal properties

We have seen in Section 7.2 (see Table 7.1) that during our nine months-long measurement period, the overall traffic seen by our IXP steadily increased, mainly due to a similarly steady increase in the number of its member ASes. In addition, Figure 7.2(a) shows that the total traffic volume over time is dominated by a pronounced time-of-day characteristic, where the observed diurnal cycle coincides with the daily business cycle in the country where this IXP is located. Such diurnal behavior has long been a trademark of the temporal nature of measured intra-domain ISP traffic matrices.

To explain this diurnal behavior for our IXP, note that *(i)* the tier-2 member ASes are responsible for a majority of the overall traffic (see Figure 7.2(a)) *(ii)* the top-10 of these tier-2 member ASes generate much of the IXP's traffic (more than 33 %), and *(iii)* despite the tier-2 member ASes being a very heterogeneous group of networks, a majority of them cover the city, region, or country where our IXP is situated. Therefore, when plotting the incoming traffic volume for the top-10 receivers among the tier-2 member ASes in Figure 7.11, we see that the temporal behavior of their traffic is well-correlated with the overall traffic and well-aligned with the daily business cycle of the area where our IXP is located. An exception to this rule is the traffic of the two member ASes shown with solid lines in Figure 7.11 that are shifted by some

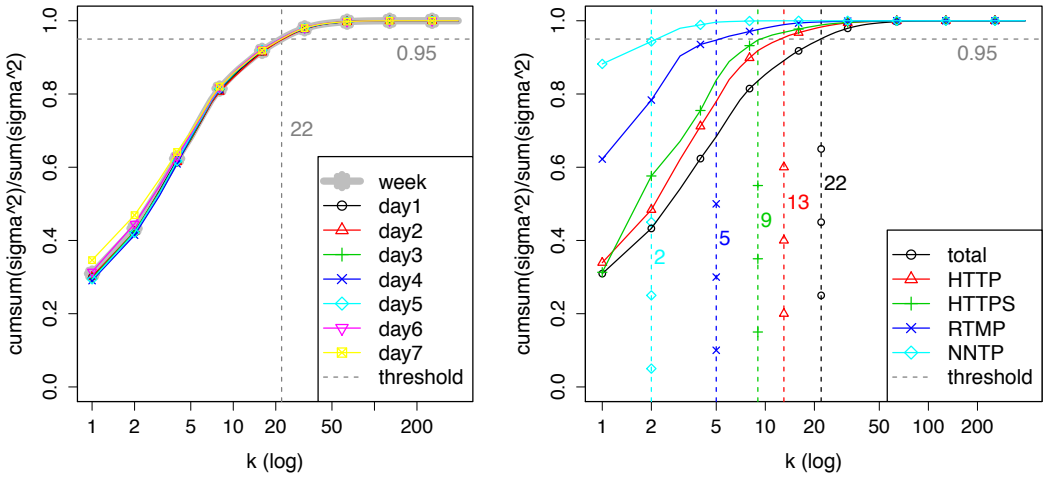
1–5 hours as a result of representing the traffic of ISPs serving geographic areas in different time zones, one in Europe and one outside of Europe. The traffic of the member AS plotted with dashed lines in Figure 7.11 visually reaches the bandwidth limits of its IXP network interfaces. We were able to confirm this observation by closer inspecting the sFlow records, which revealed that this particular member AS connects with five physical 10 Gbps links to the switching fabric of the IXP, thus it is indeed reaching its 50 Gbps capacity limit.

Knowing such temporal properties of an IXP's traffic matrix takes on a new meaning in an environment where innovation in the IXP marketplace in the form of new types of service offerings or refined routing policies is likely to increase the volatility and decrease the predictability of the traffic seen by the IXP. For example, in the presence of IXP port re-sellers, the IXP has limited visibility into the networks that use the re-seller as an intermediary and only indirect control over the traffic that these networks send or receive at the IXP. Similarly, members that can select prefix-specific peering to send certain traffic at certain times through the IXP instead of handing it off to their upstream provider(s) are likely to be a significant source for increased traffic volatility at IXPs. These are instances where a detailed understanding of the temporal dynamics of an IXP's traffic matrix will be critical for accurate prediction and successful root cause analysis of observed infrastructure performance problems.

### 7.5.2 Structural properties

A readily observable structural property of measured ISP-specific traffic matrices is their sparsity. The fact that generally only a small number of entries in these traffic matrices are populated or non-zero is largely due to a carefully planned network and complex routing decisions. In stark contrast, as reported in Section 7.3, with a "fill degree" of more than 65 %, our IXP-specific traffic matrix cannot be called sparse, and the main reason for this observed non-sparsity is economics. More precisely, an IXP's infrastructure and routing requirements are purposefully kept simple to facilitate easy network interconnection among member ASes, resulting in a rich peering fabric in support of traffic exchanges at the IXP between as many member ASes as dictated by "good" business practices.

Next, we examine how our IXP-specific traffic matrix compares to measured ISP-specific traffic matrices with respect to their widely-reported property of having approximately low rank. Recall that in SVD terminology, a matrix $X$ of (algebraic) rank $k$ has approximately or nearly low rank if only a small number $k^* \ll k$ of the largest singular values are needed to well-approximate $X$ by a $k^* \times k^*$ matrix under a certain norm of the approximation error (see for example [158] for details). In practice, to check if a matrix has approximately low rank, we consider its singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_n$, compute the energy function defined by $f(k) = (\sum_{i=1}^{k} \sigma_i^2)/(\sum_{i=1}^{n} \sigma_i^2)$, and find the smallest $k^*$ that captures, say, 95 % of the total energy.



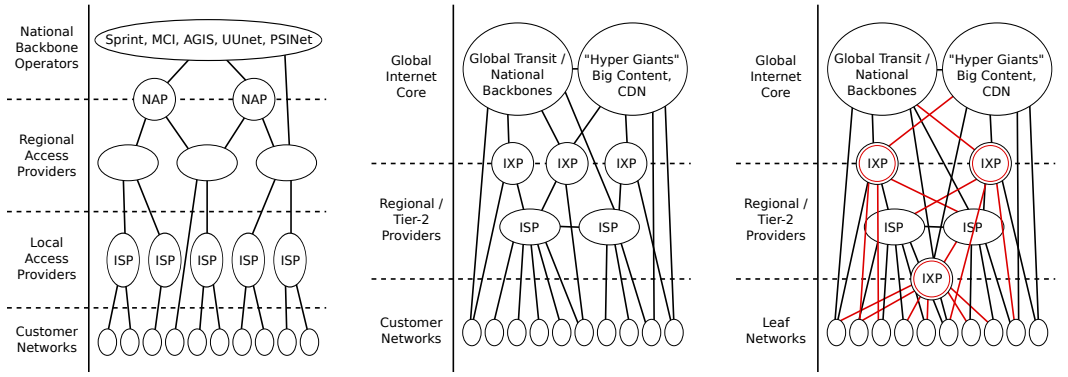

(a) Traffic matrix energy.

(b) Traffic matrix energy by application.

**Figure 7.12:** Traffic matrix properties.

Plotting the energy $f(k)$ as a function of $k$, Figure 7.12(a) shows the results of applying this method to a week-long traffic matrix (trace from Nov/Dec, thick line) and the corresponding seven daily traffic matrices (thin lines). In addition to confirming the approximately low rank of these traffic matrices (*i.e.*, out of some 380 non-zero singular values, only the 22 largest are needed to capture 95 % of the energy), the plot also illustrates consistency among the week-long trace and the different daily traces. Similarly, recalling the application mix of the total traffic (see Figure 7.2(b)), Figure 7.12(b) shows the approximately low rank nature of the week-long application-specific traffic matrices. We observe that the low rank nature becomes more pronounced as we consider, in order, HTTP, HTTPS, RTMP, and NNTP, even though all these application-specific traffic matrices have almost full rank. In fact, in the case of NNTP, a simple $2 \times 2$ matrix can successfully capture most of the NNTP portion of the traffic in the entire original IXP-specific traffic matrix of size $396 \times 396$. Even for the case of RTMP, only a handful of singular values are needed to well approximate the portion of the overall traffic generated by this application. Another more specialized set of traffic matrices that also have approximately low rank can be constructed by considering only that portion of the overall IXP traffic that is produced by the top-50 member ASes (in terms of sent bytes) that were used as input to the clustering study described in Section 7.4 (see Figure 7.10) and ended up in one and the same cluster. A reason to examine such specialized traffic matrices is to look for any connection between the different business types that roughly specify these clusters and the low rank nature of the corresponding traffic matrices, and preliminary results (not shown here) suggest that the answer is affirmative.

While there are many other types of IXP-specific traffic matrices that can be considered, having approximately low rank is a common property among them and hints




(a) Traditional Internet structure [65].

(b) Internet structure as of 2010 [84].

(c) Internet structure as of 2012 (this thesis).

**Figure 7.13:** Evolution of the Internet AS-level structure.

at the presence of enormous amounts of structure that is hidden in real-world IXP-specific traffic matrices and begs the question how this structure could possibly be exploited for practical purposes. One promising such application concerns efficient data acquisition. Specifically, as an IXP's infrastructure grows in terms of member ASes, peerings, and traffic, existing monitoring infrastructures that rely on increasingly lower sampling rate to keep up with the encountered growth in traffic may no longer be viable, especially in view of more stringent performance criteria that are promised by the IXP and typically require higher-resolution IXP measurements. The approximately low rank nature of IXP traffic matrices suggests a viable alternative whereby fewer but more intelligently collected (and hence more expensive) measurements can provide as much information as the current brute-force method of throwing more hardware at the problem to support the collection of maximal amounts of highly redundant data.

## 7.6 Discussion

It is generally known and understood that BGP-based efforts for discovering P-P links in the AS-level Internet have somewhat limited success and provide at best a lower bound for the number of such links in the Internet [38]. However, that the number of P-P links at a single IXP exceeds even very recently reported such lower bounds [38, 21, 8] has come as a big surprise. This and the realization that even when laboriously combining the most up-to-date publicly available BGP data with hard-to-get non-public control plane measurements and the latest available state-of-the-art data plane measurements, our pooled data can only account for some 30 % of all known P-P links at our IXP begs the question how we can miss so many actual P-P links and why. Clearly, obscuring the existence of a "live" P-P link can occur with

control plane data (*e.g.*, the route server from which BGP data is pulled not being close enough to the IXP) and with data plane measurements (*e.g.*, due to routing policies that prevent direct traffic exchange via an existing P-P link at the IXP and force traffic between two member ASes to take the upstream path). We plan to study the role of routing policies in hiding existing P-P links at IXPs as part of our future work and expect that an in-depth understanding of the root causes will suggest novel measurement experiments that have the potential of providing an accurate and near-complete peering matrix for each IXP and, in turn, an approximately valid snapshot of the AS-level Internet.

Irrespective of the reasons for the enormous number of encountered P-P links at our IXP, we have seen that the observed rich peering fabric by and large defies the well-known tiered structure of the Internet. As such, it is much more a reflection of the varied economic incentives and business benefits that drive the different member ASes to massively peer at this IXP and an indication of the ease with which such P-P links can be established. The resulting rich peering fabric supports massive direct interconnections or "shortcuts" as viable alternatives to sending traffic upstream and by and large agrees with recently reported findings of a "flattening" of the Internet [84] (see Figure 7.13(b) and Section 2.4.2). In fact, while some of our results are largely complementary to those of reported in [84], they do provide a different perspective. Relying exclusively on IXP data (as compared to the use of non-IXP-only data), we use an analysis of the IXP members' business types and a port-based application classification of their traffic to observe a similar consolidation of content and applications. However, our explanation for this consolidation centers more around the discovered massive peerings among all kinds of networks at this IXP and relies less on the presence and formation of "hyper-giants".

While this "flattening" of the Internet can fully co-exist with the Internet's traditionally assumed hierarchical structure (see Figure 7.13(a) and Section 2.4.2), our IXP-specific findings and the following cautious extrapolations to the Internet as a whole suggest an even more radical change in perspective of the AS-level Internet (Figure 7.13(c)). Indeed, considering only the European IXP scene (see [47] for details) and being conservative in using our large IXP as a baseline (*i.e.*, assuming only a 50 % peering rate at IXPs), counting up the P-P links we expect to encounter at the four largest IXPs (with, say, 400 unique members each) and at the 10 next-largest IXPs (with, say, 100 unique members each), we obtain a realistic lower bound for the estimated number of P-P links for just the European portion of the Internet of some 200,000. This number is more than 100 % larger than the number of *all* AS links (*i.e.*, customer-provider and peer-peer) in the entire Internet in 2010 as reported in [38] or the number of *all AS links of the customer-provider type* Internet-wide in 2008 as reported in [21][5].

Despite being extremely conservative, this estimate by itself should give reasons to

[5] Note that the reported numbers involving customer-provider links are from the published literature and cannot be derived from our IXP data.

pause. First, it indicates that there are easily an order of magnitude more P-P links in today's Internet than previously assumed. Second, contrary to conventional wisdom, there are many more P-P links in today's Internet than customer-provider type peerings, with twice as many being a conservative estimate. Third, judging from what we have seen at our IXP, most of these massive amounts of P-P links are of critical importance as they carry significant traffic. Given that past studies of the Internet peering ecosystem assumed exactly the opposite of what our findings support, there is a need for a major overhaul of the mental picture that our community has about the AS-level Internet, not only in terms of local and overall structure, but also with respect to its evolution in response to often rapidly changing business conditions locally at an IXP or within the larger Internet. In particular, we argue that assessing the "standing" of a network within the Internet's ecosystem has to account for network-specific features (*e.g.*, business type) and some notion of traffic that this network is responsible for, either as a source, sink, or transit entity. This renewed focus on traffic requires novel approaches for the measurement, modeling, analysis, and inference of the Internet's inter-AS traffic matrix. Despite some initial efforts dealing with this matrix and a large body of existing work on intra-AS traffic matrices, the Internet inter-AS traffic matrix has remained a big enigma, but the availability of IXP-specific traffic matrices promises to invigorate research activities in this area.

## 7.7 Related work

Over the past years, the AS-level Internet has been a much-studied graph structure and continues to fascinate networking and non-networking researchers alike, though typically for different reasons. Instead of attempting to provide a necessarily incomplete overview of the existing literature on this topic, we refer the reader to a number of recent studies that serve as useful surveys [57, 36, 38, 128]. A majority of published research in this area has focused on measuring, inferring, modeling, and characterizing the AS-level Internet, often for the purpose of applying inferred AS graphs or carefully-tuned models to specific problems; *e.g.*, see [40, 98, 130, 37, 54].

Given that as logical constructs, AS topologies have no explicit space for physical infrastructure components, IXPs have long been neglected and generally viewed as an unimportant "detail". This view has slowly started to change with the gradual realization of the existence of "hidden" links in the Internet AS topology [18, 153, 29, 62, 106, 63]. [8] is the first traceroute-based study that purposefully targeted the existing IXPs worldwide. In conjunction with improvements on the measurement side, there has also been an increasing awareness of the important role that IXPs play in the Internet ecosystem, *e.g.*, see [16, 139, 145, 17]. Ironically, among network operators, this realization has been there pretty much from the beginning of the commercial Internet [71, 101, 102], and there are signs that the research community is starting to give the Internet's IXP substrate the attention it deserves.

As far as traffic matrix research is concerned, despite some recent efforts [19, 128, 12, 84], little (if anything) is known about the Internet's inter-domain or inter-AS traffic matrix. The pieces of this puzzle that have received most attention by researchers have been the tier-1 ISPs and their intra-domain traffic matrices (*e.g.*, see [20, 160, 1] and references therein). In fact, the latter have been a key ingredient for many ISP-critical tasks such as traffic engineering, capacity planning, and anomaly detection [50, 127, 86]. In stark contrast, when it comes to IXP-specific traffic matrices, only very recent work [16] has considered peering and traffic trends through a longitudinal study of a small European IXP across a period of 14 years. However, no study has gone into the same level of details as for intra-domain ISP traffic matrices. As is the case with ISP-provided measurements for computing or inferring an ISP's intra-domain traffic matrix, access to IXP-provided data for studying IXP-specific traffic matrices is similarly critical and requires close collaborations with IXPs. For the work reported in this chapter, we have been fortunate to be able to work closely with one of the largest IXPs worldwide.

## 7.8 Summary

Examining readily available public information about a number of large European IXPs shows that they are very similar, not only with respect to the make-up of their constituents (*i.e.*, member ASes) and overall traffic volumes, but also in terms of offered services, underlying technologies, business models, and overall purpose. As such, access to detailed internal measurements from even just one such IXP can highlight the important role that these largely ignored entities play for the Internet as a whole and for the particular geographic regions where they are located.

To this end, we analyze in this chapter a unique dataset of nine months' worth of continuous sFlow measurements from one of the largest IXPs in Europe, and worldwide, and clarify in the process some common misconceptions that exist regarding IXPs and the AS-level Internet. These include, among others, that tier-1 ISPs do not peer at IXPs (they do), IXPs are not used for transit (they are), the number of peer-peer links in the Internet is small (it is at least an order of magnitude larger than what has been assumed), the number of customer-provider links in the Internet is much larger than the number of peer-peer links (there are easily twice as many peer-peer links than customer-provider links), and IXP peerings are mostly used for back-up (they are not). In particular, we examine in detail the peering fabric and traffic matrix of our IXP and show the existence of a very diverse ecosystem in terms of the member ASes' business types, peering strategies, traffic exchanges, and geographic coverage that mimics the Internet's AS ecosystem as a whole. We argue that these findings are a proof that the mental picture our community has about IXPs and the AS-level Internet is in much need for a major overhaul.

# 8

# Investigating the Impact of the World IPv6 Day

After our extensive examination of the IPv4 traffic as observed at one of the largest IXPs worldwide in Chapter 7, we now move our focus to the up-and-coming incarnation of the Internet Protocol for which Internet routers are urged to be well prepared: Internet Protocol version 6 (IPv6).

While the IETF standardized IPv6 more than fifteen years ago, IPv4 is still the prevalent Internet protocol today. On June 8th, 2011, several large content and service providers coordinated a large-scale IPv6 test-run, by enabling support for IPv6 simultaneously: the World IPv6 Day. In this chapter we compare IPv6 activity before, during, and after the event. We examine traffic traces recorded at a large European IXP (*cf.* Chapter 7) and on the campus of a major US university; analyzing volume, application mix, and the use of tunneling protocols for transporting IPv6 packets.

For the exchange point we find that native IPv6 traffic almost doubled during the World IPv6 Day while changes in tunneled traffic were limited. At the university, IPv6 traffic increased from 3–6 GB/day to over 130 GB/day during the World IPv6 Day, accompanied by a significant shift in the application and HTTP destination mix. Our results also show that a significant number of participants at the World IPv6 Day kept their IPv6 support online even after the test period ended, suggesting that they did not encounter any significant problems.

## 8.1 Introduction

The fourth incarnation of the Internet Protocol (IPv4) successfully supported the phenomenal growth of the Internet since its introduction in 1981. Yet, due to this unexpected success, the pressure from the IPv4 address space exhaustion is being felt more and more. This led to the standardization of IPv6 more than 15 years ago, which provides a significantly larger address space. Since then, the transition from IPv4 to IPv6 is happening at a lethargic pace. One of the reasons for the hesitant adoption of IPv6 by end-users is the limited amount of content available through IPv6. A reason for network operators is the fear of breaking critical services. Indeed, the current best practices for deploying IPv6, such as white-listing of well-known network regions, are very conservative. Furthermore, such approaches prevent us from gaining insights into the challenges involved with a global transition to IPv6.

To fill the gap, several operators coordinated a joint experiment on June 8th, 2011: the World IPv6 Day. For the duration of that day, the participants agreed to enable IPv6 support in parts of their networks. Participants included Comcast, Google, Facebook, Microsoft, and many others. Their observations have been reported at the IETF 81 meeting. They found that besides a significant and sustained increase of IPv6 traffic on and after the World IPv6 Day, the awareness of IPv6 increased dramatically, and the experience obtained through real IPv6 deployments and measurements were invaluable. The presented results were focused mainly on operational questions, *e.g.*, bandwidth, number of clients, and "IPv6 brokenness".

In this chapter, we complement these observations by investigating IPv6 traffic characteristics from two vantage points in the Internet. We examine the use of tunneled IPv6, the presence of applications in IPv6 traffic, and highlight the major IPv6 traffic contributors in the Internet. Our study is based on two traces of production Internet traffic. The first was collected at a large European Internet Exchange Point (IXP) interconnecting hundreds of networks. The second has been gathered at a major US university, a fundamentally different vantage point compared to the IXP, both in scale and level of traffic aggregation. Combined, the two datasets enable us to take a broad look at the impact of the World IPv6 Day.

To the best of our knowledge, this chapter is the first systematic study of what has happened around the World IPv6 Day. Our contributions include characterizations of:

**Traffic volume:** In both traces, we observe a steep and sustained increase of IPv6 traffic. Native IPv6 traffic doubled at the IXP and increased more than 20-fold at the campus.

**Tunneling mechanisms:** Encapsulated packets contribute a large fraction of IPv6 traffic at the IXP. Teredo tunnels are widespread but mostly idle.

| Name | Type | Location | Start date | Duration |
|---|---|---|---|---|
| JUN1 | Packet | Campus | Thu, Jun 2 | 9 d |
| JUN2 | Packet | Campus | Fri, Jun 17 | 4 d |
| JUN3 | Packet | Campus | Fri, Jun 24 | 7 d |
| IXP1 | sFlow | IXP | Wed, Jun 1 | 22 d |
| IXP2 | sFlow | IXP | Mon, Aug 8 | 7 d |

**Table 8.1:** Overview of the datasets. All datasets are from 2011.

**Application mix:** Since the World IPv6 Day, the application mix of native and 6in4 IPv6 traffic changed fundamentally and now exhibits similarities to IPv4.

**Traffic contributors:** Since the World IPv6 Day, YouTube is the main contributor at the campus vantage point. A large content provider is the main contributor at the IXP.

The remainder of this chapter is organized as follows. In Section 8.2, we provide details about our two datasets. In Section 8.3 we first investigate overall IPv6 traffic volume and tunnel encapsulations (Section 8.3.1), the application mix (Section 8.3.2), and we identify the content providers that contribute most traffic (Section 8.3.3) before, during, and after the World IPv6 Day. We present related work in Section 8.4 and summarize our results in Section 8.5.

## 8.2 Datasets

We base our analysis on network traffic gathered at the Internet uplink of a major US university and at a large European Internet Exchange Point (IXP). Table 8.1 gives an overview of our datasets.

In addition to analyzing native IPv6 traffic, we also investigate commonly used tunnel encapsulation methods to transfer IPv6 datagrams over IPv4. In particular, we analyze *Teredo* [67], *6in4* [100], and *AYIYA*[1] encapsulations. We note that 6in4 encapsulation also covers *6to4* [66] and *6rd* [143]. Some tunneled traffic can be detected by filtering on a specific UDP port; Teredo uses UDP port 3544, and AYIYA commonly runs on port 5072. In contrast, 6in4 has its own IP protocol number, 41, which can be used for filtering. In all of our analyses, we further verified that the tunnel payload actually contains an IPv6 packet to mitigate against false positives.

[1] `http://www.sixxs.net/tools/ayiya/`

#### Internet Exchange Point

The IXP datasets consist of anonymized sampled sFlow records from the whole traffic exchanged at the IXP. More than 400 networks currently exchange traffic at this IXP. sFlow does not employ flow record aggregation like NetFlow. Instead, sFlow samples one out of n packets and exports the initial portion of it as a sFlow record. The sFlow probes at the IXP use a sampling ratio of $1{:}2^{14}$. We use a customized version of `sflowtool` [131] to extract relevant portions from the sFlow data. As a sFlow record corresponds to the initial portion of a packet, it is possible to examine the protocol and tunneling stack.

#### US university

We base our analysis of IPv6 traffic at the US university campus on packet level traces collected at the university's central uplink to the Internet. We limited the trace collection to native IPv6 traffic, 6in4 encapsulated traffic and IPv4 traffic on Teredo's well-known UDP port. We then analyze these traces using a customized version of the Bro IDS [114] capable of analyzing tunneled IPv6 traffic.

## 8.3 IPv6 traffic at the IXP and the university network

In this section we analyze different aspects of the IPv6 traffic in both of our traces; the IXP and the campus network one. We first investigate the overall IPv6 traffic volume and study the presence of IPv6 over IPv4 transition technologies. We then analyze the application mix in IPv6 traffic and investigate the main traffic contributors.

### 8.3.1 Traffic volume and tunneling

We start by investigating the overall volume of IPv6 traffic before, during, and post the World IPv6 Day. This enables us to calibrate our expectations for subsequent analyses when we dig deeper into used protocols, applications, and traffic contributors.

In Figure 8.1 and Figure 8.2 we plot the total bandwidth of IPv6 traffic (native and tunneled) over time at the IXP and the US university, respectively. The World IPv6 Day is highlighted by a gray bar. We observe that before the official start of the World IPv6 Day (at midnight UTC), IPv6 traffic begins to ramp up as content providers enable IPv6 on their systems. During the World IPv6 Day, we observe a 30 % increase of IPv6 traffic at the IXP and an increase from 3–6 GB/day to over 130 GB at the university. We also find that the IPv6 traffic volume remains high after the World IPv6 Day officially ended, indicating that a significant number of participants kept their IPv6 support enabled, and suggesting that they did not encounter significant

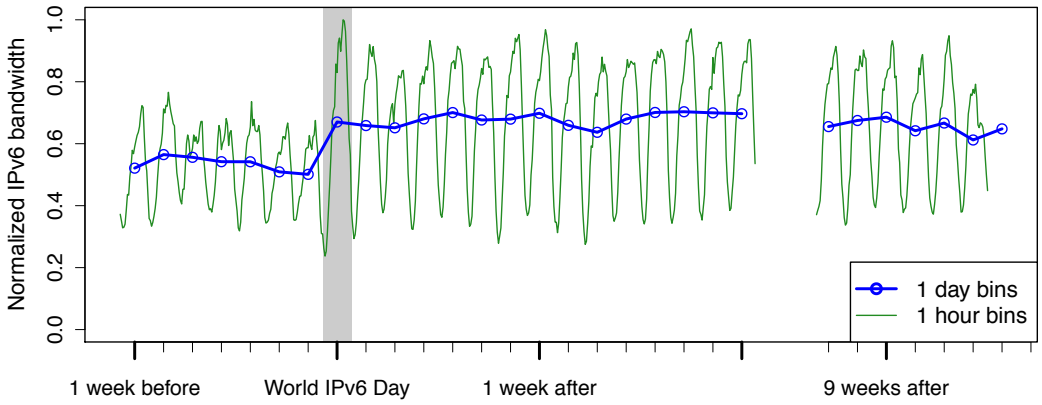



**Figure 8.1:** Total IPv6 traffic volume over time (IXP). The tick marks are at noon UTC.

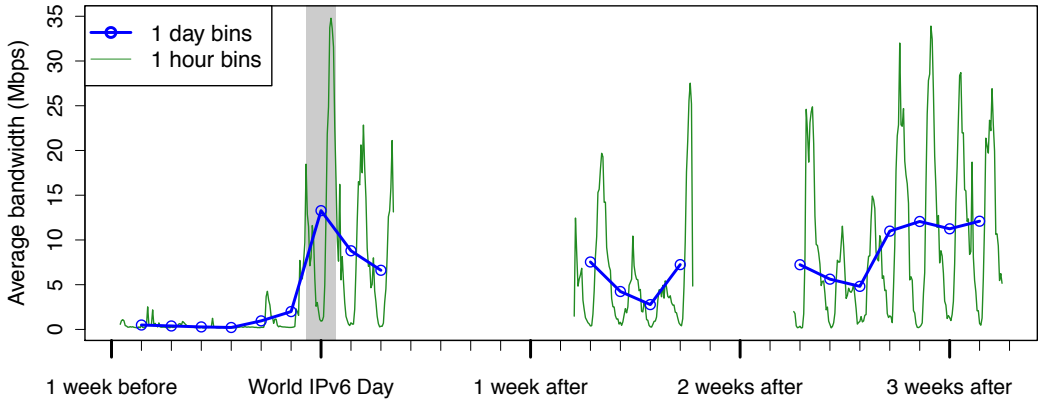



**Figure 8.2:** Total IPv6 traffic volume over time (campus). The tick marks are at noon UTC.

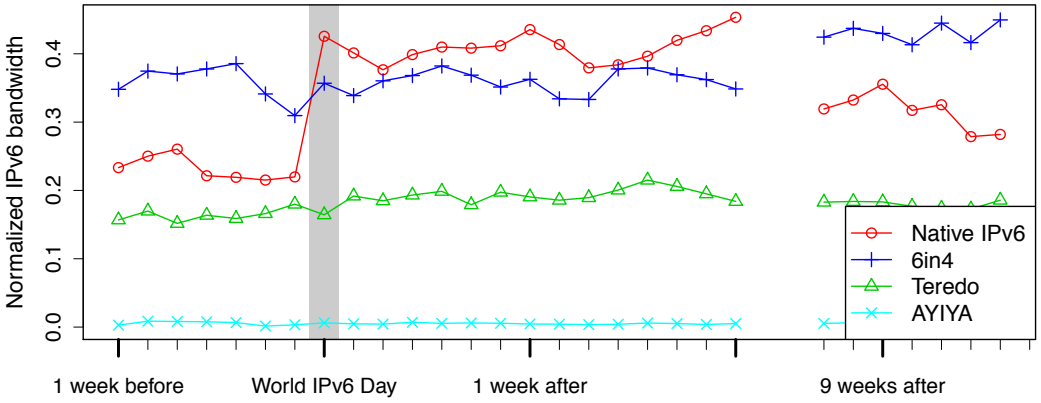



**Figure 8.3:** IPv6 traffic volume by tunnel encapsulation (IXP).

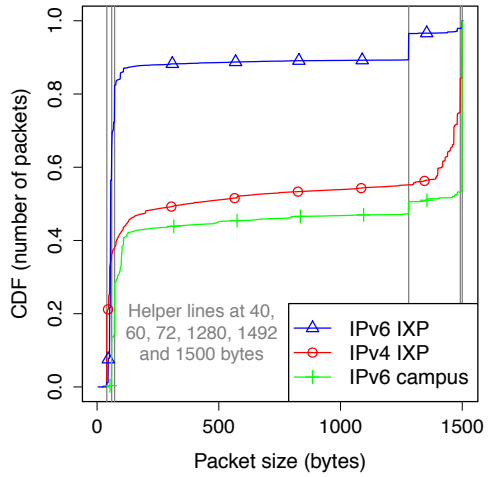
problems. This is consistent with other reports [14, 87, 150] that observed similar behavior during and after the World IPv6 Day.

Analyzing IPv6 traffic in 1 hour bins shows a clear time-of-day pattern (plot not shown). During and after the World IPv6 Day, the traffic volume during the busy-hour has increased significantly while the traffic dips during off-hours has remained unchanged, indicating that only peak usage has changed but not baseline activity.

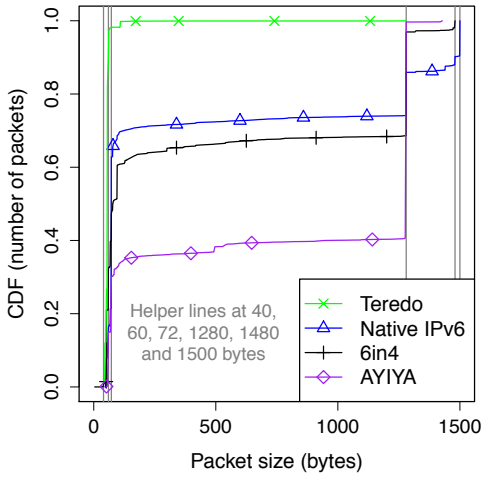
We next turn to the question of how much IPv6 traffic is tunneled versus native IPv6 traffic. At the university campus we find hardly any tunneled traffic. At the IXP tunneled traffic is more common. In Figure 8.3, we plot the IPv6 volume by tunnel encapsulation type for the IXP datasets. During and after the World IPv6 Day, we observe a significant increase in native IPv6 traffic, while tunneled traffic remains essentially unchanged. The fraction of tunneled traffic decreases accordingly from 69 % to 58 % on average.

We next compare the packet size distributions of IPv4 and IPv6 traffic during the World IPv6 Day and plot the results in Figure 8.4(a). We remove the tunnel headers and plot the size of the innermost IPv4 or IPv6 packet. IPv4 shows the usual distribution with peaks at small packet sizes (32 %) and large packet sizes ≥1,492 bytes (25 %). The packet size distribution for IPv6 at the US university resembles the one of IPv4. However, since an IPv6 header is larger than an IPv4 header without options, we find that the "small" packets for IPv6 are slightly larger. We also observe an additional mode at 1,280 bytes for IPv6. This represents the minimum MTU for IPv6 [34]), and the recommended MTU for tunneling mechanisms in order to mitigate problems with fragmentation [67, 100].

We observe a different packet size distribution for IPv6 at the IXP that shows a significantly larger fraction of small packets. More than 82 % of all IPv6 packets are at most 72 bytes in size. Moreover, we notice two modes in the distribution of larger




(a) Packet size distributions of IPv4 and IPv6 traffic (IXP and campus).




(b) Packet size distributions per encapsulation type (IXP).

**Figure 8.4:** IPv6 packet size distributions.

packets, one at the full MTU, and another one at 1,280 bytes. The latter is more pronounced than in the campus dataset.

To understand what causes this disparity, we take a closer look at the IPv6 packet size distribution at the IXP by breaking it down according to the type of packet encapsulation. Figure 8.4(b) compares the IPv6 packet size distributions for native, 6in4, Teredo, and AYIYA packets. We find strong differences between different encapsulation techniques. Native IPv6 traffic is the only significant source of full-sized 1,500 byte packets, since tunneled traffic needs room for additional encapsulation headers. In contrast to the native IPv6 traffic in the campus dataset, we still observe larger fractions of small packets and a stronger mode at 1,280 bytes. While the packet size distributions for native, 6in4, and AYIYA traffic show some similarities to IPv4, we find that 98 % of Teredo packets are small. A closer examination reveals that at our vantage point, Teredo is mostly composed of control traffic: 76 % of all observed Teredo packets are keep-alive messages (IPv6 headers without payload), and 23 % are ICMP messages. Since Teredo contributes 62 % of IPv6 packets during the IPv6 day, we conclude that Teredo skews the overall packet size distribution dramatically.

### 8.3.2 Application mix

We now turn to the application layer protocol mix of IPv6 traffic. We utilize Bro's dynamic protocol detection framework [42] to classify application layer protocols in the university datasets. As the IXP dataset only provided sampled packet headers, we rely on well-known port numbers to identify applications. We use a selection of

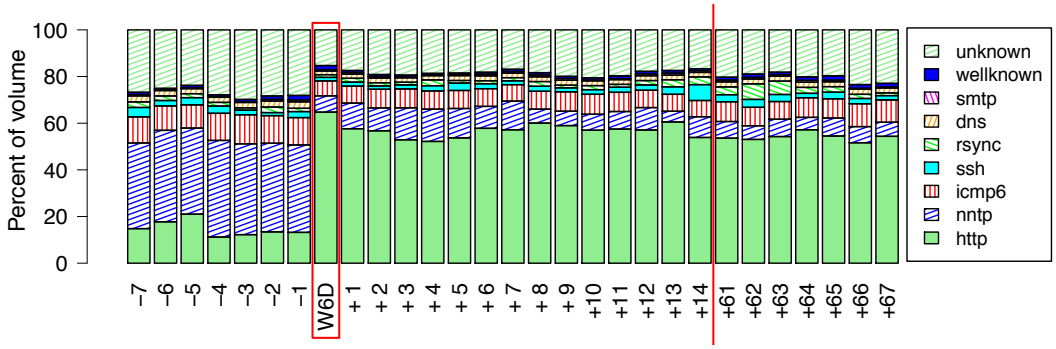



**Figure 8.5:** Application mix per day for *native* IPv6 traffic (IXP).

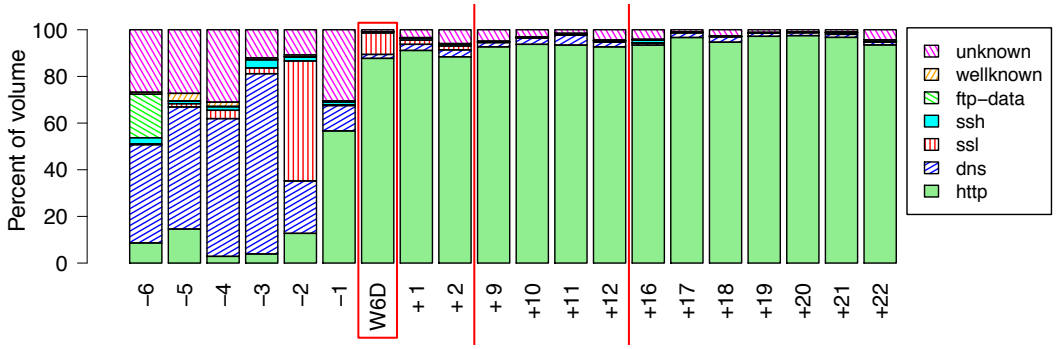



**Figure 8.6:** Application mix per day for *all* IPv6 traffic (campus).

86 well-known ports which have been shown to work reasonably well [93]. We report the top protocols and aggregate other traffic on well-known ports into the category *well-known*. If the port numbers do not allow to infer the application layer protocol, we attribute the traffic to the *unknown* category.

Figure 8.5 shows the daily application mix for native IPv6 traffic at the IXP for IXP1 and IXP2. The World IPv6 Day is highlighted by a red rectangle and IXP1 and IXP2 are separated by a red vertical line. Prior to the World IPv6 Day, NNTP was the strongest contributor with about 40 % of the volume, a protocol now frequently used for file-sharing [78]. While we cannot reliably identify P2P traffic in the IXP dataset, its share must be less than 30 % (sum of "well-known" and "unknown" categories). In contrast, Labovitz [83] reports P2P as the main contributor in IPv6 traffic before the World IPv6 Day, with 61 % of the total volume. ICMPv6 contributes 10 % to 13 % of the overall traffic volume. During the World IPv6 Day, the application mix has changed substantially. HTTP is dominating with more than 60 % of the traffic volume, NNTP dropped to 7 % and ICMP to 6 %. In addition, "unknown", and "well-known" now account for less than 15 %. After the World IPv6 Day, the application mix stays roughly similar to the one during the World IPv6 Day, with HTTP loosing about 7 to 10 % of its popularity and ICMPv6 slowly rising up to 9 %.

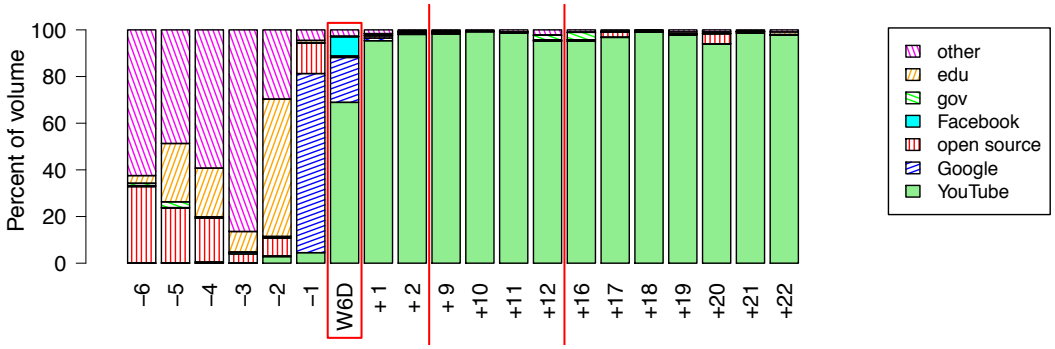



**Figure 8.7:** Daily HTTP mix (campus).

In Figure 8.6, we plot the application mix for the campus datasets. We again highlight the World IPv6 Day with a red rectangle and separate different traces with a vertical line. Similar to the IXP we notice a strong shift in the application mix during and after the World IPv6 Day. Before the World IPv6 Day, DNS traffic is in general the main contributor. During the ramp-up to the World IPv6 Day, at and post the World IPv6 Day, we see that HTTP is dominating with a share of up to 97 %. The DNS traffic volume remains unchanged (1–2 GB/day), indicating that it is caused by server-to-server DNS communication and not client requests.

**Inside 6in4 tunnels.** Since we separately observe multiple different IPv6 tunneling mechanisms at the IXP, we next analyze a breakdown of the application mix according to the tunneling protocol. However, we discuss only 6in4 tunnels since Teredo is almost entirely control traffic and AYIYA lacks volume to provide meaningful results. In 6in4 traffic, which is responsible for more than 32 % of the volume, the most prevalent packets are IPv6 fragments. Further examination of these fragments reveals that half of them have a size of 1,280 bytes (at offset 0), while the other half has 96 bytes. Almost all of the fragments use UDP as transport protocol. We investigated the fragments with offset 0 to get the UDP port numbers, which appear to be random. Assuming these fragments belong together, the size of the original IPv6 packet before fragmentation would have been 1,320 bytes, which is the minimum IPv6 MTU of 1,280 plus the size of an IPv6 header. We speculate that a broken client software tried to send packets with minimum MTU to prevent fragmentation but forgot to account for the IPv6 header. Before the World IPv6 Day, HTTP was typically at 1–5 % of the traffic volume. During and after the World IPv6 Day, the HTTP fraction increases to 10–16 %. Unknown traffic is at 45 % before and at 52 % during and after the World IPv6 Day.

### 8.3.3 Traffic sources

Since HTTP dominates in the campus environment (up to 97 % of total volume), we analyze HTTP in more detail. We utilize Bro's HTTP analyzer and extract

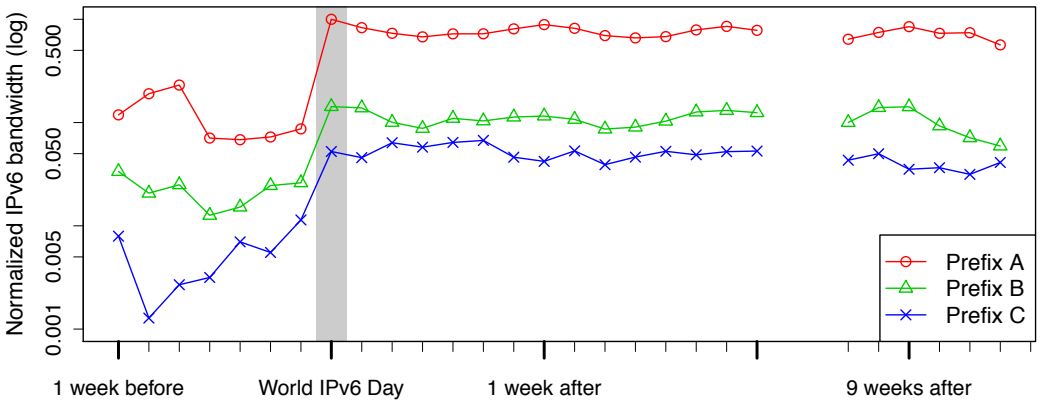



**Figure 8.8:** Normalized bandwidth of *all* IPv6 traffic per prefix before, at, and post the World IPv6 Day (IXP). The plot shows three examples out of the top ten high-volume IPv6 prefixes. Note the log-scale y-axis.

the HTTP server name from the Host header field. We use this information to group HTTP requests by their destination (*e.g.*, YouTube) and plot the result in Figure 8.7. The "open-source" category consists of HTTP-enabled open-source software sites, including `freebsd.org`, `mozilla.com`, and `ubuntu.com`. The "gov" and "edu" categories contain all sites under their respective top level domains.

We find that the mix of popular HTTP sites varies from day to day before the World IPv6 Day. Open-source and edu sites have significant shares and a large fraction of the traffic is generated by "other" sites. During and after the World IPv6 Day, we observe a significant change with YouTube and Google being responsible for most IPv6 HTTP traffic. According to our data, Google enabled IPv6 just before the official start of the World IPv6 Day and disabled IPv6 again after the World IPv6 Day. In contrast, we observe that YouTube kept IPv6 enabled after the World IPv6 Day. Considering that HTTP dominates the application mix and YouTube dominates the HTTP mix after the World IPv6 Day, we conclude that a large volume of IPv6 traffic after the World IPv6 Day is contributed by YouTube.

At the IXP we see more than 3,500 unique IPv6 prefixes. We investigate the largest prefixes in terms of IPv6 traffic volume. Figure 8.8 shows three out of the top 10 prefixes from the World IPv6 Day. With the help of the IXP we were able to identify prefix A as belonging to a large content provider, and prefixes B and C as large IPv6 enabled stub networks. Only prefix A is actively participating on the World IPv6 Day. Yet, all of them see a roughly ten-fold sustained increase in traffic volume since the IPv6 day. This highlights that passively participating networks can exhibit as much of a change as actively participating ones.

## 8.4 Related work

To the best of our knowledge this chapter is the first to perform a systematic study of the IPv6 traffic around the World IPv6 Day. However, there are a number of reports about IPv6 and the World IPv6 Day in the proceedings of the IETF 81 meeting in Canada, July 2011, contributed by the Operations and Management working group.

Palmer and Thaler from Microsoft provide an experience report [112] about the IPv6 activation of several Microsoft's domains. They report having only few connectivity issues. Windows Vista and Windows 7 dominate the observed system types. 91 % of the connections were native IPv6, and less than 1 % were using Teredo. This is consistent with our results about the idleness of Teredo tunnels, and also surprising since Microsoft has enabled Teredo tunneling as a default service since Windows Vista.

Bob Hinden from Check Point reports in [64] about their experience of enabling IPv6 for their company website by using load balancers to handle IPv6. They encountered less difficulties than expected and kept IPv6 active after the World IPv6 Day.

Comcast provides a summary of their IPv6 experiences in [14]. Comcast deployed SMTP over IPv6 by duplicating their infrastructure. Consistent with our results, they report a significant sustained increase of IPv6 traffic at the World IPv6 Day.

In contrast to this study, the above reports were limited to either a few web sites of a single operator, or in case of Comcast to a set of test customers. Still, the reported IPv6 traffic trends and conclusions are consistent with our results.

Hurricane Electric is an early IPv6 adopter – they enabled IPv6 in 2001. Similar to other reports, they observed [87] an IPv6 traffic increase during and after the World IPv6 Day. They also report on path MTU problems and ICMPv6 blocking caused by too aggressive filtering. In addition, they find that 11 % of ASes are present in the IPv6 routing table in August 2011, up from 3.6 % three years earlier.

Wijnen et al. [150] present results from active measurements including DNS, ping6, traceroute6, and HTTP probes. The data was gathered from 40 different vantage points from June 1st through June 11th, 2011. For example, they performed DNS AAAA queries to participating websites and found that nearly all World IPv6 Day participant web sites could be resolved successfully from all of their vantage points. Interestingly, the results also indicate effects due to negative caching of DNS records, as a number of vantage points were not able to resolve AAAA records of some participant, while other vantage points were. Furthermore, they show that after the World IPv6 Day, a number of web sites disabled IPv6 connectivity immediately, while DNS servers continued to return AAAA records for as long as half a day.

Claffy [26] provides an extensive survey of available data that enables tracking of IPv6 deployments, performs comparisons with IPv6 at the topology and the DNS level,

and calls out to researchers and industry to provide more data. With this chapter, we can contribute to some of the areas identified by Claffy, in particular utilization at access and interconnection links, application mix, and IPv6 tunneling.

Labovitz [83] performs a pre World IPv6 Day study of IPv6 traffic across several providers and presents an application mix including tunneled traffic in which P2P traffic dominates. In addition, this chapter characterizes how different tunneling protocols are being used.

Cho et al. [23] performed a very early study of IPv6 path problems and latencies compared to IPv4. Limoncelli et al. [89] compare rollout strategies for IPv6. Guérin et al. [56] model incentives in IPv6 deployment.

## 8.5 Summary

In this chapter, we conduct the first systematic analysis of IPv6 traffic around the World IPv6 Day. We rely on data collected at two vantage points: a large European Internet Exchange Point and the campus of a major US university. We analyze the traffic volume, application mix, and the use of tunneling protocols for transporting IPv6 packets.

We find that native IPv6 traffic almost doubled during the World IPv6 Day, while changes in tunneled traffic were limited. Teredo tunnels contribute a significant fraction to IPv6 traffic, yet only carry control traffic. We observe significant changes in the application mix during the World IPv6 Day, with the IPv6 application mix becoming similar to the IPv4 one. We find a large amount of fragmented IPv6 packets inside 6in4 tunnels for which broken software is a likely cause. Our results also show that a significant number of participants at the World IPv6 Day kept their IPv6 support online even after the test period ended, suggesting that they did not encounter any significant problems.

# 9

# Conclusion and Outlook

Routing and traffic forwarding in a global network of networks as gigantic and diverse as the Internet is a complex challenge. As of 2012, Internet routers need to deal with dynamic reachability information of more than 430,000 sub-networks, putting pressure on a router's control plane (*e.g.*, BGP) and data plane (*e.g.*, FIB). This thesis proposes a novel router design, called SDN Router, which is scalable in terms of its FIB memory by relying on a traffic offloading principle which leverages well-known and re-occurring properties in Internet traffic. Evaluation results based on real Internet traffic traces show that this router design and the corresponding algorithms meet the requirements of an ISP aggregation network while reducing FIB memory requirements by about two orders of magnitude.

Given that the design of the SDN Router relies on a profound understanding of the Internet traffic properties, this thesis reports on two extensive passive measurement studies based on IXP traffic traces. IXPs are largely understudied, but – as the results show – extremely central interconnection points in the Internet. This thesis investigates the IXP's ecosystem and challenges the common perception of the Internet AS-level structure. Another study focuses on IPv6 traffic and provides promising, forward-looking results on the increasing adoption of IPv6 in the Internet.

To wrap up, we summarize the main contributions and take-aways of each chapter individually and finish with a discussion of future research directions.

## 9.1 Summary

The key contributions of each chapter are summarized below.

**SDN Router:** An SDN-based scalable router architecture built around commodity components and open-source software.

Chapter 3 proposes the SDN Router concept, a split-architecture router design which is scalable in terms of its FIB memory. A SDN Router prototype implementation demonstrates the feasibility of the design. This prototype is based entirely on commodity and open components, such as an open-source software router, a commodity PC, and an OpenFlow-enabled switch. To transparently unite the software router with the OpenFlow-enabled switch according to the SDN Router concept, this chapter presents RouteVisor, an OpenFlow controller. Interoperability tests confirm that this prototype co-operates successfully with commercial routers.

**Traffic-aware Flow Offloading (TFO):** A traffic offloading algorithm that leverages Internet traffic properties.

Chapter 4 makes the key observation, that Internet traffic is consistent with Zipf's law: A small fraction of flows capture most of the traffic – a property which can be leveraged by traffic offloading algorithms such as TFO. In the SDN Router, TFO selects which prefixes to offload to the switch, based on recent traffic statistics. TFO addresses two main challenges: *(i)* maintaining a high traffic offloading ratio to limit the volume of the non-offloaded traffic; and *(ii)* keeping the PC-switch communication overhead low by limiting the frequency at which the offloaded prefixes are modified. The evaluation results based on three real Internet traffic traces from diverse network locations provide a promising picture: TFO is an effective strategy that achieves the before mentioned goals and stimulates applications of traffic offloading for alternative and more scalable router designs.

**Locality-aware FIB Aggregation (LFA):** An online FIB aggregation algorithm that leverages the spatial and temporal locality in BGP routing updates.

Chapter 5 focuses on another way of scaling FIB memory; with FIB aggregation. The number of entries in a FIB can be reduced, or aggregated, to about half of the original number, while maintaining forwarding equivalence. Online FIB aggregation however is challenging, as the frequent BGP routing updates need to be incorporated into the aggregated FIB. By conducting an extensive set of experiments based on real BGP traces, Chapter 5 shows that there are spatial and temporal locality properties in BGP updates, which can be leveraged to improve upon online FIB aggregation algorithms. An according algorithm,

LFA, is proposed and evaluated in the context of FIB aggregation alone, and in conjunction with TFO to further boost the traffic offloading performance. The results show that FIB aggregation can improve the traffic offloading performance of TFO in an SDN Router, while at the same time keeping the churn low.

**OFLOPS:** A combined software/hardware-framework for benchmarking OpenFlow-enabled switches.

Chapter 6 raises the need for an accurate understanding of the critical performance impairments of OpenFlow-enabled switches. This is crucial not only in the SDN Router context, but also in most other network scenarios which involve OpenFlow-enabled switches. OFLOPS is a flexible and extensible software framework which incorporates advanced hardware instrumentation for timing accuracy and performance. Using OFLOPS against five different OpenFlow implementations unveils considerable variations in their data plane and control plane performance, as well as in their level of support of the OpenFlow specification. The chapter points out, that implementing even the simple OpenFlow 1.0 protocol on an existing hardware switch is non-trivial. In conclusion, this chapter suggests to thoroughly test a switch's performance in the required OpenFlow features and their complex interactions before building application of top of it.

**Analysis of an IXP:** An in-depth analysis of one of the largest IXPs worldwide.

Chapter 7 sheds light onto the ecosystem of a large European IXP by analyzing nine month's worth of sFlow records collected in 2011. IXPs are gaining importance in the Internet as more and more ASes become IXP members and establish peering relationships with other ASes through the IXP's public peering fabric. Although the number of member ASes as well as their identities are public, it is largely unknown – even to the IXP operator – how much they peer and what traffic they exchange through the IXP's infrastructure. This study unveils an astonishingly high number of established peering links, approximately 50,000, corresponding to 67 % of all possible peering links among members of that IXP, and even more than the current estimates of the total number of peering links in the entire Internet. Moreover, most of these peering links are invisible from outside the IXP. As a consequence, the results of this study suggest a re-assessment of the mental picture that the research community has about the Internet ecosystem and the AS-level structure of the Internet.

**Analysis of the World IPv6 Day:** A close investigation of IPv6 traffic as observed at an IXP and a campus network.

Chapter 8 analyzes the progress of the adoption of IPv6 in the Internet, based on traffic traces collected at an IXP and at the campus of a major US university. The World IPv6 Day, which was a coordinated global test-run of the new version of the Internet protocol in June 2011, had a substantial impact: Native IPv6 traffic has almost doubled in volume. IPv6 traffic transported over transition technologies (tunnels) did not change much. Teredo tunnels contribute a large fraction of the overall IPv6 traffic, yet they only carry control traffic. 6in4 tunnels carry large amounts of fragmented IPv6 packets for which broken software is a likely cause. The results further show that the IPv6 application mix is becoming similar to the IPv4 one, suggesting that IPv6 is getting closer to becoming mainstream.

## 9.2 Future directions

This thesis encourages a lot of further research. We provide directions for further studies by chapter.

Both the SDN Router concept in Chapter 3 and its prototype implementation focus on building a single IP router based on one switch and one PC. We propose to design a distributed SDN Router-like concept, in which a logically centralized version of RouteVisor acts as the routing control plane for multiple OpenFlow-enabled switches. This raises a number of new challenges, including: In which order should routing table updates be applied to the different switches on route changes? Can the logically centralized view on the topology be leveraged to improve BGP's best path selection algorithm?

To follow-up on the TFO study (Chapter 4) we propose to find a proper bootstrapping-phase for this algorithm. On a fresh start of a router, no traffic statistics are available for TFO to base its prefix selection on. Another challenge is to find ways to effectively shield TFO from attacks. Questions include: Can an attacker inject traffic that enforces a high rate of churn in the offloaded prefixes? Can an attacker overload the slow forwarding path through the PC by injecting especially crafted traffic patterns?

The LFA algorithm (Chapter 5) is mainly designed for analyzing the spatial and temporal locality properties in BGP updates. We propose studying possible advances to existing locality-unaware online FIB aggregation algorithms, such as SMALTA [144]. In addition, in the traffic offloading case, in which limiting churn in the offloaded prefixes is a major goal, we propose to explicitly inform the FIB aggregation algorithm about which prefixes are offloaded. Going one step further, we suggest integrating LFA (or FIB aggregation in general) and TFO into one algorithm, the main question being: Can FIB aggregation, while being aware of the offloaded prefixes, minimize churn by limiting its updates to the heavy hitters?

The OFLOPS framework (Chapter 6) in its current form integrates the 1G version of the NetFPGA platform [91] for performance and timing accuracy, a bottleneck being the maximum packet rate and thus the timestamping granularity. With the new 10G NetFPGA now available, we propose to port the OFLOPS NetFPGA core to the new platform. Also, OFLOPS currently focuses entirely on OpenFlow 1.0 [110]. The subsequent versions 1.1 and 1.2 [109] however are major overhauls of the protocol and so much more advanced, that it seems already challenging to define an appropriate set of tests.

Our IXP study (Chapter 7) suggests a further *flattening* of the AS-level Internet as we have identified a large number of peering links that are invisible in public routing data. However, we are still missing a clear understanding of what causes links to be invisible. For example, what is the role of routing policies in hiding existing peering links at an IXP? Can an in-depth investigation of the root causes of hidden links help to design better topology estimation experiments and, with that, increase the accuracy of measurement-based estimations of the AS-level Internet?

The World IPv6 Day [150] in 2011 (Chapter 8) was the first global test-run of the IPv6 protocol and our results show a sustained increase in IPv6 traffic. One year later, a number of major ISPs and other organizations have agreed to permanently enable IPv6 at the World IPv6 Launch [152] in June 2012, which calls for further measurement studies. In addition, the number of IPv6 routing table entries has doubled in 2011 alone [70]. This suggests an analysis of the dynamics of IPv6 prefixes in BGP, and an investigation of the extend to which the AS-level topology defined by only IPv6-enabled peering relationships matches to the IPv4 one.

Questions for future research that span across all thesis topics include: How would an SDN Router for IXP members look like? Can the IXP public peering fabric be improved with the help of SDN and the traffic offloading ideas behind TFO? Can the traffic offloading concept be implemented in hardware on router line cards? How would an IPv6-capable SDN Router look like? How does FIB aggregation perform on IPv6 routing tables?

# Acknowledgments

I would like to express my gratitude to my advisor Anja Feldmann, who showed me support and guidance throughout my work on this thesis. This thesis would not have been possible without her sharp, honest, and always constructive feedback. I am especially thankful for my two stays in the Silicon Valley, which allowed me to breathe some actual Silicon Valley air. Anja, thanks for the last three years which were dominated by a series of unique and lasting experiences, far beyond my expectations!

Next, I owe sincere thankfulness to Steve Uhlig, who was a true partner in crime all along the way. Besides his broad and deep knowledge of the subject, writing papers with him was especially exciting and fun, and we never restrained a laughter. Steve, thanks for the countless hours we spent as colleagues and friends!

I would like to thank Roch Guérin for serving as a member in my thesis committee. When I first presented my thesis summary to Roch he immediately had insightful comments which helped me to further shape my thesis. Roch, thanks for your comprehensive and inspiring feedback and for making my thesis defense such a great experience!

At a time difference of nine hours, I would like to thank my colleagues and friends Rob Sherwood and Srini Seetharaman for making my Stanford stays such enjoyable and invaluable times. Also, the anonymous data providers such as the IXP operator deserve huge thanks for their trust and support!

Now it is my great pleasure to thank everyone at FG INET who helped me write my dissertation successfully, through research collaborations and/or fun distractions that most of us need every now and then. Andi, thanks for the great times we shared in and around California, this is a friendship that has already proven to outlast the work environment! Amir, you are a great friend and I hope to see you again soon! Robert, you were the perfect student worker and a great MSc candidate, you can look toward a bright future! Special thanks to those at FG INET who play table soccer, especially the Ph.D. candidate youngsters Carlo and Arne Gómez who showed me how little my skill in table soccer actually was. I would also like to thank the secretary and administrative staff at FG INET for making the day-to-day challenges a breeze!

Last but not least I would like to show my deep gratitude to my family and friends, who have always been there for me when needed, and who were happy for me even when I was gone for months. Thank you mum and dad, brother and sister-in-law, for everything and in particular for proof-reading the German translation of my thesis abstract. I would also like to thank my uncle Heiko who changed his vacation plans just to attend my thesis defense. With all of you there my defense day was a memorable experience!

Finally, thank you FG INET for the most amazing doctoral hat I've ever seen. Stay in touch, the story continues!

# List of Figures

# List of Tables

# Bibliography


[1] Adhikari, V. K., Jain, S., and Zhang, Z.-L. YouTube Traffic Dynamics and its Interplay with a Tier-1 ISP: An ISP Perspective. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2010).

[2] Agarwal, S., Chuah, C., Bhattacharyya, S., and Diot, C. The Impact of BGP Dynamics on Intra-Domain Traffic. In *Proceedings of the ACM SIGMETRICS* (2004).

[3] Agilent. N2X Router Tester. `http://advanced.comms.agilent.com/n2x/`.

[4] Arista Networks. EOS: An Extensible Operating System. `www.aristanetworks.com/en/EOS`, 2009.

[5] Arlos, P., and Fiedler, M. A Method to Estimate the Timestamp Accuracy of Measurement Hardware and Software Tools. In *Proceedings of the Passive and Active Measurement Conference (PAM)* (2007).

[6] AdvancedTCA Specifications for Next Generation Telecommunications Equipment. `http://www.advancedtca.org/`.

[7] AT&T Company Information, 2012. `http://www.att.com/gen/investor-relations?pid=5711`.

[8] Augustin, B., Krishnamurthy, B., and Willinger, W. IXPs: Mapped? In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2009).

[9] Aweya, J. On the Design of IP Routers Part 1: Router Architectures. *Journal of Systems Architecture* (2000).

[10] Balestra, G., Luciano, S., Pizzonia, M., and Vissicchio, S. Leveraging Router Programmability for Traffic Matrix Computation. In *Proceedings of the PRESTO workshop* (2010).

[11] Ballani, H., Francis, P., Cao, T., and Wang, J. Making Routers Last Longer with ViAggre. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2009).

[12] Bharti, V., Kankar, P., Setia, L., Gürsun, G., Lakhina, A., and Crovella, M. Inferring Invisible Traffic. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (2010).

[13] The BIRD Internet Routing Daemon. `http://bird.network.cz/`.

[14] Brzozowski, J., and Griffiths, C. Comcast IPv6 Trial/Deployment Experiences, 2011. Internet-Draft: draft-jjmb-v6ops-comcast-ipv6-experiences-01.

[15] CAIDA AS Ranking. http://as-rank.caida.org/.

[16] Cardona Restrepo, J. C., and Stanojevic, R. A History of an Internet eXchange Point. *ACM SIGCOMM Computer Communication Review (CCR)* (2012).

[17] Castro, I., and Gorinsky, S. T4P: Hybrid Interconnection for Cost Reduction. In *Proceedings of the Workshop on the Economics of Networks, Systems and Computation (NetEcon)* (2012).

[18] Chang, H., Govindan, R., Jamin, S., Shenker, S., and Willinger, W. Towards Capturing Representative AS-Level Internet Topologies. *Computer Networks* (2004).

[19] Chang, H., Jamin, S., Mao, Z., and Willinger, W. An Empirical Approach to Modeling Inter-AS Traffic Matrices. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2005).

[20] Chang, H., Roughan, M., Uhlig, S., Alderson, D., and Willinger, W. The Many Facets of Internet Topology and Traffic. *Networks and Heterogeneous Media* (2006).

[21] Chen, K., Choffnes, D. R., Potharaju, R., Chen, Y., Bustamante, F. E., Pei, D., and Zhao, Y. Where the Sidewalk Ends: Extending the Internet AS Graph Using Traceroutes from P2P Users. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (2009).

[22] Chi, Y.-J., Oliveira, R., and Zhang, L. Cyclops: The AS-level Connectivity Observatory. *ACM SIGCOMM Computer Communication Review (CCR)* (2008).

[23] Cho, K., Luckie, M., and Huffaker, B. Identifying IPv6 Network Problems in the Dual-Stack World. In *Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice meet Malfunctioning Reality* (New York, NY, USA, 2004), NetT '04, ACM.

[24] Cho, K., Mitsuya, K., and Kato, A. Traffic Data Repository at the WIDE Project. In *Proceedings of the USENIX Annual Technical Conference (ATC), Freenix track* (2000).

[25] Cisco. How to Choose the Best Router Switching Path for Your Network. http://www.cisco.com/warp/public/105/20.pdf, 2005.

[26] Claffy, K. C. Tracking IPv6 Evolution: Data We Have and Data We Need. *ACM SIGCOMM Computer Communication Review (CCR) 41* (2011).

[27] Claffy, K. C., and Brownlee, N. Understanding Internet Traffic Streams: Dragonflies and Tortoises. *IEEE Communications Magazine* (2002).

[28] Cohen, E., Duffield, N., Kaplan, H., Lund, C., and Thorup, M. Algorithms and Estimators for Accurate Summarization of Internet Traffic. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2007).

[29] Cohen, R., and Raz, D. The Internet Dark Matter – On the Missing Links in the AS Connectivity Map. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)* (2006).

[30] Cormode, G., Korn, F., Muthukrishnan, S., and Srivastava, D. Finding Hierarchical Heavy Hitters in Data Streams. In *Proceedings of the International Conference on Very Large Databases (VLDB)* (2003).

[31] Covington, G., Gibb, G., Lockwood, J., and Mckeown, N. A Packet Generator on the NetFPGA Platform. In *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines (FCCM)* (2009).

[32] Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., and Banerjee, S. DevoFlow: Scaling Flow Management for High-Performance Networks. In *Proceedings of the ACM SIGCOMM* (2011).

[33] The DAG project, University of Waikato. `http://dag.cs.waikato.ac.nz/`.

[34] Deering, S., and Hinden, R. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564.

[35] Deutsche Telekom ICSS, 2012. `http://ghs-internet.telekom.de/dtag/cms/content/ICSS/en/1222498`.

[36] Dhamdhere, A., and Dovrolis, C. Ten Years in the Evolution of the Internet Ecosystem. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2008).

[37] Dhamdhere, A., and Dovrolis, C. The Internet is Flat: Modeling the Transition from a Transit Hierarchy to a Peering Mesh. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (2010).

[38] Dhamdhere, A., and Dovrolis, C. Twelve Years in the Evolution of the Internet Ecosystem. *IEEE/ACM Transactions on Networking (ToN)* (2011).

[39] Dobrescu, M., Egi, N., Argyraki, K., Chun, B., Fall, K., Iannaccone, G., Knies, A., Manesh, M., and Ratnasamy, S. RouteBricks: Exploiting Parallelism to Scale Software Routers. In *Proceedings of the ACM SOSP* (2009).

[40] Dolev, D., Jamin, S., Mokryn, O., and Shavitt, Y. Internet Resiliency to Attacks and Failures under BGP Policy Routing. *Computer Networks* (2006).

[41] Draves, R. P., King, C., Venkatachary, S., and Zill, B. D. Constructing Optimal IP Routing Tables. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)* (1999).

[42] Dreger, H., Feldmann, A., Mai, M., Paxson, V., and Sommer, R. Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection. In *Proceedings of the USENIX Security Symposium* (2006).

[43] Edwards, J. Enterprises Cut Costs With Open-Source Routers. `http://www.computerworld.com/s/article/9133851`, 2009.

[44] Egi, N., Greenhalgh, A., Handley, M., Hoerdt, M., Huici, F., and Mathy, L. Towards High Performance Virtual Routers on Commodity Hardware. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (2008).

[45] Elmokashfi, A., Kvalbein, A., and Dovrolis, C. BGP Churn Evolution: A Perspective from the Core. *IEEE/ACM Transactions on Networking (ToN) 20*, 2 (2012).

[46] Estan, C., Keys, K., Moore, D., and Varghese, G. Building a Better NetFlow. In *Proceedings of the ACM SIGCOMM* (2004).

[47] Euro-IX – European Internet Exchange Association. `https://www.euro-ix.net/resources`.

[48] Fan, J., Xu, J., Ammar, M. H., and Moon, S. Prefix-Preserving IP Address Anonymization: Measurement-Based Security Evaluation and a New Cryptography-Based Scheme. *Computer Networks* (2004).

[49] Fang, W., and Peterson, L. Inter-AS Traffic Patterns and their Implications. In *Proceedings of the IEEE Global Internet* (1999).

[50] Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J., and True, F. Deriving Traffic Demands for Operational IP networks: Methodology and Experience. *IEEE/ACM Transactions on Networking (ToN)* (2001).

[51] Fouli, K., and Maier, M. The Road to Carrier-Grade Ethernet. *IEEE Communications Magazine* (2009).

[52] Freedman, D. A., Marian, T., Lee, J. H., Birman, K., Weatherspoon, H., and Xu, C. Exact Temporal Characterization of 10 Gbps Optical Wide-Area Network. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2010), IMC 2010.

[53] Gadkari, K., Massey, D., and Papadopoulos, C. Dynamics of Prefix Usage at An Edge Router. In *Proceedings of the Passive and Active Measurement Conference (PAM)* (2011).

[54] Gill, P., Schapira, M., and Goldberg, S. Let the Market Drive Deployment: A Strategy for Transitioning to BGP Security. In *Proceedings of the ACM SIGCOMM* (2011).

[55] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. NOX: Towards an Operating System for Networks. *ACM SIGCOMM Computer Communication Review (CCR)* (2008).

[56] Guérin, R., and Hosanagar, K. Fostering IPv6 Migration Through Network Quality Differentials. *ACM SIGCOMM Computer Communication Review (CCR) 40* (2010).

[57] Haddadi, H., Iannaccone, G., Moore, A., Mortier, R., and Rio, M. Network Topologies: Inference, Modelling and Generation. *IEEE Communications Surveys and Tutorials* (2008).

[58] Halabi, B., and Pherson, D. M. *Internet Routing Architectures (2nd Edition).* Cisco Press, 2000.

[59] Han, S., Jang, K., Park, K., and Moon, S. PacketShader: a GPU-Accelerated Software Router. In *Proceedings of the ACM SIGCOMM* (2010).

[60] Handigol, N., Seetharaman, S., Flajslik, M., McKeown, N., and Johari, R. Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow. In *Proceedings of the ACM SIGCOMM, Demo session* (2009).

[61] Handley, M., Hodson, O., and Kohler, E. XORP: An Open Platform for Network Research. *ACM SIGCOMM Computer Communication Review (CCR) 33*, 1 (2003).

[62] He, Y., Siganos, G., Faloutsos, M., and Krishnamurthy, S. A Systematic Framework for Unearthing the Missing Links: Measurements and Impact. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2007).

[63] He, Y., Siganos, G., Faloutsos, M., and Krishnamurthy, S. Lord of the Links: A Framework for Discovering Missing Links in the Internet Topology. *IEEE/ACM Transactions on Networking (ToN)* (2009).

[64] Hinden, B. Check Point's World IPv6 Day Experience. In *Proceedings of the IETF 81 V6OPS* (2011). `http://www.ietf.org/proceedings/81/slides/v6ops-2.pptx`.

[65] Huitema, C. *Routing in the Internet (2nd ed.)*. Prentice Hall, 1999.

[66] Huitema, C. An Anycast Prefix for 6to4 Relay Routers. RFC 3068 (Proposed Standard), 2001.

[67] Huitema, C. Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs). RFC 4380 (Proposed Standard), 2006. Updated by RFCs 5991, 6081.

[68] Hurricane Electric IPv4 Exhaustion Counters. `http://ipv6.he.net/statistics/`.

[69] Huston, G. BGP Routing Table Analysis Reports. `http://bgp.potaroo.net/`.

[70] Huston, G. IPv6 Reports. `http://bgp.potaroo.net/index-v6.html`.

[71] Huston, G. Interconnections, Peering and Financial Settlements. In *Proceedings of the Annual Conference of the Internet Society (INET)* (1999).

[72] Internet World Stats. `http://www.internetworldstats.com/stats.htm`.

[73] IP Infusion ZebOS. `http://www.ipinfusion.com/`.

[74] Ixia. Interfaces. `http://www.ixiacom.com/`.

[75] Jain, R. Characteristics of Destination Address Locality in Computer Networks: A Comparison of Caching Schemes. *Computer Networks and ISDN* (1989/90).

[76] Jose, L., Yu, M., and Rexford, J. Online Measurement of Large Traffic Aggregates on Commodity Switches. In *Proceedings of the USENIX HotICE Workshop* (2011).

[77] Kim, C., Caesar, M., Gerber, A., and Rexford, J. Revisiting Route Caching: The World Should Be Flat. In *Proceedings of the Passive and Active Measurement Conference (PAM)* (2009).

[78] Kim, J., Schneider, F., Ager, B., and Feldmann, A. Today's Usenet Usage: Characterizing NNTP Traffic. In *Proceedings of the IEEE Global Internet* (2010).

[79] Kleinrock, L. The Birth of the Internet. `http://www.lk.cs.ucla.edu/personal_history.html`.

[80] Klema, V., and Laub, A. The Singular Value Decomposition: Its Computation and Some Applications. *IEEE Transactions on Automatic Control* (1980).

[81] Kocak, T., and Basci, F. A Power-Efficient TCAM Architecture for Network Forwarding Tables. *Journal of Systems Architecture* (2006).

[82] Kushman, N., Kandula, S., and Katabi, D. Can You Hear Me Now?!: It Must be BGP. *ACM SIGCOMM Computer Communication Review (CCR)* (2007).

[83] Labovitz, C. Six Month, Six Providers and IPv6. Tech. rep., Arbor Networks, 2011. http://www.monkey.org/~labovit/papers/v6sixmonths.pdf.

[84] Labovitz, C., Lekel-Johnson, S., McPherson, D., Oberheide, J., and Jahanian, F. Internet Inter-Domain Traffic. In *Proceedings of the ACM SIGCOMM* (2010).

[85] Labovitz, C., Malan, G., and Jahanian, F. Internet Routing Instability. In *Proceedings of the ACM SIGCOMM* (1997).

[86] Lakhina, A., Crovella, M., and Diot, C. Diagnosing Network-Wide Traffic Anomalies. In *Proceedings of the ACM SIGCOMM* (2004).

[87] Levy, M. IETF 81 – World IPv6 Day Operators Review. In *Proceedings of the IETF 81 V6OPS* (2011). http://www.ietf.org/proceedings/81/slides/v6ops-19.pdf.

[88] Li, J., Guidero, M., Wu, Z., Purpus, E., and Ehrenkranz, T. BGP Routing Dynamics Revisited. *ACM SIGCOMM Computer Communication Review (CCR) 37* (2007).

[89] Limoncelli, T. A., and Cerf, V. G. Successful Strategies for IPv6 Rollouts. Really. *Communications of the ACM 54* (2011).

[90] Liu, Y., Zhao, X., Nam, K., Wang, L., and Zhang, B. Incremental Forwarding Table Aggregation. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)* (2010).

[91] Lockwood, J. W., McKeown, N., Watson, G., Gibb, G., Hartke, P., Naous, J., Raghuraman, R., and Luo, J. NetFPGA – An Open Platform for Gigabit-Rate Network Switching and Routing. In *Proceedings of the IEEE MSE* (2007).

[92] Maennel, O., and Feldmann, A. Realistic BGP Traffic for Test Labs. In *Proceedings of the ACM SIGCOMM* (2002).

[93] Maier, G., Feldmann, A., Paxson, V., and Allman, M. On Dominant Characteristics of Residential Broadband Internet Traffic. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2009).

[94] GeoLite City. http://www.maxmind.com/app/geolitecity/.

[95] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review (CCR)* (2008).

[96] Meyer, D., Zhang, L., and Fall, K. Report from the IAB Workshop on Routing and Addressing. RFC 4984 (Informational), 2007.

[97] Moy, J. *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley, 1998.

[98] Muhlbauer, W., Feldmann, A., Maennel, O., Roughan, M., and Uhlig, S. Building an AS-Topology Model that Captures Route Diversity. In *Proceedings of the ACM SIGCOMM* (2006).

[99] Naous, J., Erickson, D., Covington, G. A., Appenzeller, G., and McKeown, N. Implementing an OpenFlow Switch on the NetFPGA Platform. In *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications (ANCS)* (2008).

[100] Nordmark, E., and Gilligan, R. Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC 4213 (Proposed Standard), 2005.

[101] Norton, W. B. The Evolution of the U.S. Internet Peering Ecosystem. *Equinix White Papers* (2004).

[102] Norton, W. B. A Business Case for Peering in 2010, 2010. `http://drpeering.net/`.

[103] Norton, W. B. A Study of 28 Peering Policies, 2010. `http://drpeering.net/`.

[104] Norton, W. B. Public vs Private Peering: The Great Debate, 2010. `http://drpeering.net/`.

[105] OECD. Broadband Portal. www.oecd.org/sti/ict/broadband, 2008.

[106] Oliveira, R., Pei, D., Willinger, W., Zhang, B., , and Zhang, L. In Search of the Elusive Ground Truth: The Internet's AS-Level Connectivity Structure. In *Proceedings of the ACM SIGMETRICS* (2008).

[107] Oliveira, R., Pei, D., Willinger, W., Zhang, B., and Zhang, L. The (In)completeness of the Observed Internet AS-Level Structure. *IEEE/ACM Transactions on Networking (ToN)* (2010).

[108] Olsson, R. pktgen the Linux Packet Generator. In *Proceedings of the Linux symposium* (2005).

[109] Open Networking Foundation. `http://www.opennetworking.org/`.

[110] OpenFlow Switch Specification (version 1.0.0), 2009. `www.openflow.org/documents/openflow-spec-v1.0.0.pdf`.

[111] Oran, D. OSI IS-IS Intra-Domain Routing Protocol. RFC 1142 (Informational), 1990.

[112] Palmer, C., and Thaler, D. World IPv6 Day at Microsoft. In *Proceedings of the IETF 81 V6OPS* (2011). `http://www.ietf.org/proceedings/81/slides/v6ops-1.pptx`.

[113] Papagiannaki, K., Taft, N., and Diot, C. Impact of Flow Dynamics on Traffic Engineering Design Principles. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)* (2004).

[114] Paxson, V. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks Journal 31*, 23–24 (1999). Bro homepage: `http://www.bro-ids.org`.

[115] PeeringDB. `http://www.peeringdb.com/`.

[116] Perros, H. G. *Connection-Oriented Networks: SONET/SDH, ATM, MPLS and Optical Networks*. Wiley, 2005.

[117] Pettit, J., Gross, J., Pfaff, B., Casado, M., and Crosby, S. Virtualizing the Network Forwarding Plane. In *Proceedings of the 2nd Workshop on Data Center - Converged and Virtual Ethernet Switching (DC-CAVES)* (2010).

[118] Phaal, P., Panchen, S., and McKee, N. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176 (Informational), 2001.

[119] Poese, I., Uhlig, S., Kaafar, M. A., Donnet, B., and Gueye, B. IP Geolocation Databases: Unreliable? *ACM SIGCOMM Computer Communication Review (CCR)* (2011).

[120] Quagga Routing Suite. `http://www.quagga.net/`.

[121] Raszuk, R., Heitz, J., Lo, A., Zhang, L., and Xu, X. Simple Virtual Aggregation (S-VA), 2012. Internet-Draft (Informational): draft-ietf-grow-simple-va-12.txt.

[122] Rekhter, Y., Li, T., and Hares, S. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), 2006. Updated by RFCs 6286, 6608.

[123] Renesys. `http://renesys.com/`.

[124] Rexford, J., Wang, J., Xiao, Z., and Zhang, Y. BGP Routing Stability of Popular Destinations. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2002).

[125] Routing Information Service (RIS) – RIPE Network Coordination Center. `http://www.ris.ripe.net/`.

[126] Rothenberg, C. E., Nascimento, M. R., Salvador, M. R., Corrêa, C. N. A., Cunha de Lucena, S., and Raszuk, R. Revisiting routing control platforms with the eyes and muscles of software-defined networking. In *Proceedings of the ACM First Workshop on Hot Topics in Software Defined Networks (HotSDN)* (2012).

[127] Roughan, M. Robust Network Planning. In *Guide to Reliable Internet Services and Applications*. Springer, 2009.

[128] Roughan, M., Willinger, W., Maennel, O., Perouli, D., and Bush, R. 10 Lessons from 10 Years of Measuring and Modeling the Internet's Autonomous Systems. *IEEE Journal on Selected Areas in Communications (JSAC)* (2012).

[129] Route Views Project, University of Oregon. `http://www.routeviews.org/`.

[130] Schapira, M., Zhu, Y., and Rexford, J. Putting BGP on the Right Path: Better Performance via Next-Hop Routing. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)* (2010).

[131] InMon sFlow Toolkit. `http://www.inmon.com/technology/sflowTools.php`.

[132] Shah, D., and Gupta, P. Fast Incremental Updates on Ternary-CAMs for Routing Lookups and Packet Classification. *IEEE Micro* (2001).

[133] Shaikh, A., and Greenberg, A. Experience in Black-Box OSPF Measurement. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2001).

[134] Sherwood, R., Gibb, G., Yapa, K., Cassado, M., Appenzeller, G., McKeown, N., and Parulkar, G. Can the Production Network be the Test-Bed? In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (2010).

[135] Shyu, W., Wu, C., and Hou, T. Efficiency Analyses on Routing Cache Replacement Algorithms. In *Proceedings of the International Conference on Communications (ICC)* (2002).

[136] Simcoe, R. J., and Pei, T.-B. Perspectives on ATM Switch Architecture and the Influence of Traffic Pattern Assumptions on Switch Design. *ACM SIGCOMM Computer Communication Review (CCR)* (1995).

[137] Software-Defined Networking: The New Norm for Networks, 2012. ONF White Paper.

[138] Srinivasan, V., and Varghese, G. Fast Address Lookups Using Controlled Prefix Expansion. *IEEE/ACM Transactions on Networking (ToN)* (1999).

[139] Stanojevic, R., Castro, I., and Gorinsky, S. CIPT: Using Tuangou to Reduce IP Transit Costs. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (2011).

[140] Suri, S., Sandholm, T., and Warkhede, P. R. Compressing Two-Dimensional Routing Tables. *Proceedings of the Algorithmica 35*, 4 (2003).

[141] Taylor, D. E., Lockwood, J. W., Sproull, T. S., Turner, J. S., and Parlour, D. B. Scalable IP Lookup for Programmable Routers. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)* (2002).

[142] Tootoonchian, A., Ghobadi, M., and Ganjali, Y. OpenTM: Traffic Matrix Estimator for OpenFlow Networks. In *Proceedings of the Passive and Active Measurement Conference (PAM)* (2010).

[143] Townsley, W., and Troan, O. IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) – Protocol Specification. RFC 5969 (Proposed Standard), 2010.

[144] Uzmi, Z. A., Nebel, M., Tariq, A., Jawad, S., Chen, R., Shaikh, A., Wang, J., and Francis, P. SMALTA: Practical and Near-Optimal FIB Aggregation. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (2011).

[145] Valancius, V., Lumezanu, C., Feamster, N., Johari, R., and Vazirani, V. How Many Tiers? Pricing in the Internet Transit Market. In *Proceedings of the ACM SIGCOMM* (2011).

[146] Wallerich, J., Dreger, H., Feldmann, A., Krishnamurthy, B., and Willinger, W. A Methodology for Studying Persistency Aspects of Internet Flows. *ACM SIGCOMM Computer Communication Review (CCR)* (2005).

[147] Wallerich, J., and Feldmann, A. Capturing the Variability of Internet Flows Across Time. In *Proceedings of the IEEE Global Internet* (2006).

[148] Wang, L., Zhao, X., Pei, D., Bush, R., Massey, D., Mankin, A., Wu, S., and Zhang, L. Observation and Analysis of BGP Behavior under Stress. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)* (2002).

[149] Whittle, R. SRAM-Based IP Forwarding Eliminates the Need for Route Aggregation. Internet-Draft, draft-whittle-sram-ip-forwarding-01.txt, work in progress, 2007.

[150] Wijnen, B., Aben, E., Wilhelm, R., and Kisteleki, R. World IPv6 Day – What did we learn? In *Proceedings of the IETF 81 V6OPS* (2011). http://www.ietf.org/proceedings/81/slides/v6ops-4.pdf.

[151] Woodcock, B., and Adhikari, V. Survey of Characteristics of Internet Carrier Interconnection Agreements, 2011. http://www.pch.net/docs/papers/peering-survey/.

[152] Infographic: World IPv6 Launch by the Numbers. http://www.worldipv6launch.org/infographic/.

[153] Xu, K., Duan, Z., Zhang, Z.-L., and Chandrashekar, J. On Properties of Internet Exchange Points and their Impact on AS Topology and Relationship. In *Proceedings of the IFIP Networking* (2004).

[154] Yap, K.-K., Kobayashi, M., Underhill, D., Seetharaman, S., Kazemian, P., and McKeown, N. The Stanford OpenRoads Deployment. In *Proceedings of the ACM WINTECH* (2009).

[155] Yu, M., Rexford, J., Freedman, M. J., and Wang, J. Scalable Flow-Based Networking with DIFANE. *ACM SIGCOMM Computer Communication Review (CCR) 40* (2010).

[156] Zane, F., Narlikar, G., and Basu, A. CoolCAMs: Power-Efficient TCAMs for Forwarding Engines. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)* (2003).

[157] Zhang, Y., Breslau, L., Paxson, V., and Shenker, S. On the Characteristics and Origins of Internet Flow Rates. In *Proceedings of the ACM SIGCOMM* (2002).

[158] Zhang, Y., Roughan, M., Willinger, W., and Qiu, L. Spatio-Temporal Compressive Sensing and Internet Traffic Matrices. In *Proceedings of the ACM SIGCOMM* (2009).

[159] Zhang, Y., Singh, S., Sen, S., Duffield, N., and Lund, C. Online Identification of Hierarchical Heavy Hitters: Algorithms, Evaluation, and Applications. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2004).

[160] Zhang, Y., Zhang, Z., Mao, Z., Hu, C., and Maggs, B. On the Impact of Route Monitor Selection. In *Proceedings of the ACM Internet Measurement Conference (IMC)* (2007).

[161] Zhao, X., Liu, Y., Wang, L., and Zhang, B. On the Aggregatability of Router Forwarding Tables. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)* (2010).