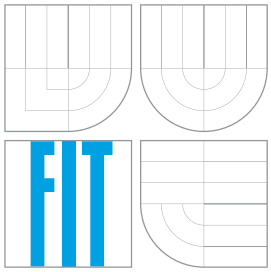
BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

USER ACCOUNTING IN NEXT GENERATION NETWORKS

ÚČTOVÁNÍ UŽIVATELŮ V SÍTÍCH NOVÉ GENERACE

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE
Ing. MATĚJ GRÉGR

SUPERVISOR
ŠKOLITEL
prof. Ing. MIROSLAV ŠVÉDA, CSc.

BRNO 2016

Abstract

The number of devices connected to the Internet is such enormous that it is impossible to assign a globally unique address to every device in today's TCP/IPv4 architecture. Since the discussion how to solve the problem began in 1990s, there has been several proposals of new protocols and architectures trying to solve the problem. However, the only proposal that is widely deployed today is the IPv6 protocol. The IPv6 protocol enlarges the network address size, thus, it is possible to assign a globally unique IPv6 address to unlimited number of devices. Furthermore, the protocol introduces a paradigm shift, especially for address assignment and length of the IPv6 header. The IPv6 protocol is, however, incompatible with IPv4. To overcome the limitation, several transition mechanisms were proposed. The thesis discusses issues introduced by IPv6 protocol to user accounting process. In particular, it focuses on new approaches that can eliminate problems of current accounting methods that use NetFlow or IPFIX protocols. The aim of the thesis is to solve user accounting process for networks running a transition mechanism or a network address translation. Part of the thesis discusses the global IPv6 deployment and measures IPv6 penetration among content providers, internet service providers and transit operators.

Abstrakt

Velikost sítě Internet dosáhla takového rozměru, že globálně jednoznačná adresace všech připojených zařízení již není možná při zachování současné architektury TCP/IPv4. Tímto problémem se začalo zabývat již v 90. letech a od té doby bylo představeno několik návrhů nových architektur a síťových protokolů, které mají či měly ambice omezení adresace vyřešit. V současné době, v roce 2016, je jediným globálně nasazovaným řešením problému adresace protokol IPv6. Tento protokol zvětšuje velikosti síťové adresy čímž umožňuje adresovat téměř libovolné množství zařízení, ovšem za cenu nekompatibility se současným protokolem IPv4. Rozdílně se také staví ke způsobu automatické konfigurace koncových zařízení, proměnlivé velikosti síťové hlavičky a omezení nekompatibility řeší různými přechodovými mechanismy. Tato práce diskutuje dopady, které tyto změny mají na oblast monitorování a účtování uživatelů. Zejména změny způsobu konfigurace adresy vyžadují jiný přístup než v současných monitorovacích systémech, které ukládají pouze metadata o síťové komunikace pomocí protokolu NetFlow/IPFIX. Práce je zaměřena primárně na vyřešení problému účtování uživatelů v sítích kde jsou souběžně nasazeny protokoly IPv4 i IPv6, použity tunelovací přechodové mechanismy nebo překlad adres. Část práce je zaměřena na měření globálního vývoje a nasazení protokolu IPv6 mezi koncovými poskytovateli internetového připojení, poskytovateli obsahu a páteřními operátory.

Keywords

User accounting, IPv6, measuring, deployment, NAT, CGN, NetFlow, IPFIX

Klíčová slova

NetFlow, NAT, IPv6, monitorování sítě, zavádění IPv6

Reference

GRÉGR, Matěj. *User accounting in next generation networks.* Brno, 2016. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Švéda Miroslav.

User accounting in next generation networks

Declaration

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně pod vedením pana prof. Ing. Miroslava Švédy, CSc.

.....
Matěj Grégr
August 10, 2016

Acknowledgements

Děkuji Karolíně, za to, že se mnou stále je. Mamince, taťkovi a sestře, že mě mají rádi. Vladimírovi, Marii a Sváťovi za skvělé mudrovačky. Mirkovi Švédovi za velkou podporu v průběhu studia. Petrovi Matouškovi, Ondrovi Ryšavému a Jardovi Rábovi za cenné připomínky a diskuze. Tomášovi Podermańskému za dlouhé debaty o všem. Honzovi Drápelovi za príma středy. Celé řadě dalších kolegů a přátel. Díky moc všem!

Contents

List of Figures

List of Tables

1

Introduction

The Internet has traditionally been used by research and educational organizations. End users had a free access to the network, with the actual costs absorbed by budgets of their research or government agencies [3]. Because of funding by research organizations and government, there has been little or no requirement to collect data at the level of an individual user, because an accounting of the user was not necessary. The network operators usually collected only aggregated data that were used for network management.

This model has been abandoned by the commercialization of the Internet in the 1990s. Commercial network operators need more detailed information to prevent violation of their service-level agreements or enforcing quality of service. The requirement to collect data at the level of an individual user has been increasing since then and the lack of detailed data usage for individual users becomes a serious shortcoming.

The increasing number of users demanding access to the Internet from their smart phones and tablets in recent years is putting even bigger emphasis on the requirement for detailed information about the individual user. The reason is that the today's users' access to the Internet from a mobile device is usually not based on a flat monthly fee for unlimited access as in broadband but it is limited. Internet Service Providers (ISPs) thus need a robust system to account the users knowing exactly the number of transferred bytes per month and other information. Solving security incidents, trace back virus propagation from an infected computer in a network can be further examples, why user monitoring and accounting play an important role in ISP networks today.

Reasons for user accounting and monitoring mentioned above are primarily oriented to business cases or network management. However, there is another reason given by legal requirements. There has been always a demand from government and law enforcement agencies to access records of exactly who is talking to whom. Considering the world of telephony, we are talking about call detail records (CDRs). These records usually store information about calling and called party, starting time and duration of the call and other information such as location of a telephone device. These data carry wealthy information for government and law enforcement agencies because they can help identifying suspects or can be used as an evidence at the court. Misconduct of these data can have serious consequences, thus the access to these data is usually protected. The protection varies from country to country with some countries requiring a court order to access the data while others are less restricted. Either way, telephone service providers always store call detail records, because they are critical to the production of revenue. The reason is that call detail records are the basis for the generation of telephone bills. The telephone world has not changed too much since the beginning thus the same principles for user accounting

and monitoring are valid today.

The analogous model to CDRs for the Internet is the Internet registry of allocated addresses. Historically the end users obtained direct allocation of addresses that were recorded in the registry. The information in the registry was freely available to anyone, thus query the register was sufficient to know, who is the owner of an IP address. The commercialization of the Internet in the 1990's came with a different model. Since then, it is the ISP who obtains an address allocation and delegates it further to its end users. This has serious consequences for data retention mechanism, because, at present, it is only the ISP who can accurately say, who is the owner of an IP address. Although the IP address is used as an identifier from the data retention point of view, different technologies used in ISPs networks have different requirements for ISP to be able to fully identify a user. Unfortunately, these technologies are changing rapidly. Contrary to these difficulties, the demand for data retention is increasing in recent years, because the communication between people quickly leaves telephone networks towards the Internet. Calls and SMS messages are superseded by applications using WiFi or 3G/4G data networks such as Skype, WhatsApp, iMessages, just to name a few.

This fast growth of mobile devices in recent years together with the overall rapid growth of the Internet have a serious impact on the number of available IP addresses. With the IPv4 address exhaustion, the ISPs are forced to add support for the IPv6 protocol, which is, however, not backward compatible with the IPv4. This migration from one incompatible protocol to another is probably the second time in the history of the Internet, where the first was the transition from NCP to TCP/IP in January 1983 [4]. The incompatibility forbids the ISP to migrate to IPv6 only networks because the overall majority of content is available using IPv4 protocol only. The IPv6 protocol also changes the paradigm of address assignment for connected devices such as several IPv6 addresses per one interface, temporary addresses or different usage of DHCPv6 protocol. The ISP must monitor both protocols and cope with new features of IPv6, which is not an easy task, because there is a lack of information available, e.g., Is it possible to use similar approach for user accounting in IPv6 network as in IPv4 network?, Is it enough to just store an IPv6 address per user? How does the monitoring of dual stack infrastructure scale in large networks?

1.1 Goal of the thesis and research questions

The thesis deals with the issues of user monitoring and accounting especially in next generation networks. The definition of a next generation network or future Internet is cumbersome because the term has been used from several point of views, e.g., a complete redesign of the Internet (clean slate approach) or an evolutionary approach. The thesis defines the future network as a network, which is incompatible with the today's network architecture (TCP/IPv4). The example of a future network can be IPv6 protocol, but the thesis is not limited only to the IPv6. The aim of the thesis can be summarized in the following points:

- **Describe current approaches in user accounting.** The thesis summarizes the background information needed for understanding the development presented in later chapters. Overview of address configuration techniques will be provided, as these techniques are necessary for better understanding of user accounting.
- **What are challenges and approaches to reach the next generation network?** With the incompatible architecture and several different transition mechanisms, the

transition from the IPv4 protocol to the IPv6 protocol can be seen as an example of the transition to a next generation network. Because there is fundamentally different situation than in 1983, experience learned from the migration will be presented. Understanding of the transition between IPv4 and IPv6 protocols can help us to understand more general questions. How to run an user accounting process in a potential network that come after IPv6? How long does it take to move from one incompatible network architecture to another one? There are currently several proposals of different architectures. Some of them take benefit of IPv6 and try to extend it, some of them are again incompatible.

- **Which solution for user accounting in next generation networks can be used?** A different architecture comes with different requirements for user accounting and maintaining data retention policies. Novel approaches how to handle these different requirements will be presented. The approaches should comply with the current data retention regulations. Using transition between IPv4 and IPv6 as an example and solving the user accounting problem for this transition process, it can bring us closer to a better understanding of the transition process between IPv6 and future network architecture.
- **Are the proposed accounting techniques feasible for deployment? Are they scalable?** The approaches used to cope with user accounting in a next generation network should be deployable and scalable. As a measurement of scalability, the solutions should be tested in a reasonable big network and the thesis should present statistics to support the statement of scalability and deployability.

1.2 Structure of the thesis

The objectives of this thesis are described in the previous section. This section describes the overall structure of the thesis and individual chapters.

- Chapter 2 discusses in detail a transition of a network from one network protocol to another. IPv6 is used as an example of the future networking protocol. The chapter presents an analysis of IPv6 support in routing infrastructure, content distribution and among end users. We discuss how to measure transition progress to obtain meaningful and precise results and describe the impact of the transition to the process of user accounting. We use publicly available data and own dataset collected during IPv6 deployment at our university.
- Chapter 3 describes several issues connected with user accounting in dual stacked networks. Transition techniques between IPv4 and IPv6 protocols are discussed as well. We show which information is available for user accounting, which information is missing and how we can build a scalable system that is able to provide necessary information for user accounting process in these networks.
- Chapter 4 discusses user accounting in networks, where network address translation techniques are used. These techniques hide user identity that is necessary for accounting, backtracking security incidents or troubleshooting user's problems. We extend the system presented in chapter 3 to be able to account users even without the support of middleware that provides the translation.
- Chapter 5 concludes this thesis, describes main contributions and discusses the future work.

2

Toward Next Generation Network

"If you cannot measure it, you cannot improve it."
– William Thompson

The initial idea or the first conception of what would became the Internet was a vision of J.C.R. Licklider about Intergalactic Computer Network. The vision and ideas (online communities, software as a service, sharing resources, etc.) presented in Licklider's first papers - *Man-Computer symbiosis* and *The computer as a Communication Device* were maybe ground-breaking in the 1960s, but the current Internet is working almost the same way as he predicted 50 years ago.

Licklider's vision of a universal network provided the original intellectual push that led to the realization of the ARPANET. The ARPANET started with four interconnected devices (Interface Message Processors - IMPs) in 1969 and grew steadily to more than 200 IMPs in 1981. The addressing scheme used 8 bits split into host part (2 bits) and site number (6 bits) [5]. The 2 bits for the host part were considered as sufficient, because no one could imagine a site with more than one computer. The host part started with zero, thus, e.g., third ARPA node was numbered as 0/3. The additional research on computer networks found that Network Control Program (NCP), which provided host-to-host protocol used in the ARPANET, is inadequate to run on networks based on different underlying communication techniques (e.g., digital packet broadcast radio and satellite networks) [4].

The TCP and IP protocols were developed as protocols that could replace the NCP and the transition from NCP to TCP/IP started in 1982. This could be perceived as the first transition between two incompatible architectures. The NCP was officially obsoleted on January 1, 1983. All hosts (approximately 250) had to be TCP/IP capable till this date, or they could not connect to the network. This hard deadline is also called a flag day. IP protocol extended the 8 bits address to 32 bits where 10.0.0.0/8 was reserved for ARPANET nodes. The 32-bit address was perceived as large enough in the 1980s because no one could imagine such number of computers will ever exist.

In the early 1990s, it was obvious, that the Internet is growing much faster than anyone expected. The discussion over how the 32-bit IP address could be expanded began and took almost eight years. Steve Deering and Robert Hinden recorded the results and submitted them as RFC 1883 in December 1995 (IPv6 Protocol). At the same time, another effort how to restrain IPv4 addresses was in full swing. This effort introduced several techniques - classless interdomain routing (CIDR) that made usage of address space much more efficient, and network address translation (NAT) that allowed multiple devices to share a single public address. These techniques preserved the global IPv4 address pool till 2011 when the Internet

Assigned Numbers Authority (IANA) allocated the remaining last five `/8` address blocks of IPv4 address space to the Regional Internet Registries (RIRs). APNIC, LACNIC, RIPE NCC and ARIN already depleted their address pools. The only remaining RIR is AFRINIC, where the projected exhaustion date is in 2019. Thus, we are facing the second transition from one incompatible protocol to another. The situation is however very different from the previous one (NCP to TCP/IP).

The number of networks, devices and connected users using IPv4 protocol is tremendously big. Also the facts, that there are still IPv4 addresses available in several regions together with heavy usage of NAT technique and lack of features parity between IPv4 and IPv6 across software and hardware devices hold back the transition from IPv4 to IPv6 even more. The consequence is that every network device and network supports IPv4 protocol in these days, but the same is not true for IPv6. There are also additional costs connected with the transition. Firstly, the support staff needs to be trained for IPv6 protocol, secondly, IPv6 support is required across the whole network – hosts, routing system, switches, routers and every service, such as DNS, HTTP, mail servers, firewalls or security and management systems must have IPv6 support enabled. These additional costs and the fact that everything works right now with IPv4 lead to a situation that network operators do not deploy IPv6 and play a waiting game. This situation was well described in the Geoff Huston's paper *Is the Transition to IPv6 a „Market Failure?“* [6]. The problem is also described as *chicken-and-egg* problem. Network operators are not interested in deploying IPv6 for end users because there is a lack of content available over IPv6. Content providers are not interested in deploying IPv6 as well because there are not many end users to view their web pages over IPv6. The slow deployment of IPv6 and the fact that dual stack (network nodes run both protocols simultaneously) is perceived as a next step in the transition from IPv4 to IPv6 mean that there is a continual demand for IPv4 addresses. Traditional way how to fulfill this continual demand for IPv4 addresses is address sharing using NAT. This is however not the only way, how ISPs are stretching out the life of IPv4. Transfer markets have emerged as another mechanism to acquire IPv4 address space. RIRs allowed intra-registry transfers of IPv4 addresses between organization as a reaction to the demand. Livedariu et al.'s paper *A First Look at IPv4 Transfer Markets* pointed out the fact that about half of the autonomous systems (AS) receiving IPv4 transfers have not deployed IPv6. This indicates, that some organizations (especially newly joining edge ASes) view the IPv4 transfer market as a mechanism to avoid deploying IPv6 immediately [7]. IPv6 deployment could be attractive for these organizations probably only if there is enough content available over IPv6.

These two examples of transitions use different approaches for switching from an old to a new architecture. Dual stack hosts and relays between NCP and TCP/IP architectures were used in the NCP to TCP/IP transition. The transition also had a hard deadline and all nodes had to be upgraded till that date because the previous protocol was switched off. The transition from IPv4 to IPv6 also uses dual stacked hosts as a transition technique. However, there are many more different transition technologies and there are also several techniques how to prolong the operation of IPv4 architecture. There was also an attempt to set a transition plan in RFC 5211 [8]; unfortunately, it failed. There are simply too many nodes, protocols and different technologies in the current Internet that it is impossible to convert all of them till some specific deadline. The consequence is that the IPv6 transition is going much slower than it was expected. The another observation is that a flag day is not a viable option in such a large network as the current Internet.

Furthermore, every network architecture that has been proposed so far uses clean slate

design. The clean slate is an approach, where everything is designed from scratch without maintaining compatibility with the previous architecture. It was the case of TCP/IP architecture, which was designed as incompatible with NCP. The same approach was also used with IPv6. Recursive InterNetwork Architecture (RINA), Content Centric Networking (CCN), Named Data Networking (NDN) are other examples of new networking architectures that are currently proposed. All these new architectures are, however, incompatible with IPv4/IPv6. We can expect that these architectures will face the same issues with migration as IPv6. Fortunately, we can learn much from the current IPv4 – IPv6 transition. This transition is extensively monitored since the beginning, and the data are freely available. The analysis of these data can help us to discover main issues so the further architectures can avoid them.

ISPs or content providers ask in essence the same following questions when they are thinking about migration to a new architecture.

- What is the actual number of content available over a new architecture?
- How is the availability of the content measured? Which dataset is used?
- If we deploy a new architecture, how many of our users will support it? How much traffic will flow over it? How much it will cost?
- How many users are using the new architecture?

The priority of these questions depends on the type of the network. The ISP is more interesting in the first three; the last question is more interesting for content providers. The rest of this chapter tries to answer these questions using historical data, statistics and real experience from the transition of Brno University of Technology (BUT) network. A part of this chapter was submitted and presented at International Conference on Networking and Services [9] and International Conference on Network Protocols [10] conferences.

2.1 State of the Art

There are several approaches to measuring IPv6 adoption and quality of IPv6 service. Measurement can be performed on the content provider's side. The methodology is typically based on inserting a small invisible image [11] or JavaScript fragment [12] into the content of content provider's web page. Client's browser executes the code or tries to download the image using IPv4, IPv6 or other transition technology (6to4, Teredo). The analysis of requests can reveal the number of clients that can or cannot connect to dual stack Web servers and their latency. This methodology measures clients. It shows, if IPv6 is supported by an application (web browser), operating system and client's ISP (Internet Service Provider). The number of „consumers“ is essential for content providers because without IPv6 active customers they will not invest their time and money to IPv6 transition. The numbers are between `9 - 11 %` in July 2016, as it is shown in Google's global IPv6 statistics[1]. There are however countries with much higher IPv6 penetration – currently Belgium, USA, Switzerland or Greece and Portugal as shown in Geoff Huston's [13] and Cisco's statistics[2]. There is also a new approach started recently by Geoff Huston. He measures IPv6 and DNSSEC

[1] `https://google.com/ipv6`
[2] http://6lab.cisco.com/stats/

adoption using embedded code in Google's online advertisements system [14]. This methodology provides very good results as the ads are spread over the world because of globally deployed Google online advertisement network.

Another method is based on measuring the number of autonomous systems announcing an IPv6 prefix. The statistic informs how ISPs and transition networks are prepared to provide IPv6 connectivity for customers. One analysis was presented by Karpilovsky et al. [15]. Their study has shown, that almost half of assigned IPv6 prefixes is not used at all, and the rest of them is announced long after the allocation. The drawback of the methodology is that the presence of an ISP's IPv6 prefix in the global BGP table does not really mean its availability for ISP's customers. Despite that, the number of IPv6 prefixes in the global routing table is increasing and can be seen as a part of IPv6 transition progress. Thorough analysis of global routing table, the number of IPv4/IPv6 ASNs, prefixes, etc., is done by Geoff Huston and these data are available on his website [16]. These data will be analyzed in detail in the next section.

One way of measuring the quality of IPv6 service is to measure the one-way delay. Zhou et al. [17, 18] published a study comparing IPv4 and IPv6 one-way delay between several measurement points. They analyzed the RIPE IPv6 data, which include traceroute, delay and loss measurements among a list of IPv6 sites since 2003. Their conclusion was that native IPv6 paths had small 2.5 percentile and median end-to-end delay, and comparable delay to their IPv4 counterparts. The study presented by Arthur Berger [19] found that the latency is less over IPv4 than IPv6. He showed that the mean latency is 55 ms over IPv4 for destinations in the North America but substantially higher, 101 ms, for the same destinations over IPv6. The difference between the IPv4 – IPv6 performance is more likely correlated with a different forward AS-level path as was reported by Amogh Dhamdhere et al. [20]. The measurement by Mehdi Nikkhah et al. [21] compares the performance of IPv6 and IPv4 protocol by measuring the web page download time. They found that control plane (routing) was responsible for differences between IPv4 and IPv6, because the data plane (implementations of IPv4 and IPv6 stacks) performs comparably.

The methods described above provide information about IPv6 prefix allocation, number of clients, or test the path delay. There is also an approach presented by Jakub Czyz et al. [22] who try to analyze the IPv6 adoption from several perspectives – allocation of IPv6 prefixes, clients readiness, etc. These different metrics give entirely different insight into the adoption of IPv6, and show orders of different magnitude progress. For instance, 12% of cumulative allocated prefixes are IPv6, but just 0.63% of average traffic is carried over IPv6 – a two-order-of-magnitude difference. This difference follows the prerequisites for IPv6 deployment (e.g., allocation precedes routing, which precedes clients, which precedes actual traffic).

All above-described techniques present IPv6 deployment from the Internet infrastructure point of view. However, the ISPs are also interested in the number of content providers that enabled IPv6 for their services. The amount of available IPv6 content also indicates how much IPv6 traffic can service providers expect in case they decide to deploy IPv6. The next important information is the quantity of sites reachable from IPv6 only networks. Several measurements have been published to describe this information – [23, 24, 25, 26, 27, 28]. These papers and methodologies are analyzed in more detail in section 2.2.2.

The following section describes the IPv6 transition progress from several points of view, e.g., global routing or IPv6 content penetration. The measuring platform for gathering long-term statistics about IPv6 penetration will be described in the next section as well.

2.2 Measuring the IPv6 transition progress

The previous section gave an overview of approaches used for measuring IPv6 transition progress. This section describes two of these approaches in detail. Firstly, BGP and IPv6 prefix analysis are presented. We examine global BGP table to find out current trends in IPv6 adoption. We also correlate BGP analysis with NetFlow data from Brno University of Technology (BUT) and CESNET networks. It is a novel approach, as BGP analysis is usually presented without any correlation with real traffic flows. We believe the correlation can help with understanding of IPv6 deployment status.

Secondly, IPv6 penetration among content providers will be evaluated in detail. Several projects measuring IPv6 penetration from a content provider point of view were described in the State of the Art section. However, these projects often use only a subset of available data. The consequence is that IPv6 penetration reported by these projects is usually overestimated. We highlight these issues and describe a methodology and analysis that overcome their limitations.

2.2.1 BGP and IPv6 prefix analysis

The IPv6 allocation process maintains the same allocation hierarchy as with IPv4 addresses. IANA allocates IPv6 address blocks to the five regional Internet registries (RIRs). The RIRs allocate the IPv6 addresses to various local registries (LIRs) which allocates the addresses to their customers and end users. RIRs publish a daily snapshot of allocated blocks of addresses which can be downloaded for further analysis. The first assessment of IPv6 deployment could be an analysis of these allocations. However, there could be a bias because each RIR has a different allocation policy. For example, RIPE NCC address policy states that IPv4 allocations will only be made to LIRs if they have already received an IPv6 allocation from an upstream LIR or the RIPE NCC. The consequence is, that LIRs allocate both IPv4 and IPv6, but use only IPv4 address space. This behavior of LIRs is also confirmed by research community [15].

Therefore, it is better to analyze the number of IPv6 prefixes found in the Internet's global routing table rather than just allocations. Unfortunately, even this approach can have a bias. The presence of an IPv6 prefix in BGP table does not mean that the prefix is used in a production network. ISPs often announce a prefix before the prefix is put in production. Twitter, Inc. can be used as an example of this behavior. Twitter, Inc. is using ASN 13414, which originates three IPv6 prefixes since the beginning of 2015[3] without IPv6 enabled on their websites or in mobile applications.

Dataset

As a dataset, we use BGP data collected by Route Views project [29] since January 2004. The calculation of statistics is ongoing, and data are regularly updated every day. NetFlow data from Brno University of Technology (BUT) and CESNET networks are used to find out how many IPv6 prefixes are used, how many users use IPv6 and which content is accessed over IPv6, etc. The BUT network has approximately 25 000 users (students and staff). According to statistics from Akamai [30], the BUT network was in the second place worldwide among universities in the IPv6 penetration. CESNET is National research and education network in the Czech Republic. The overall capacity of the CESNET network

[3] https://stat.ripe.net/widget/announced-prefixes#w.resource=13414

Table 2.1: Flow data statistics obtained on 13th January 2015 in Brno University of Technology and CESNET NREN network

	BUT	CESNET
Average Bit Rate	2.127 Gb/s	18.4 Gb/s
Maximum Bit Rate (peak)	3.604 Gb/s	32.3 Gb/s
Total Number of Flows	608 000 000	8 062 000 000
Average Flow Rate	7 000 flow/s	93 000 flow/s
Maximum Flow Rate (peak)	10 000 flow/s	152 000 flow/s
Consumed Disk Space (compressed)	25 GB	240.3 GB
Consumed Disk Space (uncompressed)	48 GB	482 GB

is 1.478 Tbps. The NetFlow data are collected on BUT/CESNET backbone links. An example of data stored during a day is shown in Table 2.1.

Before we analyze these data, it is important to understand possible biases. Routing table view biases are well described in Jakub Czyz et al. paper *Measuring IPv6 Adoption* [22]. Firstly, BGP dumps are collected from a finite set of collectors, whose global distribution is not uniform. Secondly, the dumps provide only an incomplete view of each AS's connections and cannot see, e.g., a peer-to-peer connectivity between smaller ISPs, since these routes are never propagated. These biases are a limitation of the BGP data dumps and stem from the nature of the BGP protocol. Even though the BGP dataset is not perfect, we believe that the dataset still can be used to provide useful information about trends in IPv6 adoption.

The NetFlow dataset can also have a bias. The data are collected only in the Czech Republic and because the Czech Republic is not an English speaking country, users will probably consume mainly regional content. The BUT dataset provides a view from edge network only; thus the dataset will only show information about what is important for BUT users only. However, we believe that the dataset can still provide valuable information for the following reasons: *(i)* the number of users in BUT network is reasonable big; *(ii)* there are a lot of international students and researchers which increase a demand for international content; *(iii)* whole BUT student network provides native IPv6 connection; *(iv)* the BUT NetFlow dataset is collected without any sampling.

Therefore, we will use NetFlow and BGP datasets for analyzing trends in IPv6 deployment knowing well their limitation.

Autonomous system analysis

Within the Internet, an autonomous system (AS) is a single entity, typically an Internet service provider or a large organization. A unique Autonomous System Number (ASN) is allocated to each AS for usage in BGP routing. The ASN allocation process is similar to allocation of IP prefix. IANA allocates blocks of AS numbers to RIRs. Each RIR then assigns AS numbers to entities within its region. Autonomous System number space use 16-bit field (possible 65,536 unique values). The size, however, became a problem, because RIRs were running out of the 16-bit ASN numbers. RFC 4893 [31] introduced support for four-octet (32-bit) AS number to overcome the problem. RFC 4893 was written to change the BGP protocol as little as possible. It introduces a new capability and two new optional transitive attributes `AS4_PATH` and `AS4_AGGREGATOR`. The compatibility for the 32-bit extension is negotiated during BGP session establishment. Both implementations

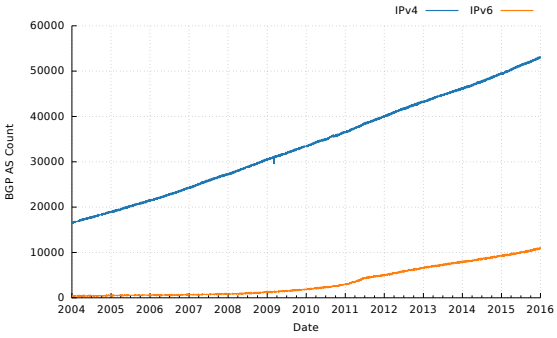


Figure 2.1: Number of unique ASes supporting IPv4 or IPv6

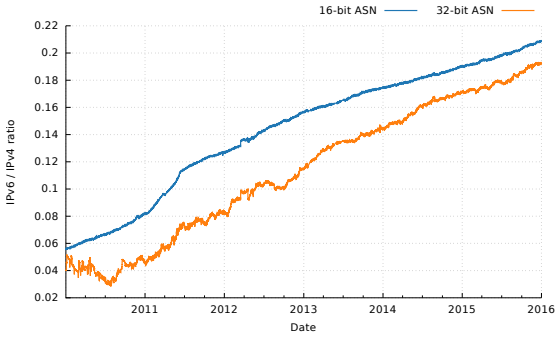


Figure 2.2: Ratio of IPv6 support among 16-bit and 32-bit ASes

(32-bit and 16-bit) can operate together thus allowing gentle transition and incremental deployment.

It is interesting to see that the transition from ASN-16bit to ASN-32bit was driven by the same reason as the transition from IPv4 to IPv6; to have a larger number space. Contrary to transition from IPv4 to IPv6, the enlarged ASN space was designed to be backward compatible. The consequence is that ISPs can freely move to the new, larger AS Number space without any serious issues.

The global BGP table contained around 53 000 unique ASNs at the end of 2015. Figure 2.1 shows a number of unique ASes supporting IPv4 or IPv6 protocol. Although a few IPv6-only[4] ASes exist in the global BGP table, these ASes are tied with entities that have IPv4 ASN as well. The consequence is that there are not any entities (ISPs or large organizations) supporting IPv6 protocol only. Thus, the IPv6 line in Figure 2.1 should be seen as ASes that support both protocols (IPv4 together with IPv6).

The overall number of unique ASes depicted in Figure 2.1 can be split to 16-bit and 32-bit numbers. These numbers can indicate the support for IPv6 protocol among new companies because the default ASN allocation is currently a 32-bit ASN.

The ratio of IPv6 support among 16-bit and 32-bit ASes is shown in Figure 2.2. The ratio is computed according to Equation 2.1. The same formula is used to compute ratio among 32-bit ASes.

$$ratio16bit = \frac{number\ of\ 16\text{-}bit\ ASNs\ supporting\ IPv6}{total\ number\ of\ 16\text{-}bit\ ASNs} \tag{2.1}$$

Figure 2.2 shows that approximately 21% of 16-bit ASes and 19% of 32-bit ASes support IPv6. The conclusion can be that support for IPv6 is slightly lower among new companies. It is interesting because new companies can usually request only a small amount of IPv4 addresses (`/22` = 1024 addresses in case of RIPE region) due to depletion of IPv4 address pool, but even the small number of assigned addresses is not a reason to deploy IPv6 for them. These results support the conclusion of Livedariu et al. work [7] that some organizations (especially newly joining edge ASes) using IPv4 transfer market as a mechanism to avoid deploying IPv6 immediately.

[4] AS that originate IPv6 prefix only

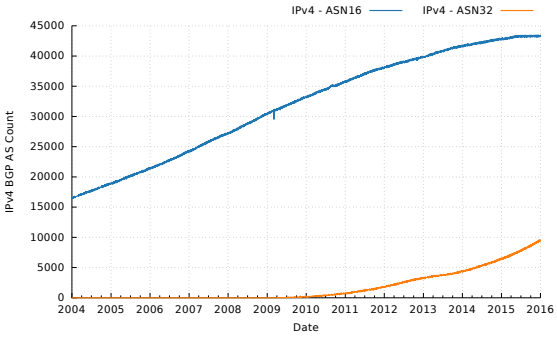


Figure 2.3: 16-bit ASN and 32-bit ASN supporting IPv4

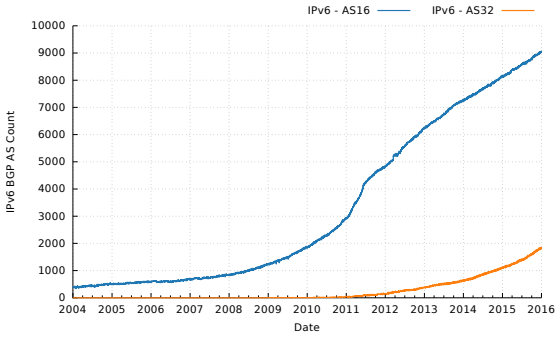


Figure 2.4: 16-bit ASN and 32-bit ASN supporting IPv6

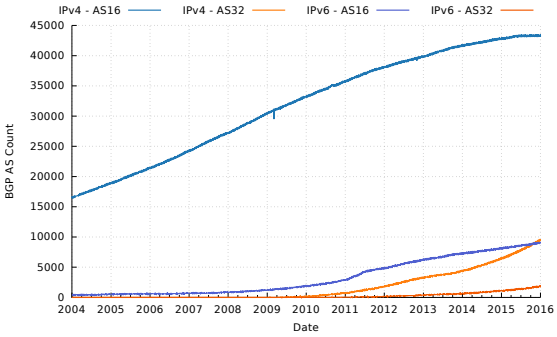


Figure 2.5: 16-bit and 32-bit IPv4 and IPv6 autonomous systems in BGP

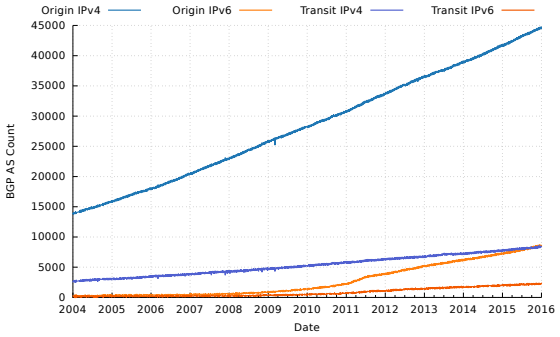


Figure 2.6: Origin and mixed autonomous systems supporting IPv4 or IPv6

Figures 2.3 and 2.4 show number of 16-bit and 32-bit IPv4 and IPv6 ASes. To put the growth of IPv6 in the context, Figure 2.5 compares these numbers with each other. These figures indicate that support for IPv6 is growing, but very steady.

The overall number of autonomous systems presented in previous figures can be further divided between origin only, mixed or transit ASes. Origin only autonomous systems are systems that only originate an IP prefix and appears at the end of an AS path. It means that origin only autonomous system does not allow traffic from other networks to go through itself (does not provide any transit service to anybody). A mixed autonomous system is a system that announces an IP prefix and it also appears in the middle of AS paths. Thus, it allows passing the traffic through itself to other networks and provides transit service to someone else. Transit only autonomous systems are systems, which provide transit service only and do not originate any IP prefix.

By dividing the overall number of autonomous systems between origin, mixed and transit only, we can observe IPv6 adoption between edge networks (origin only ASes) and the core of the Internet (mixed and transit only ASes). Figure 2.6 shows IPv6 adoption between origin only and mixed ASes. Figure 2.7 depicts IPv6 adoption between transit only ASes and finally, Figure 2.8 shows the IPv6/IPv4 ratio among these autonomous systems. These Figures show that at the end of 2015 IPv6 is supported in 19% origin ASes, 26.1% mixed ASes and 59% transit only ASes. The IPv6 penetration is thus higher in the core of the network and lower at edges. These numbers follow the general prerequisites for IPv6 deployment where the IPv6 supports must be enabled in the core first to allow interconnection

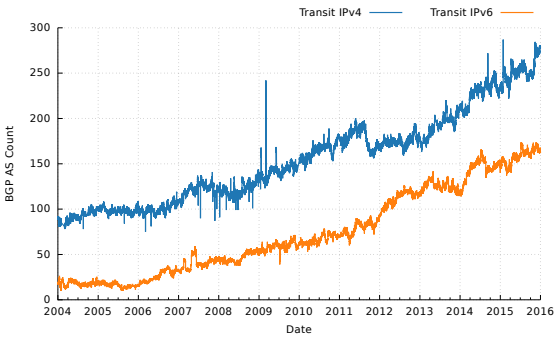


Figure 2.7: Transit only autonomous systems supporting IPv4 or IPv6

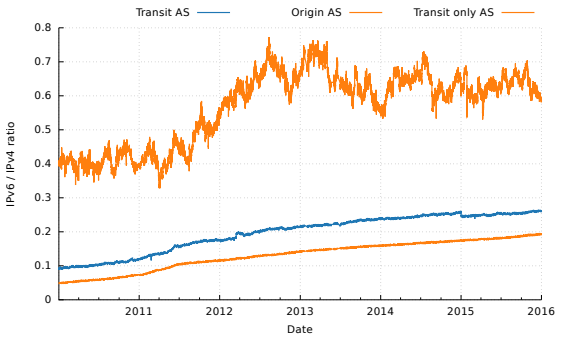


Figure 2.8: Ratio of IPv6 support among transit only, mixed and origin ASes

Table 2.2: Increment in IPv4 and IPv6 ASes over the 2010 - 2015 period

	Origin AS		**Transit AS**	
Date	IPv4	IPv6	IPv4	IPv6
2009	2500 (9.6%)	500 (55.3%)	500 (10.7%)	100 (42.5%)
2010	2500 (9.1%)	900 (62.9%)	600 (10.7%)	200 (45.1%)
2011	3000 (9.7%)	1600 (71.8%)	500 (8.6%)	400 (58.9%)
2012	2800 (8.2%)	1300 (33.3%)	400 (6.4%)	300 (30.3%)
2013	2400 (6.6%)	1000 (19.6%)	500 (7.3%)	300 (19.5%)
2014	2750 (7.1%)	1100 (17.5%)	550 (7.6%)	200 (6.9%)
2015	2900 (6.9%)	1300 (18.3%)	650 (8.3%)	430 (23.5%)

of edge networks.

Table 2.2 analyzes IPv6 adoption in more detail by comparing the increment in IPv4 and IPv6 ASes over the 2009-2015 period. What this table indicates is that the growth of IPv4 ASes (new companies) is stable in last six years. IPv6 ASes, on the other hand, raise faster in percentage, but the actual growth in number is still low. For example, there were about 2900 new IPv4 origin ASes but only 1300 new IPv6 ASes over 2014 - 2015 period. The highest growth of IPv6 ASes in 2012 was caused by World IPv6 Launch on 6 June 2012 – an event that promoted the IPv6 deployment among network operators and content providers. You may notice a drop in IPv6 transit ASes growth in 2014 – 2015 period. We detected only 200 new IPv6 transit ASes in global BGP table. We are not exactly sure what caused this drop, but it was the period when IPv4 addresses were still available in ARIN region. This was not the case in the next year where we can see that the growth returned to 23.5%.

Figures 2.6, 2.7 and 2.8 show a rising trend in IPv6 adoption. However, we find the trend still quite slow. Despite the fact that IPv6 has been deployed among several biggest companies – Facebook, Google, Comcast, Verizon just to name a few, it appears, that other providers are still not convinced, that deploying IPv6 is a viable option for them. This growth rate among autonomous systems should be much higher considering the fact, that IPv4 address pools are depleted in all regions except AFRINIC.

IP prefix analysis

The allocation process of IP prefixes is similar to allocation process of autonomous system numbers described above. Central coordination is required to ensure that different networks use unique non-overlapping IP prefixes. IP addresses are allocated by the IANA from pools of unused address space and delegated to the appropriate RIRs. Each RIR then assigns IP prefixes to entities within its region. These allocated prefixes can be in any of five states:

- Reserved for special use
- Part of the IANA unallocated address pool
- Allocated to a RIR but unassigned to any entity
- Assigned to entity, but not advertised in BGP
- Assigned and advertised in BGP

At the beginning of 2016, the status of IPv4 address pools is following. IANA address pool was exhausted in February 2011 followed by APNIC in April 2011, RIPE NCC in September 2012, LACNIC in June 2014 and ARIN in September 2015. Projected exhaustion for AFRINIC is at the beginning of 2019. In this case, the definition of exhaustion is following: IPv4 address pool of available addresses reaches the threshold of no more general use allocations of IPv4 addresses. Each RIR defines the threshold differently. For example, APNIC and RIPE NCC set the threshold to the last `/8` of available IPv4 addresses in their address pool. When their address pools reached the last `/8` (they had less than 16,777,216 of available IPv4 addresses) usually only newcomers can ask for IPv4 allocation. ARIN, LACNIC and AFRINIC have different policies - ARIN sets the threshold to `/10`, LACNIC and AFRINIC to `/11`. The consequence is that almost all IPv4 addresses are allocated to end entities. Figure 2.9 confirms this situation by showing the number of IPv4 addresses assigned, allocated and advertised for each RIR. We can see that RIRs have depleted their IPv4 address pools because a majority of IPv4 addresses are in the assigned state. One exception is AFRINIC with approximately 2.71 `/8` available IPv4 addresses in its pool. Despite the fact that RIRs pools were depleted, there is still a small room for growth – especially in ARIN region, because a lot of IPv4 addresses are not even advertised in the routing system. More precisely, around 25% of overall IPv4 addresses (about 1 billion of IPv4 addresses) has not been advertised in BGP yet. Thus, these addresses are not used for end-to-end connectivity.

Figure 2.10 shows the overall number of IPv4 and IPv6 prefixes in BGP table over the 2004 – 2014 period[5]. The figure shows similar trend line as we could see in the autonomous system number analysis – the number of IPv6 prefixes is growing. The growth is, however, steady. According to Geoff Huston [32], the rate of IPv6 growth has increased to the current level of some 15 to 20 new entries per day. It is still, however, lower than the IPv4 growth, which is growing by some 100 - 150 prefixes per day even today, where all main pools are depleted.

Based on routing table snapshots, we obtained the number of IPv6 prefixes announced in the global routing table as shown in Figure 2.10. The number of IPv6 prefixes is much lower comparing to the number of IPv4 prefixes in DFZ (Default-free Zone). It is obvious

[5]The drops in prefix count in the figure is caused by drops in number of routeview's BGP neighbors. These are temporary issues.

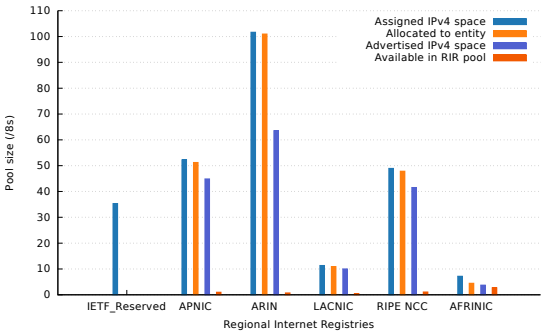


Figure 2.9: Regional Internet Registries IPv4 pool size on 17.2.2015

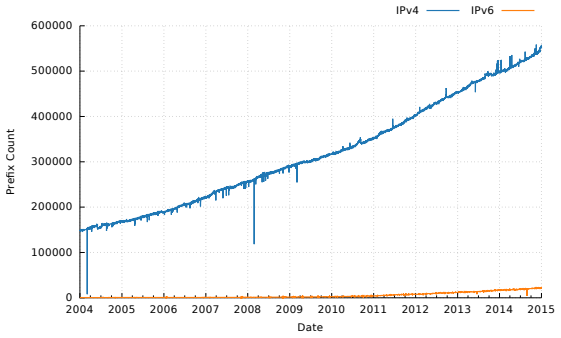


Figure 2.10: IPv4 and IPv6 prefixes in BGP table over 2004 - 2014 period

that IPv6 protocol has not been deployed widely yet. However, the overall number of IPv6 prefixes could be lower in the future as well. IPv6 prefixes allocated to ISPs and end users are much larger than IPv4 prefixes. Depending on RIR policy, IPv6 allocation is typically a `/32` IPv6 prefix (`/29` can be assigned to a LIR without any justification from the LIR about the network size or number of users), and the longest route allowed and accepted by DFZ is currently `/48`. The consequence is that IPv6 address space is allocated in more continuous blocks, thus, an autonomous system usually has much lower number of IPv6 prefixes. The conclusion can be, that the number of IPv6 prefixes in the global routing table will be smaller than the number of IPv4 prefixes, however, this conclusion is with a high degree of uncertainty as traffic engineering or different RIR policies can change it in the future.

Unfortunately, presence of IPv6 prefix in the global routing table does not say anything about the actual usage of the prefix in a production network. Can we obtain the information? Is it possible to find if an IPv6 prefix is used or if it is just advertised in the BGP table? We used the following approach to try to find the answer.

An organization usually obtains IPv6 prefix of `/32` or `/48` length from a RIR. If IPv6 is deployed in the organization, the actual IPv6 prefix size used for addressing end devices is, however, larger. The IPv6 Addressing Architecture described in RFC 4291 [33] requires a unique `/64` prefix for every individual network segment. This requirement is necessary because all unicast addresses should use a 64-bit Interface Identifier. There are some exceptions, e.g., point-to-point links where prefix size can be `/127`, but these exceptions are not important. The consequence of using IPv6 addressing architecture is that a company usually uses many `/64` networks to logically separate its physical network topology. To illustrate this approach, let us use the BUT network as an example.

BUT obtained IPv6 PI prefix `2001:67C:1220::/46` from RIPE NCC. This prefix is used for addressing end users, servers, VoIP phones, etc. Currently, there are approximately 150 `/64` networks in BUT internal OSPFv3 network as every server farm or campus VLAN use their own prefix. The conclusion is that a `/48` prefix should contain several `/64` prefixes if IPv6 is deployed in production. Of course, the global BGP table does not contain prefixes of `/64` size, because these prefixes are aggregated on ASNs boundaries. However, we can distinguish unique `/64` prefixes from the network traffic. NetFlow data collected on BUT and CESNET networks described in dataset section 2.2.1 can be used for this purpose. We can aggregate IPv6 addresses on unique `/64` and `/48` boundaries. If there are more `/64` prefixes per one `/48` prefix, we can conclude, that the network deployed IPv6 in production.

Table 2.3: Unique /48 and /64 IPv6 prefixes seen by BUT network on 9.1.2015

	ASN	Prefix	/48	/64
Brno University of Technology	197451	2001:67c:1220::/46	3	99
Facebook, Inc.	32934	2a03:2880::/32	39	580
O2 Czech Republic, a.s.	5610	2a00:1028::/32	116	1545
Google, Inc.	15169	2a00:1450::/32	20	266
CESNET z.s.p.o.	2852	2001:718::/32	41	199

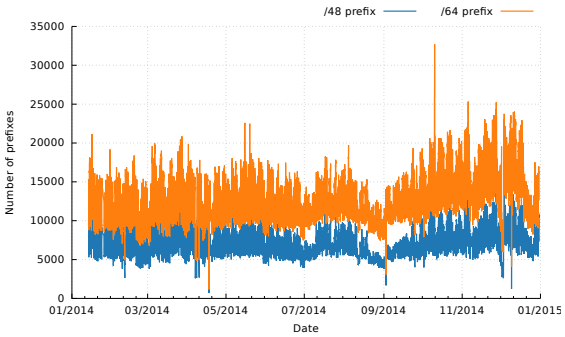


Figure 2.11: Number of unique /64 and /48 prefixes seen in CESNET NREN network

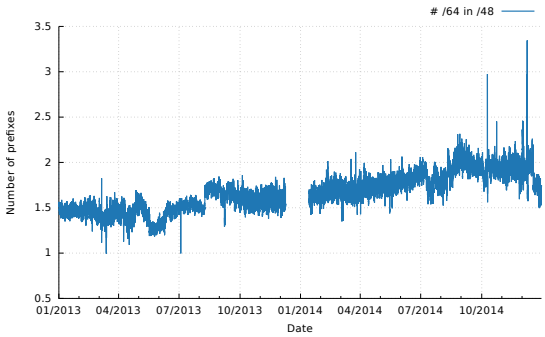


Figure 2.12: Number of unique /64 prefixes in one /48 prefix

If there are only 1 or 2 /64 prefixes per a /48 prefix, we can conclude, that the /48 network is probably still in a testing phase.

Is this hypothesis right? Table 2.3 shows statistics obtained from NetFlow data on BUT network. We probably do not have to introduce Facebook or Google - big players in IPv6 deployment. The rest are CESNET – the biggest NREN network in the Czech Republic and O2, one of the biggest ISP in the Czech Republic. Both companies are known for promoting and deploying IPv6 in production networks in the Czech Republic. The table supports the hypothesis that a production network has several /64 per /48 – there are approximately ten times more /64 in a /48 in these networks.

We can also analyze the long-term data from CESNET and BUT datasets. CESNET NetFlow dataset was analyzed, and the results are depicted in Figures 2.11 and 2.12. Figure 2.11 shows the number of unique /64 and /48 prefixes in CESNET NREN network in 2014. The figure shows, that the number of these prefixes steadily increases. The number of unique /64 prefixes in one /48 prefix is depicted in Figure 2.12. The figure shows that there are around 1.5 unique /64 prefixes in one /48 prefix in 2013 and around two unique /64 prefixes in 2014.

BUT dataset is analyzed in Figures 2.13 and 2.14. These figures show a similar trend as figures analyzing CESNET dataset. The number of /64 prefixes in one /48 prefix is lower, but this is obvious, because BUT network is not as big as CESNET network.

We should, however, highlight possible biases that these data could have.

1. If the end network belongs to a content provider, traffic from BUT network could be routed just to the nearest datacenter, thus it could be seen that we are accessing always the same IPv6 network. For example, if a content provider uses IPv6 prefix 2001:db8::/48 and the nearest datacenter uses IPv6 prefix 2001:db8:0:aaaa::/64

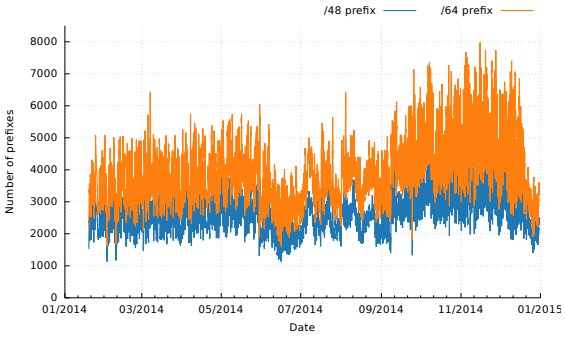


Figure 2.13: Number of unique `/64` and `/48` prefixes seen in BUT network

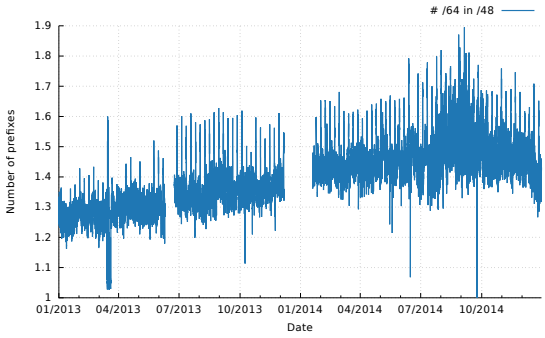


Figure 2.14: Number of unique `/64` prefixes in one `/48` prefix

it would appear that there is only one `/64` prefix in `/48` prefix because whole BUT traffic is routed to the nearest datacenter. Figures 2.11 and 2.13 show however similar trend so this should not be a problem. The bias should be also eliminated by using the dataset from CESNET NREN network since its network spans through whole country.

2. Network administrator can use only one `/64` prefix for the whole network as this prefix provides enough IPv6 addresses for the whole current Internet. It should not happen in the real world as there is usually a need for network separation.

3. The traffic profile is different in every country as each country has different popular services. For example, a huge amount of eyeball traffic comes from NetFlix, Google, FaceBook, Hulu or Amazon in the US, but the same is not true in the Czech Republic as some services are not available in the Czech Republic. Different traffic profiles can be responsible for a different amount of IPv6 traffic, but it should not cause differences in the number of `/64` prefixes in one `/48` prefix as network administrators tend to use similar network deployment practices.

The above mentioned limitations have no significant impact on the hypothesis that networks where IPv6 protocol was deployed have several unique `/64` prefixes in one `/48`. Bias introduced in the first item is eliminated by using significantly large network dataset. The second item should not happen in the real world as it denies best practices in network design. The third item can have an impact on the volume of IPv6 traffic as there could be more services available over IPv6 in different countries, but different traffic profile has not the impact on the number of IPv6 prefixes.

Even though that analysis of BGP DFZ shows the increasing trend, correlation with the real world IPv6 traffic suggest, that these prefixes are probably still used mainly for testing and not for the real traffic. We discussed this hypothesis with David Plonka, the researcher from Akamai Technologies, and he suggested that there could be another bias we have not mentioned. An ISP can use `/48` prefix per subscriber. In that case, there is only a few `/64` prefixes in the `/48` prefix as an end user does not separate home network to more then one or two logical networks (e.g., private and guest network). Furthermore, there were several discussions in NANOG and RIPE mailing lists that the model, `/48` per subscriber, is used by some ISPs in Germany or Japan. However, if a subscriber receives `/48` prefix, there should be several `/48` prefixes in one `/32` prefix. We tried to run an analysis to find out if

the bias has an impact on the presented hypothesis, but the analysis does not reveal any significant impact. However, we are going to study this bias further in the future work as the analysis was run only on a subset of our dataset.

2.2.2 IPv6 content analysis

There were several concerns among content providers about the stability of IPv6 connectivity before 2011. IPv6 traffic used several transition techniques (6to4, Teredo, ISATAP etc.) at that time and it was reported, that reliability of these techniques is low [34, 35]. A dual stack client could experience significant connection delay compared to an IPv4-only client if the path between the client and the server worked for IPv4 but was not working for IPv6. The delay was caused by the fact that a client tried IPv6 connection first and if the connection failed, it took several seconds to switch the connection back to IPv4. This problem was described already in 1994 during the discussion of requirements for IPng protocol [36].

Content providers observed that enabling IPv6 on their websites caused that a small percentage of end users had this slow or otherwise impaired access to the given website compared to IPv4. Several approaches were used to limit the bad IPv6 experience. The first approach tried to solve the problem from the content provider side using a technique called DNS whitelisting. Content providers used the technique to serve AAAA resource records (RR) to DNS recursive resolvers on the whitelist while returning no AAAA RRs to DNS resolvers that are not on the whitelist. This approach ensured that only networks without broken IPv6 connectivity would be served with AAAA RR, thus, eliminating problems with a broken IPv6 path between clients and the content provider's server. There were, however, several objections and criticism against this concept. Mainly, DNS whitelisting behavior is very different from the current practice and it may create a two-tiered Internet. There were worries that policies concerning whitelisting and de-whitelisting would be opaque and that DNS whitelisting would reduce interest in the deployment of IPv6 [37]. Today, the technique is still used, but the behavior has changed. Content providers serve AAAA records by default for most of the users but for some users AAAA records are filtered. For example, Google performs measurements of IPv6 connectivity and latency for users visiting Google's dual stack sites. The Google DNS servers do not return AAAA records to DNS resolvers if their measurements indicate that users using these resolvers have substantially worse HTTP/HTTPS access times over IPv6 than over IPv4.

Another approach emerged later – Happy Eyeballs[6] described in RFC 6555 [38]. This approach shifts the responsibility for picking a working protocol to an end device. The basic idea of this approach is the following. A client tries to connect to a server using both protocols simultaneously. If IPv6 path is broken, IPv4 connection wins. If both paths are working, the IPv6 is usually preferred. The Happy Eyball algorithm is usually implemented in HTTP browsers only. The consequence is that other than HTTP traffic is still broken.

To overcome this limitation, several operating systems use similar approach trying to find the working protocol for all outgoing connections. For example Windows 8 attempts to connect to the IPv6 enabled URL. Once it determines it can connect to that URL with IPv6, it updates the prefix policy table to indicate that IPv6 is preferred. Otherwise, IPv4 protocol will be preferred for an outgoing connection. Happy Eyeballs approach ensures a good user experience but it causes problems for a network administrator to troubleshoot and diagnose connection issues with a particular address family, since the traffic flow from

[6] the term „eyeball“ describes endpoints that represent human Internet end-users, as opposed to servers.

a user to a server can use different protocol family for different connections. The client's behavior is more dynamic and can cause problems in some cases. For example, if a user tries to reach a server behind a load balancer, every client's attempt can be forwarded by the load balancer to a different server. If the server maintains a stateful session (e.g., holding information if the user is successfully logged in), Happy Eyeball breaks the stateful session thus creates random log off events for the client [39].

To promote the deployment of IPv6, a significant effort was made by several operators and content providers. The IPv6 Day on June 8th, 2011 and IPv6 Launch on June 6th, 2012 were events that should motivate the activity of other service and content providers to enable IPv6. The IPv6 Day was considered as a testing day on which mainly the content providers enabled IPv6 for 24 hours. Such a short period was intentionally used as there were serious concerns about IPv6 stability for end users as we mentioned before. The event should test these problems. Furthermore, content providers were interested in the number of clients and volume of traffic that would use IPv6 protocol. Observation confirmed an increase of IPv6 traffic and the number of users. Google kept IPv6 enabled for several YouTube servers, thus the main contributor for IPv6 traffic since then is YouTube as Sarrar et al. [40] showed in their report, and our observation confirms their results. Thanks to the results from IPv6 Day, big content providers decided to turn on IPv6 permanently one year later on "IPv6 Launch day".

Nevertheless, big content providers such as Google, Akamai, Facebook or Netflix do not represent the whole Internet. Millions of other websites remain without IPv6 addresses. This situation raises several interesting questions.

i What is the ratio of enabled IPv6 websites and services (e.g., mail servers, name servers)?

ii Is the ratio increasing or stagnant?

iii If IPv6 is enabled, is the quality of service (e.g., response time) better, worse or the same as in IPv4?

iv If we deploy a new architecture, how many users will use it? How much traffic will flow over it?

These questions are important to ask, since answers to these questions can help us to estimate, how long does it take to move from one incompatible network architecture to another.

Methodology and dataset for measuring IPv6 content

IPv6 penetration among content providers can be measured by checking the appropriate resource records (RR) in DNS. Table 2.4 briefly describes the RRs that can be tested. If any record contains `CNAME` record it should be followed until a valid record for IPv4 (`A`) or IPv6 (`AAAA`) is found and availability of these records should be tested periodically. The web services should also be checked if an alternative record for IPv6 (e.g. `www6, ipv6, www.v6`) is available. The alternative records are sometimes used by network administrators for testing purposes.

The total number of tested domains will determine the precision of the measurement. The results will be more accurate with the larger number of tested domains. A favorite

Table 2.4: Resource records checked

Service	Record type	Test
Web	A	www.<domain>
	AAAA	www.<domain>
	AAAA	www6|ipv6|www.v6.<domain>
Mail	A for MX	<domain>
	AAAA for MX	<domain>
DNS	A for NS	<domain>
	AAAA for NS	<domain>

source of domains is a list of the most visited domains provided by Alexa The Web Information Company. It is possible to download a daily updated list of the top 1 million sites for free in `csv` format from the Alexa's webpage. However, we believe that only the top list is not enough. For example, there are about 4 700 domains within Czech TLD in the Alexa's top list. The total number of domains in Czech TLD is approximately 1 200 000 thus the list contains less than `0.4%` registered CZ TLD domains. We believe that this extensively small number of domains is not enough to provide accurate results of IPv6 penetration. Another drawback of using Alexa dataset only is that all sub domains are aggregated to appropriate TLD. It is quite common that subdomains are used for different services, e.g., `scholar.google.com` and `maps.google.com`, however, subdomains are included neither in top 1 million sites nor the list of top 500 sites per country. Using only top 1 million sites, it is impossible to check if subdomains are accessible via IPv6 or not. It is, however, quite common that IPv6 is enabled only on the main website and not on other subdomains - especially if it is a large organization. In the previous example, `scholar.google.com` is accessible only over IPv4 and `maps.google.com` is dual stacked website (tested in July, 2016).

Are there other data sources available for creating a list of popular domains? One possible way is to use mail servers logs and do a reverse lookup for addresses sending the emails to our network. Domains can also be collected from DNS cache of a resolver. The amount of domains that can be collected using these methods, however, depends mainly on the network design and the number of users. In our case, we were able initially to obtain approximately 100 000 of valid domains. Unfortunately, the growth of domains in the database was slow. The main reason for the slow growth is that we are a campus network without control of DNS and mail servers used by different faculties of the university.

We developed the following solution to extend the number of domains in database and overcome the drawbacks of using only Alexa top sites and DNS cache. Several probes were deployed in the Brno University of Technology's campus network. These probes are still running and the monitoring process is still in place. Probes listen to all HTTP requests performed by users. The requests are inspected using the `httpry` HTTP analyzer [41]. The advantage of this solution is that the probes are able to intercept all HTTP traffic for all users from the whole university. The output from the analyzer is sent to a collector where the requests are stored in a database. Once per day, the update process adds the new unique domains into the central database. In April 2016, the database contained approximately 12 million of working domains [42]. The number is depicted in Figure 2.15, and it is still growing as connected users are actively building the database with their HTTP requests.

The collected list of domains is also used for the quality of service measurement. Web

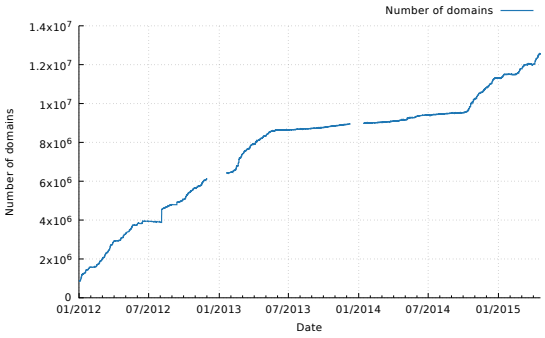


Figure 2.15: The number of domains in the database

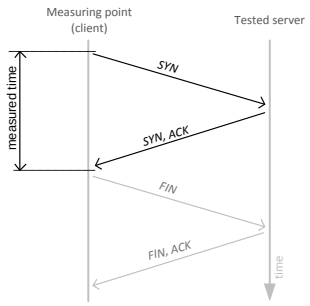


Figure 2.16: Round trip time measurement

domains supporting both protocols are checked and the response times for both protocols are measured and stored in the database. Using IPv4 and IPv6, the system tries to connect to the remote web server. The time is measured between the first packet initiating relation (SYN) and the answer from the server (SYN, ACK) as shown in Figure 2.16. Comparing the results obtained via IPv4 and IPv6, it can be observed, which protocol has a better response time. We are aware of the fact, that the IPv6/IPv4 response time can vary from location to location (due to asymmetric routing, missing IPv4/IPv6 peering, different number of hops, link quality etc.), however, the goal is to measure the quality from the perspective of our users.

To summarize, we tried to avoid to use a limited dataset as other projects that measure the IPv6 adoption among content providers. The benefits of our approach are following:

- Independence on third party datasets (e.g., Alexa list).
- Visibility of IPv6 enabled sites, which are interesting for our users.
- Visibility of subdomains in a TLD domain. The visibility is useful because Alexa list does not provide information, about visited subdomains, only aggregated data.
- Ability to check the quality of connection and compare IPv4 and IPv6 connection speeds.
- Long term solution with minimum maintenance.

Architecture and Implementation

The architecture of the monitoring system presented in the previous section is divided into several blocks and it is depicted in Figure 2.17. The core of the system is a database containing a list of domains and statistical data. There are two subsystems connected to the database. The first one is responsible for querying DNS system. It takes the list of domains from the database and periodically updates data with the information gathered from DNS. The history of changes for each record is stored as well, allowing us to provide current and historical data for each domain in the database. The next subsystem performs the check of IPv6 quality by measuring the one path delay as was described in the previous

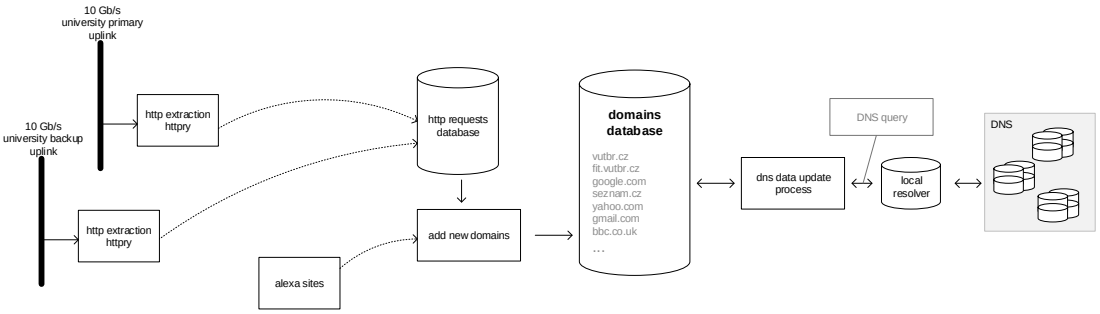


Figure 2.17: Architecture of the system

section. It also updates information into domains database and stores historical information about each measurement.

To update records and gather all data related to one domain, approximately ten queries are needed. The central database contains currently almost 13 million records. That means the system must perform around 130 million queries per day – more than 1500 queries per second because the DNS update process is executed every day. The system is using asynchronous DNS queries and threads to get the maximum performance. Domains are queried in a random order to avoid overloading of DNS servers. It is common that DNS system contains errors causing data inaccuracies. The measurement system uses following rules to get rid of invalid data:

- Queries returning addresses that belong to the reserved address space (private, link local) are ignored, e.g., `192.168.0.0/16, 10.0.0.0/8, fe80::/10`.
- Loopback addresses are ignored (`127.0.0.1/8, ::1/128`)
- Records with endless loop are ignored - a `CNAME` record points to another `CNAME` which points back to the original `CNAME`.
- The `CNAME` chain is followed up to level 10
- Manual patterns for domain names, e.g., domains containing random names like `www.335297538.acaoradical.com, www.335325653.acaoradical.com` etc. are ignored.

We remove domains that do not exist or have disappeared from DNS. If a domain does not contain any valid record (`A`, `AAAA`, `CNAME`, `MX`, `NS`), the domain is removed from our database including all related historical records. If the domain is added later using one of the approaches described above, it is treated as a new record in the database.

All dual stacked domains in the database are periodically checked once a week for IPv4 – IPv6 speed comparison. The whole update process takes close to ten hours for the database containing approximately 800 000 of dual stacked domains.

Table 2.5: IPv6 penetration among web, mail and DNS services - ratio on 8th of July 2016

	IPv4 only	IPv6 only	Dual stack	IPv6 alt. name
Web service	91.77 %	0.003 %	7.77 %	0.45 %
Mail service	84.19 %	0.011 %	15.8 %	
DNS service	70.13 %	0.008 %	29.86 %	

Data Analysis

Data collection is still ongoing. This thesis presents data collected up to April 2015. The total number of web domains is defined as NWT, the number of web domains supporting web services over IPv6 as NWV6, the number of web domains supporting dual stack as NWDS and the number of web domains supporting IPv6 web through alternative name as NWA6.

- NWT - domains having at least one IPv4 or IPv6 record announced for www.<domain>.
- NWV6 - domains having at least one IPv6 (AAAA) record and not having IPv4 record (A) announced for www.<domain>.
- NWV4 - domains having at least one IPv4 (A) record and not having IPv6 record (AAAA) announced for www.<domain> and not having alternative IPv6 record - e.g. www6|ipv6|www.v6.<domain>
- NWDS - domains having both IPv6 and IPv4 records announced for www.<domain>.
- NWA6 - domains announcing any of www6|ipv6|www.v6.<domain> via IPv6 (AAAA) and not announcing IPv6 for a record www.<domain>.

The penetration ratio of IPv4 only sites is computed using Equation 2.2. Other ratios are computed using the similar formula, but the numerator is changed accordingly to NWV6 for IPv6 only sites ratio, NWDS for dual stack ratio, etc.

$$NWV4ratio = \frac{NWV4}{NWT}100 \tag{2.2}$$

Based on these rules, we can analyze the data in our database to obtain the IPv6 penetration for the web, mail and DNS services. Table 2.5 shows that all these services are accessible using IPv4 protocol. More precisely, 99.997 % of web domains, 99.992 % of NS domains and 99.989 % of mail domains are accessible over IPv4. There is a slight number of IPv6 only domains (both web, mail or DNS), but these domains are without any meaningful content for end users. IPv6 only domains are mainly test websites and servers used for testing user's IPv4/IPv6 connectivity [13], sites where a www.<domain> has only AAAA record, but there is also a record for <domain> that is accessible via IPv4 or sites that are used for testing mail or DNS services over IPv6 only connection. The availability of DNS and mail services over IPv6 protocol is higher in comparison to websites. The higher penetration of these services corresponds to deployment strategy for a new service, where an administrator goes from the vital services to the less important ones, or he updates services where minimum changes are required. Updating DNS service for listening and answering DNS requests over IPv6 can be done easily. Updating mail service with IPv6 support introduces however many more challenges, e.g., missing antispam support,

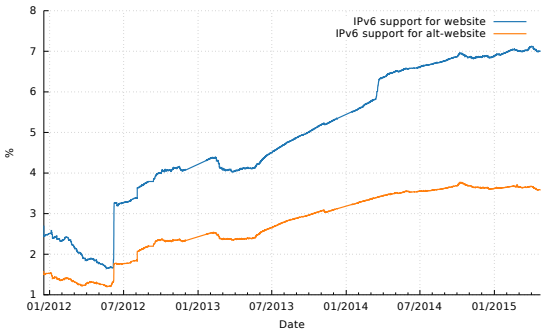


Figure 2.18: IPv6 penetration for web services, January 2012 - August 2015

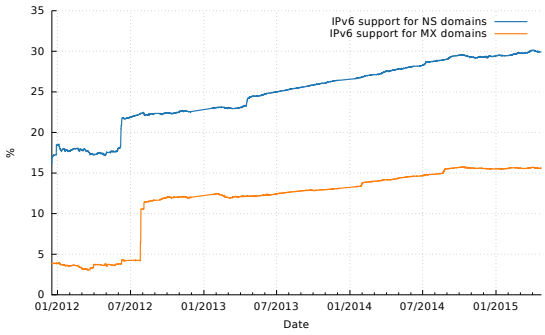


Figure 2.19: IPv6 penetration for NS and MX services, January 2012 - August 2015

intercompatibility issues, validation of DNS mail domains, etc.[7] Enabling IPv6 support for web service depends on the complexity of the service itself. Static content available over HTTP can be updated easily. If these websites are hosted by a large provider, sometimes the provider enables the IPv6 support for the web page as did Cloudflare in 2014[8]. IPv6 support for the rest of web pages is, however, problematic as these web pages are often owned by people with lack of necessary knowledge. There are also several issues with large and dynamic web pages. These web pages often use third party web plugins, advertisement network, etc. and these services are sometimes available only over IPv4 so they blocks the IPv6 support for the whole website.

Figures 2.18 and 2.19 show the progress of IPv6 penetration for mail, web and DNS services since 2012. These numbers confirm the observations described above. As we measure the content for a long period, all the main IPv6 events are visible. For example IPv6 Launch day in June 2012 and Cloudflare „IPv6 on by default“ in March 2014 are clearly visible in Figure 2.18. There was a high jump in IPv6 penetration for MX and NS domains as well as shown in Figure 2.19. According to our statistics, the main reason for the bounce in August 2012 for MX and July 2012 for NS records is that Google enabled IPv6 support for their Apps. Every website which uses Google Apps, e.g., for their work mails, started to be accessible over IPv6 since that time.

Validity of the Results

Every measuring system must prove that data provided by the system are trustworthy. If we would like to know the most accurate penetration of IPv6 services among content providers, we would need to collect all DNS data available in DNS system. This approach is, however, impossible due to the decentralized way how DNS system works. Fortunately, there are several projects trying to measure IPv6 penetration among content providers. Thus, we can compare our data with these projects. All similar projects use a list of domains provided by Alexa as a dataset, and usually only a subset of the top 1 million websites is used. It has all benefits and drawbacks described in the previous section. Hurricane Electric is an exception as they use the whole TLD zones for analysis. However, using only TLD zones, they cannot access information about subdomains.

[7] These issues are often discussed in NANOG, IPv6OPS and other mailing lists. Detailed description of these issues is, however, out of the scope of the thesis.

[8] https://blog.cloudflare.com/i-joined-cloudflare-on-monday-along-with-5-000-others/

Table 2.6: Comparison of projects measuring web content available over IPv6

Project	Data	Records	Tests	Freq.
[23] IPv6matrix	Alexa	top 1 million	`web`, `alt`, `MX`, `NS`	month
[24] IPv6observatory	Alexa	top 500/TLD	`web`, `alt`, `MX`, `NS`	defunct
[25] Eric Vyncke	Alexa	top 50/TLD	`web`, `alt`, `MX`, `NS`	daily
[26] Hurricane Electric	Alexa, zones	171 million	`web`, `MX`, `NS`	daily
[28] 6lab.cisco.com	Alexa	top 500/TLD	`web`, `alt`	daily
[43] cz.nic	.cz TLD	1.2 million	`web`, `MX`, `NS`, `DNSSEC`	month
[42] 6lab.cz	Alexa, Users	12 million	`web`, `alt`, `MX`, `NS`, `avail`, `DNSSEC`, `SPF`	daily

Table 2.7: Results of IPv6 penetration provided by different projects, July 2015

Project	.com	.cz	.de	.fr	.be	.ch
IPv6matrix	-	11.7%	12.98%	3.23%	3.59%	0.85%
IPv6observatory	-	15.9%	9.3%	9.6%	9.6%	-
Eric Vyncke	-	65.46%	30.76%	33.8%	30.6%	32.68%
Hurricane Electric	2.6%	-	18.08%	-	-	-
6lab.cisco.com	-	62.61%	45.72%	50.5%	50.43%	52.2%
cz.nic	-	24.6%	-	-	-	-
6lab.cz	5.89%	20.01%	17.4%	8.21%	4.74%	3.94%

The projects datasets and methodologies are compared in Table 2.6. The **Tests** column describes which of the following tests the project run.

- `web` test obtains the evidence of an `AAAA` record for selected domain.
- `alt` test checks an existence of alternative names for the domain (e.g., `v6.<domain>`).
- `MX` and `NS` tests are testing presence of `AAAA` record for mail and DNS services
- `DNSSEC` and `SPF` tests verify the support for these services.
- `avail` test measures the quality of connection using both IPv4 and IPv6.

There are other projects measuring IPv6 penetration that are not part of the comparison, e.g., RFC 6948 [27] or Dan Wing's statistics [44]. The reason for this is that these projects present the same results as others; they are using Alexa dataset, test the same metrics, etc. Thus, it would be just a duplication of information.

Results if IPv6 penetration for web services of selected TLDs that have the largest IPv6 penetration are compared in Table 2.7. The comparison is based on the data from the 1st of July 2015 or latest available. As we can see in Table 2.7, the obtained results are very different. Why is there such a big difference between projects? Figure 2.20 shows one of the reasons for such distinction. The chart can be interpreted as follows: IPv6 penetration (y-axis) is calculated for every number of domains (x axis). For example, IPv6 penetration for first three domains in Alexa list is `100%` for `.com` TLD. If the ratio is evaluated for top 5 domains, it drops to `80%`. If we use the same data source and top 500 domains, the penetration decreases to `8.4%`. The results show that it strongly depends on the number of involved records used for calculation of IPv6 ratio. It also means, that IPv6 penetration

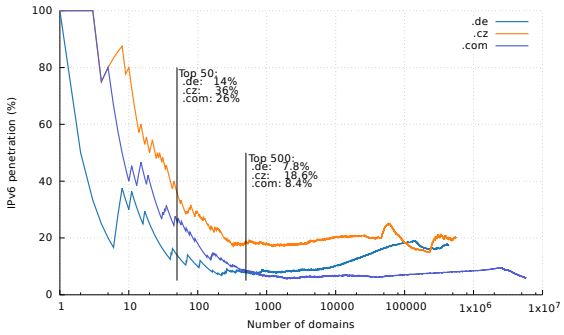


Figure 2.20: Dependence of the IPv6 ratio on the number of domains

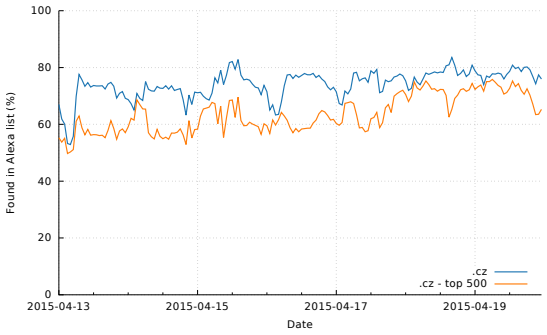


Figure 2.21: Percentage of .cz requested domains found in in Alexa list

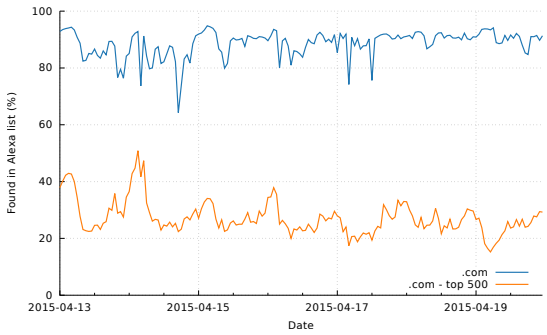


Figure 2.22: Percentage of .com requested domains found in in Alexa list

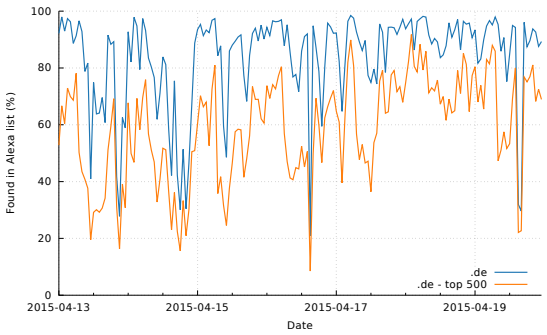


Figure 2.23: Percentage of .de requested domains found in in Alexa list

calculated only on several top websites and use only Alexa dataset is not ideal and does not report meaningful numbers. It can be seen that the IPv6 ratio becomes more stable for more than 1000 records. The increasing IPv6 penetration for some TLDs after 10 000 domains is caused by the fact, that the rest of the dataset does not have any ranking, thus is sorted randomly. If we want to receive meaningful numbers of IPv6 penetration, the number of analyzed domains must be increased at least to 10 000 records per TLD or more in the case of global IPv6 penetration. Using a smaller number of domains, we do not receive any precise number, and the IPv6 penetration is over-estimated. Similar observation was published by IPv6 Observatory [24]. A minimum of 10 000 domains is necessary to estimate the ratio of domains having an AAAA record as calculated from the top 1 million domains with an error in the range [-0.5;0.5].

Another reason for such a big difference between the projects' IPv6 penetration is that projects use different methodologies to compute IPv6 penetration. There are two main distinctions.

- **Geolocation:** 6lab.cisco.com and Eric Vyncke measurements use geolocation to identify a country for a particular domain [45]. This approach is useful for generic domains such as `.com`, where it is not clear to which country the domain belongs. For example, domain `wedos.com` is a popular hosting company in the Czech Republic. Their domain is registered as a generic `.com` domain, but domain's IP address is geolocated to the Czech Republic. The domain thus can be marked as a Czech Republic's domain, and it can be included in the Czech Republic's IPv6 stats. This approach is, however,

problematic as there are a lot of domains with the local content served outside of a country – a domain served by CDN (Content Delivery Network), a domain of a branch of a foreign company, etc. The problem is also a quality of geolocation databases for IPv6 addresses. These databases are not mature enough and often show an IPv6 prefix in a different country or region. We believe that the geolocation of domains is unnecessary and creates more problems than it solves. Furthermore, domains are usually also registered in the country's TLD thus they will be counted in the country's IPv6 penetration anyway.

- **Weight:** Some projects (e.g. 6lab.cisco.com or Eric Vyncke stats) use a weight of a domain to compute the IPv6 penetration in a country. The weight of a domain is an approximation based on position of a domain in Alexa list. The consequence of the weight is, that IPv6 enabled domains with better Alexa rank increase the IPv6 ratio for the country. This is based on an idea that users are more likely to connect, spend time and access content on a slight number of sites as stated in their report [28].

These ideas, however, introduce several questions. Can we just use the number of top N domains from the Alexa list because users mainly visit these sites? What is the ratio of domains that are found and not found in the Alexa list? Furthermore, what is the difference of IPv6 penetration between popular domains and domains not included in the Alexa list?

Answering these questions is very difficult. Fortunately, we can use information about users' behavior from BUT network to provide some numbers that can enable insight into this problem. To analyze the assumptions, we used all HTTP requests made by all users in the BUT network during `13.4.2015 - 19.4.2015` period. We analyzed all these requests and selected domains (`.cz`, `.com` and `.de`) were analyzed in more details. These domains were chosen as they were the most visited domains by our users, thus provided the largest dataset. Furthermore, `.cz` and `.de` domains were reported to have the highest IPv6 penetration. There were approximately 620 million unique HTTP requests during the period. We divided the HTTP analysis into two steps. Firstly, we tried to answer the question, if users visited mainly sites in Alexa list. Secondly, we measured the IPv6 penetration for domains found or not found in Alexa list to see if there was any difference.

The requested domains were aggregated to the first subdomain after TLD, e.g., domain `maps.google.com` was aggregated to `google.com`. It was due to the fact, that Alexa list did not contain subdomains. Figures 2.21, 2.22 and 2.23 show the ratio of requested domains found or not found in the Alexa list. Each figure displays the ratio of a domain found in top 500 per TLD and the whole Alexa list. We can see that there is a probability between 60 – 80 % that a user's request will be found in Alexa list for `.cz` and this probability is approximately 10 % smaller if we use only top 500 domains from Alexa dataset. The probability that a domain from `.de` TLD requested by BUT users will be found in Alexa dataset is between 20 – 90%. The reason for such a big difference lies in the fact, that `.de` TLD is the second largest TLD with approximately 16 million domains. Alexa list, however, contains only a small subset of `.de` domains (0.18 %). Even with our rather a small dataset, that contains 2.77 % of all `.de` domains, we can see, that there are a lot of domains regularly visited by BUT users that are not included in the Alexa list. The results confirm that a majority of requests is found in the Alexa top list, but there is still a significant number of requests that is not found. We can also see, that using only subset of domains, e.g., top 50 or top 500, provides very different results.

The HTTP requests were further analyzed to help us understand what is the IPv6 ratio of domains found and not found in the Alexa list. Figure 2.24 shows IPv6 penetration

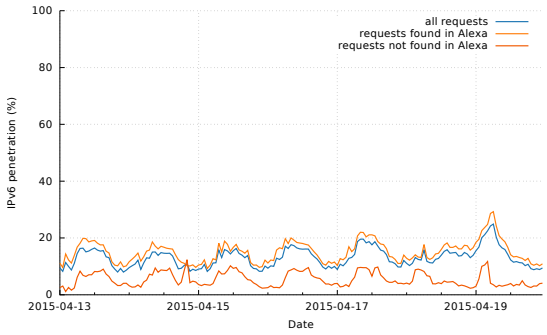


Figure 2.24: IPv6 penetration measured among all requests.

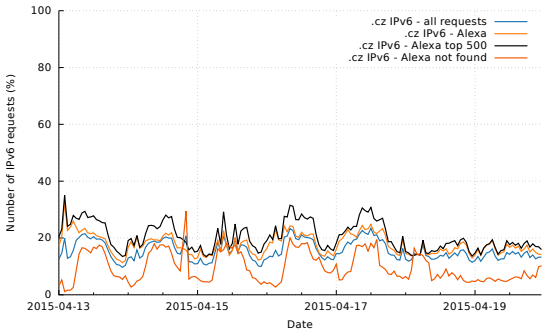


Figure 2.25: IPv6 penetration measured for .cz domain.

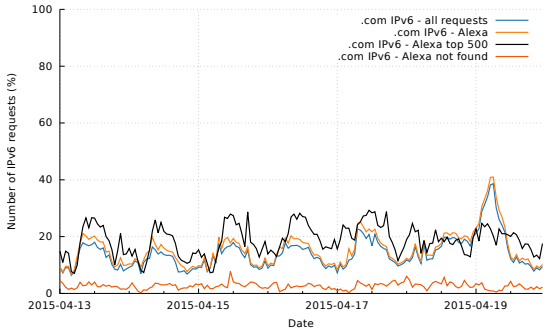


Figure 2.26: IPv6 penetration measured for .com domain.

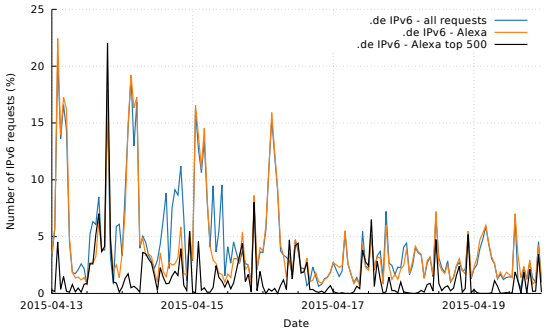


Figure 2.27: IPv6 penetration measured for .de domain.

of all requests, requests found in Alexa list and requests not found in the list. It can be seen that IPv6 penetration of domains found in Alexa list is higher than for domains that are not included in the list. However, the difference between IPv6 penetration of all requests and requests found in Alexa list is small. Figures 2.25, 2.26 and 2.27 show detailed measurements for `.cz`, `.com` and `.de` TLDs. We can see, that `.cz` and `.com` domains have similar patterns, e.g., domains found in Alexa list have higher IPv6 penetration, domains in top 500 have even higher IPv6 penetration and domains not found in the list have much lower IPv6 penetration. The exception is `.de` TLD where we can see, that there are a lot of IPv6 enabled domains that are not found in Alexa list.

Performance and availability of IPv6 domains

Implementing IPv6 support for a domain can introduce several new issues. It is not uncommon, that there is an `AAAA` record in the DNS system for a particular domain, but the domain is not accessible over IPv6 as there is no route to the host or a security admin forget to allow IPv6 HTTP traffic on a firewall. Unfortunately, finding and debugging these problems is very hard thus it takes a long time before they are fixed[9]. Missing robust routing infrastructure for IPv6 is another problem. Currently, there are fewer peering contracts and redundant paths for IPv6 compared to IPv4. These issues can lead to non-optimal traffic

[9]For example, fixing non-working IPv6 connection for several government websites in the Czech Republic took more than two years![46]

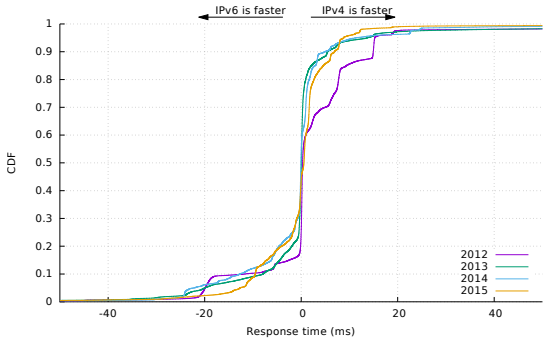


Figure 2.28: IPv4,IPv6 performance - CDF function.

Table 2.8: Number of measurements per year

Year	# measurements
2012	6,108,164
2013	19,198,299
2014	30,603,493
2015	26,660,007

routing, increase in latency or missing a redundant path if there is a route failure. An example of this behavior is missing IPv6 peering between Cogent and Hurricane Electric [47]. These two large companies are perceived as Tier 1 operators. Missing peering or transit contract between them create a split IPv6 Internet as BGP announcements from Cogent do not reach Hurricane Electric.

On contrary, there are several examples, where IPv6 outperforms IPv4. The most cited example is probably Paul Saab's (Facebook employee) presentation at V6 World Congress 2015 [48]. He stated that Facebook had seen users' *News Feeds* loading 20 percent to 40 percent faster on mobile devices using IPv6. Unfortunately, there was no explanation why it happened as the data analysis was still ongoing. Some consider that absence of NAT middleboxes is the reason for the better performance of IPv6 protocol, but it is just a speculation.

To obtain more precise results, we regularly check availability and quality of RTT of domains with `A` and `AAAA` RRs. Using IPv4 and IPv6, we try to connect to a remote web server and measure the time between the first packet initiating relation (SYN) and the answer from the server (SYN, ACK).

Figure 2.28 depicts the difference between IPv4 and IPv6 response times using data from January 2012 – August 2015 period. The number of measurements in each year is shown in Table 2.8. The round trip time is measured as described in section 2.2.2. The RTT difference between IPv4 and IPv6 protocols is counted using Formula 2.3, thus negative values represent measurements where an IPv6 response is faster than a response over IPv4.

$$RTTdiff = RTT\ IPv6 - RTT\ IPv4 \tag{2.3}$$

The graph in Figure 2.28 plots cumulative distribution function (CDF) in the interval <`-50 ms, 50 ms`>. CDF function represents the probability that the random variable X takes on a value less than or equal to x. We are using the CDF function, because if we normalize the output, we can use the output of the function to describe and compare all measurements in each year between each other. Because measured values are discrete, we can use following Formula 2.4 to compute the CDF function:

$$CDF(x) = P(X \leq x) = \frac{1}{n}\sum_{i=1}^{n} x_i \leq x \tag{2.4}$$

where n is number of measurements and x is response time of measured event. The cumulative distribution function at position x thus gives us the fraction of events that

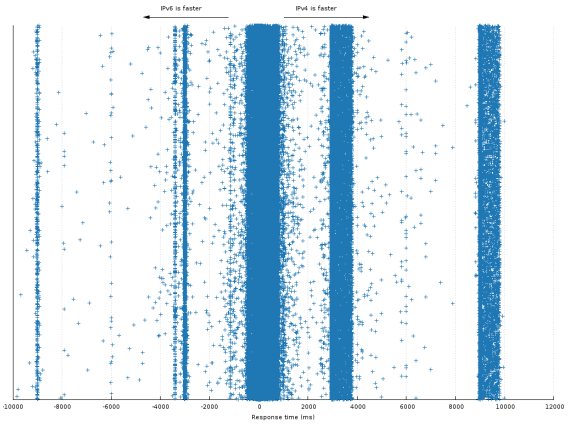


Figure 2.29: All measured RTT values in 2012.

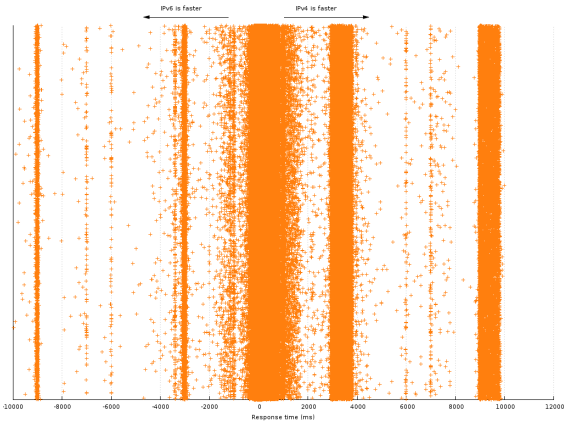


Figure 2.30: All measured RTT values in 2013.

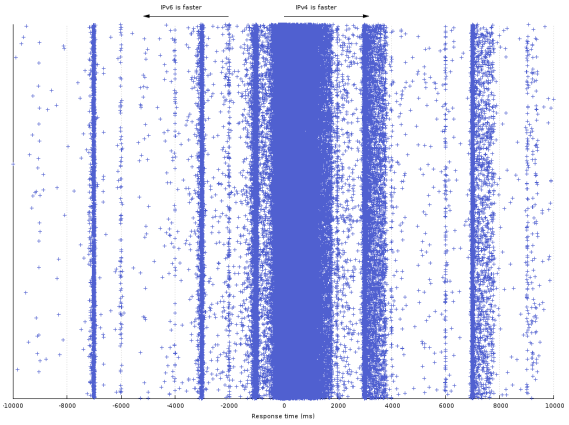


Figure 2.31: All measured RTT values in 2014

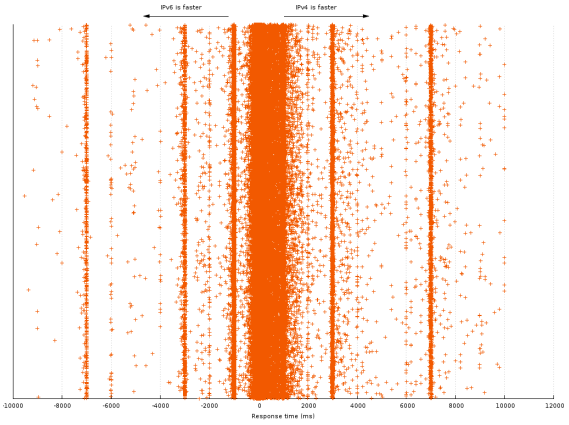


Figure 2.32: All measured RTT values in 2015

have occurred with response time x_i less than x. We can see, that round trip time values between IPv4 and IPv6 are very similar in confidence interval between 10 – 90 %. There were disproportions between response times in 2012, where IPv4 was faster more than 1 ms in 40 % of measurements but since then the performance of IPv6 improved and is very similar to IPv4.

Figure 2.28 clearly shows, that the performance of IPv4 and IPv6 are very similar. There are however cases, where IPv6 performs significantly better or worse than IPv4. As these cases are not visible in Figure 2.28, we depict all measurements in each year as a scatter plot. Figures 2.29, 2.30, 2.31 and 2.32 were created by shifting each data point in a year vertically by a random amount. Such jittering by a random number can be used to detect clusters. We used all RTT values in each year. It can be seen, that there are patterns in each figure. For example, there were a lot of RTT values, where IPv4 response time was between 3000 – 4000 ms or 9000 – 10000 ms faster than IPv6, especially in 2012 and 2013 period. We can find similar patterns for IPv6 protocol as well. However, the number of measured RTT values where IPv6 performs better than IPv4 is lower. It is also visible that these patterns are converging to 0 during the years, which means that IPv6 performs similarly as IPv4. We analyzed differences between RTT values in detail and found, that

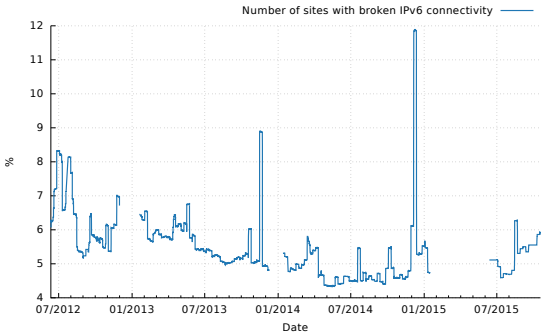


Figure 2.33: Number of domains we are not able to connect to, 2011 – 2015 period

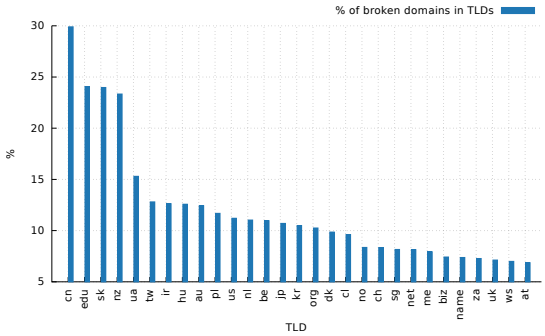


Figure 2.34: Percentage of broken domains in TLDs on 21.10.2015

they were caused by different paths in the routing system. IPv6 performed better when there were fewer hops in a path. The same applied for IPv4 too.

Our dataset and analysis thus confirm Paul Saab's claim that IPv6 can perform significantly better than IPv4. The reason, however, is probably not a middleware (e.g. NAT) as it is believed, but differences between IPv4 and IPv6 paths. Furthermore, situations where IPv6 outperform IPv4 significantly (IPv6 is more than 100 ms faster) are rather rare and we measured more RTT where IPv4 performed better. Nevertheless, the good sign is that majority of RTT values is very similar, thus if everything is set up correctly, there is no significant difference between both protocols.

Unfortunately, there is still a substantial number of sites that announce AAAA records but we are not able to connect to. Figure 2.33 shows the percentage of all domains to which we were not able to establish a connection in 2012 – 2015 period. We can see, that there was around 5 % of all websites with broken IPv6 connectivity. The reasons can vary. According to our experiences, the main problems involve routing, security filtering, misconfiguration of DNS or application issues. These non-working domains or slow IPv6 connectivity are nowadays usually handled by implementation of Happy Eyeballs in modern web browsers. However, there is still a large fraction of clients without Happy Eyeballs implementation. Moreover, Happy Eyeballs works usually only for TCP and HTTP thus other protocols are still affected.

Figure 2.34 shows the number of broken domains per TLD. The figure is based on data from October 2015 and shows 30 TLDs with the highest number of broken domains. Only TLDs with more than 10 000 domains are considered. We can see a significant number of broken websites from `.cn`, `.edu` or `.sk` TLDs. We analyzed `.sk` TLD more deeply to find out, why there was almost 25 % of domains inaccessible. We discovered that a large content and hosting provider in the Slovak Republic published `AAAA` RR for many of their hosted web pages. Unfortunately, they probably misconfigured routing or security filters in their datacenter thus all our RTT probes failed. Interestingly, we reported the problem to their helpdesk 14 days after the outage. They were not aware of the fact that several hundreds of their hosted web pages are inaccessible! Moreover, the issue was not resolved even after half a year. The reason why users are not complaining is probably the presence of Happy Eyeballs in their browsers. The user's browser usually switches from non-working IPv6 to working IPv4 protocol without user intervention, thus, everything works fine from user's perspective. The underlying network infrastructure is, however, broken.

2.2.3 User penetration analysis

Several new networking architectures have been proposed as incompatible with running architecture since the beginning of ARPANET. This incompatibility, however, introduces high capital and operating expenses (CAPEX and OPEX). ISPs have to upgrade their devices and train their staff, application developers have to update their application, security engineers have to consider new threats and update their security policies, etc. If we take economical perspective into a consideration, we can say that a new network architecture will be deployed only if the architecture solves some problems the ISPs or content providers have. IPv6 is an example of such an architecture as it tries to solve a problem the current architecture has – a shortage of available addresses.

ISPs are however evaluating benefits and drawback of every new technology very carefully, and IPv6 protocol is no exception. For several of them, it is still expensive to migrate to IPv6. These ISPs would consider to deploy IPv6 if there were enough evidences that IPv6 is well supported by operating systems, devices, content and routing infrastructure. We already covered the routing infrastructure and content in the previous section of this thesis. This section will be focused on users' support for IPv6 protocol, thus, trying to answer the following questions.

- If we deploy a new architecture, how many users will support it?
- How much traffic will flow over it?

The dataset used for generating following statistics is based on data collected in our campus network. We have been measuring the IPv6 support among clients since 2013. Figure 2.35 shows a percentage of IPv6 support among clients on our campus network. We can see, that there are approximately 80 – 85 % of users having IPv6 enabled and supported. There are drops in IPv6 support to 40 % during December and 40 – 50 % in June – August 2013 and 2015. These are caused by the fact that a lot of students and staffs go on holiday. Drop in IPv6 support during November – December 2014 is caused by changes in our routing infrastructure, thus, the numbers from this period are not relevant.

We only consider dual stacked eyeballs networks in our dataset, thus Figure 2.35 can be used as the answer to the first question. We can see, that the support for IPv6 is very high among end user devices. However, we have to highlight the fact, that these numbers are relevant only for campus network or similar network (e.g. enterprise), where users' devices are connected directly to the network. ISPs providing Internet access via cable, FTTx, xDSL or even Ethernet are in different position as there is a middleware (CPE device) between users and ISP core network. IPv6 support for CPE devices is unfortunately very problematic and typically only a small number of CPE devices have a decent IPv6 support.

What about the second question? How much traffic will flow over a new protocol? To answer that question, we have been measuring IPv6 traffic volume as well. Figure 2.36 shows the ratio of BUT traffic carried by IPv6 protocol. Since July 2015, we have been also measuring a ratio between IPv4 and IPv6 flows. We can see, that the traffic volume oscillates around 20 % and flow ratio around 10 %. There are drops in IPv6 traffic during summer vacation and Christmas as there are not so many users connected to our campus network. The average IPv6 traffic volume was following – 12.6 % in 2013, 13.04 % in 2014 and 13.25 % in 2015.

It can be seen there is no big difference between traffic volume now and three years ago. It is caused by the fact that a lot of popular websites producing and delivering a large

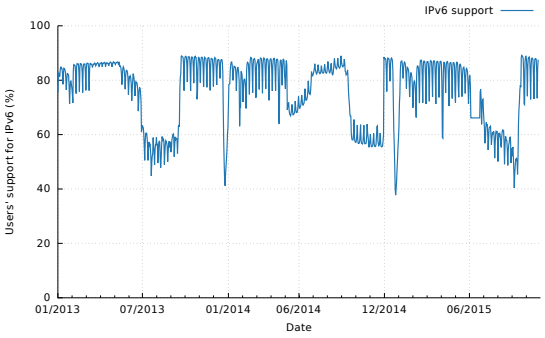


Figure 2.35: IPv6 support among BUT users devices, 2013 – 2015 period.

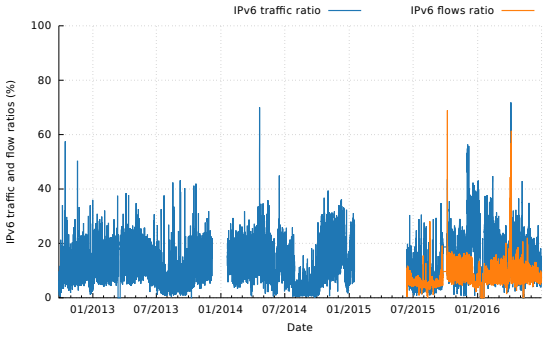


Figure 2.36: Ratio of IPv6 flows and IPv6 traffic

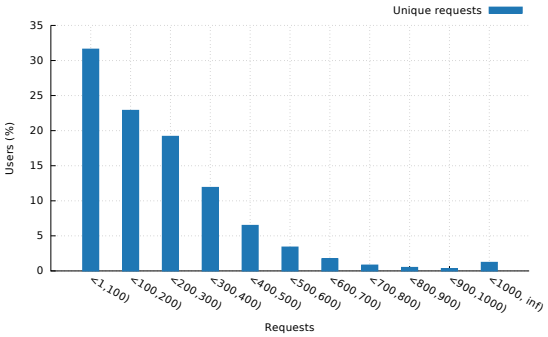


Figure 2.37: Number of unique websites visited by users in a day

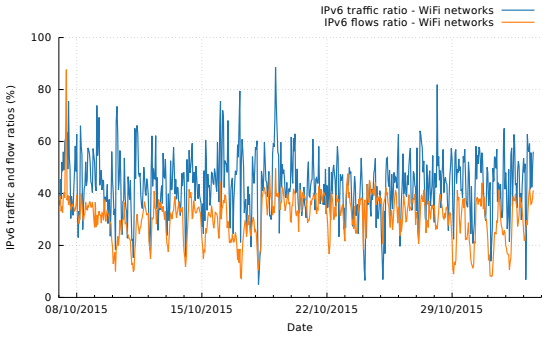


Figure 2.38: Ratio of IPv6 flow and IPv6 traffic in WiFi networks only

amount of content (e.g., Facebook, Youtube, etc.) deployed IPv6 already in 2013. Deeper analysis of flows confirms the fact that majority of our current IPv6 traffic is Google, Facebook, Akamai and several local content providers in the Czech Republic. It will be interesting to observe, if the traffic raise at same, higher or lower speed in future because the percentage of IPv6 traffic stays more or less at the same level. We believe that it will be very hard to raise the ratio from the current level as introducing IPv6 support for all small websites that comprises the rest of the network traffic will be a tremendous effort.

We have heard at several conferences during informal discussions that IPv6 support for small, rather static websites is not a big issue as users tend to visit only a few websites (social, mail, news) where IPv6 is already enabled. According to our statistics, this is not correct. We analyzed all individual HTTP requests of 9000 users in a day. There were 9.8 % of users that visited less than ten unique websites. The average number of unique domains visited per user was 231. We split the number of visited domains to the intervals as shown in Figure 2.37.

We see that 68.4 % users visited more than 100 unique domains in a day. It does not correspond with the premise that users visit only a few websites. On the other hand, the premise is quite valid if we consider only mobile devices. We have analyzed traffic and flows in our wireless networks. Figure 2.38 shows that the percentage of traffic and flows is much higher compare to desktop and laptop usage. The traffic volume ratio is 43 % in average, and flows ratio is 30.87 % in average. It is roughly two times more than the flows and traffic ratio in a wired network. Why are these ratios higher? Deeper analysis of traffic

shows that Facebook, Google (Youtube), Microsoft and Akamai are responsible for the majority of traffic. Mobile devices are thus mainly used for consuming content rather than for work. This observation corresponds to the number of users as presented by Google[10]. Google monitors IPv6 user penetration very carefully and see a difference between IPv6 penetration during working days (12 %) and weekdays (10 %). One can argue, that mobile market is rising while PC market is falling; thus IPv6 will gain more and more traffic in future. On the other hand, people still have to work thus the differences between IPv6 support at work and home will probably remain.

[10] https://google.com/ipv6

2.3 Summary

This chapter discussed issues with the transition to the next generation network. Next generation network protocols and future networking architectures are proposed as incompatible with the current TCP/IP architecture and current protocol of the network layer. We used IPv6 as an example of a next generation network protocol to describe the process of transition from the old architecture to the new one. We have been carefully monitoring the IPv6 usage in our campus network since the beginning of our IPv6 deployment. We thus provide a large dataset, statistics and insights into transition process that are usually not presented publicly. This summary highlights the most interesting results.

We divided the analysis into three main sections.

1. **Routing infrastructure analysis:** Section 2.2.1 examined global BGP table trying to find out current trends in IPv6 adoption. The overall trend is, that deployment of IPv6 is steadily growing. Considering the depleted IPv4 pools, the number of events, presentations and conferences promoting IPv6, this is not a surprise. There are, however, several interesting facts that we would like to highlight. The support for IPv6 is slightly lower among 32-bit ASes than 16-bit ASes (see Figure 2.2). Currently, a new company receives 32-bit ASN. Thus, it is interesting that support among new companies is low. The number of IPv4 32-bit ASes (new companies) is rising at a higher speed than IPv6 support 16-bit and 32-bit ASes together (Fig. 2.5). The growth of transit only ASes supporting IPv6 drops significantly in 2014 - 2015 period. Slower growth of IPv6 support in the routing core means smaller path diversity and robustness. Analysis of the number of IPv4 and IPv6 prefixes in DFZ did not reveal any surprises. All RIRs except AFRINIC depleted their address pools. Depletion of IPv4 address pools, however, does not mean that IPv4 addresses are not available anymore. There are still a lot of prefixes not announced in BGP and transfer markets emerged as an alternative to obtaining additional IPv4 addresses. The number of IPv6 prefixes is growing, but the IPv4 growth is still far faster than IPv6.

   We did a correlation of BGP analysis with NetFlow data from BUT and CESNET networks. It is a novel approach, as BGP analysis is usually presented without any relationship with the real network traffic. We found out that there were around 1.5 unique `/64` prefixes in one `/48` prefix in 2013 and around 2 unique `/64` prefixes in 2014. If IPv6 is deployed in a production network, the number of `/64` prefixes in one `/48` prefix should be much higher. The correlation thus suggests that IPv6 prefixes are probably still used mainly for testing purposes and not for the real traffic.

2. **Content availability and quality analysis:** Section 2.2.2 describes methodology an architecture of our measurement system. We presented several figures showing the speed of IPv6 support among the web, mail and DNS services since 2012. All these numbers are based on our dataset we have been actively building for several years. The primary benefit of our dataset is independence on a proprietary, third party dataset used by other projects. We argue that other projects using Alexa list overestimate the IPv6 penetration, as they only run the analysis on a subset of domains. We also found out that IPv6 support for domains not contained in Alexa list is usually lower.

   We performed analysis of the availability of IPv6 domains and measured the quality of connection. The conclusion is that IPv4 and IPv6 perform similarly in most cases. We also found out that occurrences where one protocol performs better than other are

mainly caused by differences in routing paths. Our measurements show that there is a substantial number of sites (around 5 %) announcing AAAA records, but we are not able to connect to them (see Figures 2.33 and 2.34). These issues are usually corrected by Happy Eyeballs implementation. Unfortunately, Happy Eyeballs works only for TCP and is implemented mainly in web browsers. Other protocols and services are thus still broken.

3. **User support and traffic volume analysis:** The last part of this chapter, section 2.2.3, describes support for IPv6 protocol among users' devices and IPv6 traffic volume. We presented long-term statistics of user IPv6 support in our campus network. IPv6 support is very high – around 80 % of devices (laptops, PCs and mobiles) actively use IPv6. The IPv6 support stays almost on the same level as three years ago. Comparing the results with other large networks, we see higher support in BUT campus. It is caused by the fact that we do not have middleware devices in our network. By analyzing traffic volume, we did not observe substantial differences between traffic volumes today and three years ago, because the main content providers already enabled IPv6 in 2012. There was not any bigger movement in IPv6 support for small websites that comprises the rest of the network traffic in recent years. Therefore, the traffic volume is very similar to the previous years. We also argue that an assumption about users visiting only a limited number of websites is not entirely correct. We found out that 68.4 % users visit more than 100 unique domains in a day (average is 231 unique domains). On the other hand, users tend to visit a smaller number of sites on mobile devices – mainly social networks, email services and chat. These services usually support IPv6, thus the IPv6 traffic and flow ratio is roughly two times higher than the flows and traffic ratio in our wired network.

This chapter presents several statistics and long-term views on IPv6 support. We described and documented the whole process of measurement the transition of one network protocol to another. We analyzed routing infrastructure, content and support among users. Several observations emerged. Firstly, the IPv6 support is steadily growing. Although one can argue it should be at higher speeds as IPv4 pools are mostly depleted, the overall trend is going up. Secondly, IPv4 is growing as well and even at higher speeds than IPv6. We can draw the following conclusions from these observations.

- IPv6 deployment in a production network introduces changes for user accounting. There are several differences between IPv6 protocol that make current techniques for user accounting more difficult. As we know that IPv6 support and deployment will grow, these issues with user accounting will be encountered by more and more ISPs and enterprises. Although some ISPs can benefit from IPv6 deployment and simplify their accounting tools, others will not. Therefore, there will be a demand for a scalable solution that will help with user accounting, network troubleshooting and tracking security incidents. We will cover issues in IPv6 accounting in the following chapter 3.

- To accommodate rising demand for IPv4 addresses, ISPs will have to employ more and more network address translation devices. Network translation is however introducing problems with user identification and accounting as well. Furthermore, current devices are often not prepared to export necessary information needed for accounting a user behind NAT middleware. We will cover these issues and introduce a scalable solution in chapter 4.

3

Transition technologies and users accounting

"Engineers always deliver the minimum so if you asked for a network that spans the galaxy you can hope that at least you can get one that spans the world."
– Bret Victor, *Future of programming*

The previous chapter described a history of transitions between incompatible protocols since the beginning of the ARPANET. We discussed the issues with the transition toward a new and incompatible networking protocol and presented several statistics and analyzes to find out the current trends in migration from IPv4 to IPv6 protocol. The summary is that we are slowly moving up with the IPv6 deployment. Other networking protocols and architectures seem to be still in a development stage and not widely deployed. Thus, we turn our attention to differences in users accounting between IPv4 and IPv6 protocols in this chapter. This chapter will firstly describe the principles used for users accounting in the current TCP/IP architecture. As it is out of the scope of the thesis to cover all possible information about address assignment, monitoring and accounting, the reader is expected to have an advanced understanding of IPv4 protocol.

A historical perspective

There were several discussions going on about the inevitability of a new protocol for the network layer at the beginning of the 1990s. Several proposals emerged – CNAT, IP Encaps, Nimrod, and Simple CLNP were the first ones. Later, PIP (The P Internet Protocol), SIP (The Simple Internet Protocol) and TP/IX were added. In 1992, the Simple CLNP evolved into TUBA (TCP and UDP with Bigger Addresses) and IP Encaps evolved into IPAE (IP Address Encapsulation). The IPAE proposal was merged with SIP and later merged with PIP to became SIPP (Simple Internet Protocol Plus). These proposals tried to solve issues with IPv4 address exhaustion while providing additional features for the new protocol. In the end, engineers did not yield to the temptation to redesign IP protocol from the ground up. The main ideas for the clean slate design were the following:

i) Free the new protocol from the historical shortcomings.
ii) Expand the addressing capabilities.
iii) Simplify the header.
iv) Improve security and Quality of Service.

Also, it was believed that there is no other way. A new protocol for network layer **must** contain a larger address space, thus, it could not be compatible with the previous one. IPv6 was born.

As stated above, it has been claimed that IPv6 will be better for a quality of service, security, that will simplify network configuration and many more. The reality is, however, the same as the quote at the beginning of this chapter. Engineers always deliver the minimum and the IPv6 protocol is no exception. Several features that were proposed ended as unrealistic or impossible to deploy at scale, e.g., security, quality of service or ease of transition. The only benefit that remains is the large address space, but even there are some concerns and unresolved questions – especially in the field of RIR policies for address assignment and burning the whole half address space for end networks (`/64` requirement for the end user identifier). These are, however, out of the scope of this thesis. Despite the fact that IPv6 protocol is not perfect, currently it is the only protocol that is deployed on a large scale. Considering all money that has been pushed into IPv6 migration, we can expect that IPv6 will probably become the next protocol for the networking layer.

We already highlight the fact that IPv6 was designed from the clean slate perspective. Although IPv6 may look like IPv4 protocol only with larger address fields on the first sight, the underlying details are very different. We will discuss the relevant part of the IPv6 protocol in more detail because there are many issues and misunderstandings about the protocol behavior.

Structure of the chapter

The goal of the first part of this chapter is to equip the reader with all necessary background information about user accounting in today's ISPs networks. The information stored by ISP's accounting process is often used by ISP for internal purposes, e.g., to track back a security incident. Accounting information is also demanded by law enforcing agencies, thus ISPs store them to obey a data retention law. The data retention law varies from country to country, but the user identification is always necessary for ISP internal purposes. There are of course exceptions, e.g., WiFi hotspots or guest networks where the identification is not stored, but in general, every ISP implements some techniques, how to identify a user on its network.

The rest of the chapter discusses differences between users accounting in networks that migrated to the IPv6 protocol. The incompatibility between IPv4 and IPv6 protocols introduces several new transition mechanisms that can be used, e.g., dual stack or tunneling techniques such as 6to4, Teredo, ISATAP, etc. These mechanisms, however, present several problems for the current accounting methods. We will highlight these issues and outline a system that is able to process all necessary information for user accounting in these networks at scale.

3.1 Current accounting techniques in IPv4 networks

Commercial ISPs in the early 1990's developed a business of providing Internet access for a fee. The first access usually took a form of a dial-up connection [49]. An IP address was assigned to a customer by his ISP for the duration of customer's call and then returned to the pool of addresses for subsequent reassignment for other customers. The ISP were storing the call detail records of the customer's call to be able to generate a telephone bill for him/her later.

If there was a demand to obtain the information who was using a particular IP address, the central `whois` registry was queried to receive information whom to ask (which ISP is responsible for the IP address range). If the questioner obtained an appropriate authorization, it could ask the ISP for access to the accounting records. The time of the day and the IP address were two pieces of information for the ISP to be able to pair the IP address with the telephone number and thus with the user.

The dial-up connection has been superseded by different technologies, although it is still used in these days. Cable connection, xDSL, WiFi, Ethernet or optical fiber are the technologies that have started to become predominant to provide an Internet access. These connections put different requirements to which information must ISP preserve to comply with the data retention policies. These requirements can be summarized in the following points: [1]

- **IP address:** The IP address is usually an identifier from the data retention point of view [50] as well as from the ISP management point of view (information necessary for billing, etc.). It depends on which model of address assignment the ISP is using because the access to the Internet quickly changed from a dial-in model to an always-on model. If a user always has the same IP address for the whole period of the contract with the ISP, the ISP only needs to store information, which user has which IP address. This is relatively straightforward and the information is usually stored in a relational database with other information such as name, address, phone number of the subscriber, etc.

- **Timestamp:** The IP addresses can also be assigned in a dynamic way. An example can be a dial-up connection as described above, but the dynamic assignment method can be used with any other technology (e.g., Ethernet + DHCP where the DHCP server does not provide fixed addresses). If the dynamic assignment is used, the law enforcing agency must provide a timestamp together with the IP address. The timestamp is necessary, because without the timestamp, the ISP could not pair the IP address with the correct subscriber. Which address assignment model the ISP is using is not public information, thus, the timestamp is always a part of a data retention request.

- **Timestamp to IP address mapping:** Previous two points described two primary attributes that a law enforcing agency uses when requesting accounting records from the ISP. To be able to satisfy the request, the ISP must be able to pair a given IP address with the correct subscriber. Different address assignment methods for end devices can be used, but two are used the most – DHCP and PPP. It is easy to provide the mapping between an IP address and a subscriber if the subscriber gets a fixed IP address. If an ISP uses dynamic assignment, then the ISP must store logs from DHCP or BRAS[2] server. These logs provide information about the mapping between time and IP address. In case of fixed configuration, the mapping is available in DHCP's configuration files or network management system.

- **Network layer to data link layer mapping:** If a dynamic address assignment model is used, the mapping between time and IP address is necessary but not sufficient. The another required information is a mapping between network and data link

[1] The following part of the text describes the situation where the **globally unique** IP address is assigned to an end user. Later sections will describe different scenarios.

[2] Broadband Remote Access Server

layer. It strongly depends on a design of an ISP network together with a software which the ISP is using, because there are several different options and ways how to store the necessary information. Considering the Ethernet and DHCP as one of the most used technologies in these days for address assignment and last mile access, usually it is the MAC address of subscriber's computer or CPE device that is stored. The ISP can, however, store different information like DHCP Relay Agent Information Option [51] or logs from RADIUS server. If PPP, PPPoE, PPPoA are used, a session-id or a username can be the information that maps an IP address to the correct subscriber.

All of the mentioned information must be stored by an ISP as without the information, the ISP will not be able to fulfill the data retention request. The Brno University of Technology's dormitory network will be used as an example to illustrate in detail which information must be used and stored to comply with data retention requests in the Czech Republic. The dormitory network provides access to approximately 6 000 of students in different parts of Brno.

- Whenever a student plugs his/her computer to the BUT network, the DHCP is used for address assignment. If user's MAC address is not registered in our system, a private IP address from a reserved pool is used. These addresses do not have access to the Internet. All HTTP requests from these addresses are redirected to a web portal. The user logs in to the portal to view a simple web form. MAC address of the user's computer and port on the switch, where the user is connected are automatically filled in the form[3]. If the user confirms the form, the information is stored in a database.
- A relational PostgreSQL database is used to store student's personal information together with a room location, MAC and registered IP address as described in the previous step.
- The registered MAC addresses from the central database are used to generate DHCP configuration, ensuring that a student's device always obtains the same IPv4 address. The DHCP configuration uses different pools of IPv4 addresses which are allowed to access the Internet. The addresses are globally unique. Dormitory network segment does not contain NAT or any filtration.
- If there is a data retention request or a network administrator must solve a security incident connected with an IP address, the admin only queries the central database to obtain all of the information. IP address and timestamp are sufficient to query the database. The L3 to L2 mapping is still necessary at the beginning of the device's registration.

The steps described above ensure that a network administrator always has all the necessary information who is responsible for which IP address. This is not only important from the data retention point of view, but also from the view of network management. It depends on the ISP's network and technologies used in the network, which information must the ISP store. The similar model to BUT is used by other universities in the Czech Republic, as well, e.g., [52], [53] and [54].

Regardless of which accounting method is employed, binding between an IP address and a user is becoming insufficient. For example, the data retention regulation in the Czech

[3]These fields are obtained from DHCP log and using SNMP requests

Republic also requires IP addresses to which the user communicated with[4]. To fulfill the requirement, ISP must store some level of packet activity of all subscribers.

There are several possible options that can be used for monitoring subscribers activity. The main techniques are packet traces, flow statistics or volume statistics. Packet traces provide the highest amount of information, but scale badly. It is very hard to store all records in the bigger network for the requested data retention period, which varies from 1 to 6 months. There is a problem with privacy of subscribers as well. Volume information, e.g., an amount of transferred data on an interface, can be used only for network management purposes as this technique does not provide any detail about user communication. A flow based monitoring stores only a part of packet headers without a payload. It provides a summary who communicates with whom and fits nicely for network management and data retention purposes. The flow based monitoring is described in the following section.

3.2 NetFlow

NetFlow accounting evolved from a caching technique introduced by Cisco Systems Inc. to speed up routing in their devices. A side benefit of the caching technique was the collection of useful statistics which could be reported to a network administrator. Initially, the statistics were reported using a proprietary protocol (NetFlow) that was available on devices produced by Cisco. Other vendors adopted the technology as well in a short period of time. Today, the NetFlow protocol or its modification is supported by most vendors. There is also IPFIX – an open standard protocol defined in RFC 7011 [55], but its support vary among vendors. The whole flow accounting process is also called NetFlow, which can sometimes be confusing for the reader. The thesis is using NetFlow as the whole process – from generating flows to delivery the flows to the collector. Flow monitoring is de-facto a standard way, how the accounting is done in today's networks.

The flow monitoring creates a flow. The flow definition can vary, but it is usually the following: „*All packets with the same source/destination IP addresses, source/destination ports, protocol, interface and class of service*“ [56]. These packets are grouped into a flow and then a number of packets and transferred bytes are tallied as shown in Figure 3.1 [1].

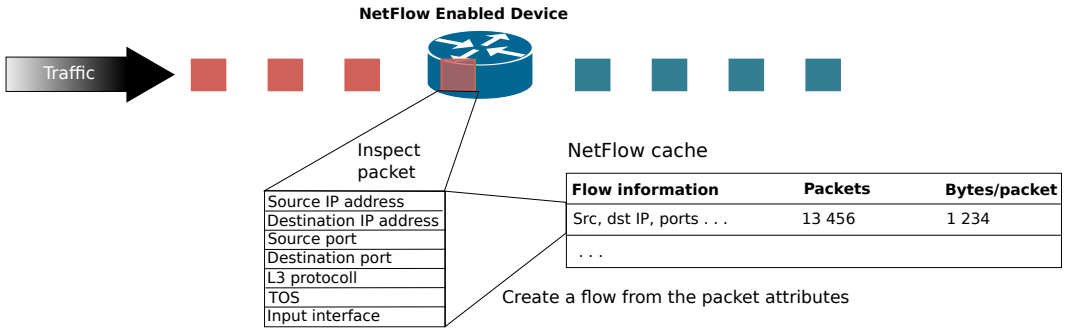


Figure 3.1: Creating a flow in the NetFlow cache [1]

Information, such as who communicates with whom (source, destination IP addresses), which application (source, destination ports) and time (start, end timestamps), are usually

[4]This is no longer the case, because the last revision of the data retention directive in the Czech Republic requires **only** the user identification – it means that ISP must not log the destination IP addresses.

Table 3.1: Flow data obtained on 13th January 2015 in Brno University of Technology and CESNET NREN network

	BUT	CESNET
Average Bit Rate	2.127 Gb/s	18.4 Gb/s
Maximum Bit Rate (peak)	3.604 Gb/s	32.3 Gb/s
Total Number of Flows	608 000 000	8 062 000 000
Average Flow Rate	7 000 flow/s	93 000 flow/s
Maximum Flow Rate (peak)	10 000 flow/s	152 000 flow/s
Consumed Disk Space (compressed)	25 GB	240.3 GB
Consumed Disk Space (uncompressed)	48 GB	482 GB

sufficient to fulfill a data retention request. The same information can be used for network management, where NetFlow can facilitate identification of a new application, detect unauthorized WAN traffic, trace back security incidents, etc.

Because NetFlow data can be used for network management purposes and also for data retention, the flow monitoring is prevalent technique used by network administrators. Reduction of consumed disk space is the another benefit of storing only relevant parts of packet headers. The amount of data that are stored per day is shown in Table 3.1[5]. Saving full packet traces with the average bit rate of 2.127 Gb/s would mean approximately 183,7 TB of data per day, thus, the disk space reduction is significant (85 %).

3.3 Address assignment in IPv6

This section describes address assignment techniques in IPv6 protocol. These techniques are discussed in more detail compared to IPv4, because we assume, that the reader is more familiar with the IPv4 and DHCPv4 than with the IPv6 protocol. We expect that the reader is familiar with the address types and notation in IPv6, if not RFC 4291 [33] and RFC 5952 [57] could be used as references.

One of the main differences between IPv4 and IPv6 protocols is that IPv4 uses only one address per interface. Although a workaround exists and almost all operating systems and network devices allow to configure a secondary IP address on an interface, it cannot be done automatically. Some network designs use more IPv4 addresses per interface in a situation where there is a need to have a public and private prefix on the same VLAN interface, but the typical case for an end node is one IPv4 address per interface.

Contrary to IPv4, IPv6 capable device can have (must have in a real world deployment) multiple addresses per interface. Together with a link-local address which is mandatory, the interface may have several other global unicast, unique local or any other addresses from assigned address space [58].

3.3.1 Stateless address configuration

IPv4 protocol supports two ways of IPv4 address configuration - manual address configuration or DHCP (BOOTP). IPv6 maintains both methods (DHCP is replaced by DHCPv6) and introduces a new one - Stateless Address Autoconfiguration (SLAAC) [59].

[5]The table is same as in the previous chapter. For the clarity, it is presented here as well.

Both SLAAC and DHCPv6 provide automatic address configuration. The main difference between these two approaches is that SLAAC allows a host to determine an IPv6 address by itself, thus, there is no need to keep track of assigned addresses at any node (server, router). It is the exact opposite of DHCPv6 operation where a DHCPv6 server maintains information about assigned addresses or prefixes. However, there is a case where a client can generate own address even if DHCPv6 protocol is used – if DHCPv6 Prefix Delegation mechanism is used.

SLAAC can be described in the following simple terms: A network router tells all the connected nodes in a network segment what network they appear in (network prefix), and what router they should use for packets traveling to the Internet (default gateway). This information is carried in a Router Advertisement message (RA) that is transmitted periodically. A host can send Router Solicitation (RS) message to initiate the process.

After receiving RA message, end nodes have information which network prefix should they use and how to route packets. This information is, however, insufficient. The host still needs to know, how to create a host part (end user identifier – EUI or Interface Identifier – IID) of his IPv6 address. The host part of the IPv6 address can be derived from information that the host already has, such as, a network-card link-address [33], generated randomly [60], cryptographically generated [61] or via a different mechanism.

To describe it in more detail, the Router Advertisement message is depicted in Figure 3.2. The important fields related to this thesis are `Router Lifetime`, flags `M`, `O`, `Prefix Information Option` and `A` flag in the `Prefix Information Option`.

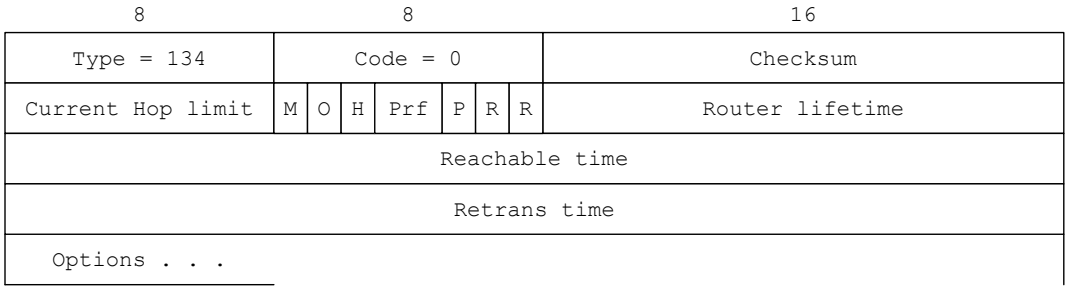


Figure 3.2: Router Advertisement message

- `Router Lifetime` – a value from interval $< 0, 9000 >$ seconds that indicates for how long the router is willing to behave as a default router. Lifetime of 0 indicates that the router should not be used as default one.
- `M` – Managed Address Configuration flag – when set, it indicates, that an address or addresses should be configured statefully using DHCPv6.
- `O` – Other Configuration flag – when set, it indicates, that other configuration information can be obtained from a DHCPv6 server, e.g., DNS servers.
- `Options` – The RA message allows several types of TLV[6]-encoded options. Relevant option for the thesis is the `Prefix Information Option` depicted in Figure 3.3 and fields `Prefix`, `Prefix Length` and `A` flag.

[6]Type-Length-Value

– `Prefix` and `Prefix Length` – IPv6 prefix and prefix length.
– `A` flag – Autonomous address-configuration flag – when set it indicates that this prefix can be used for stateless address configuration.

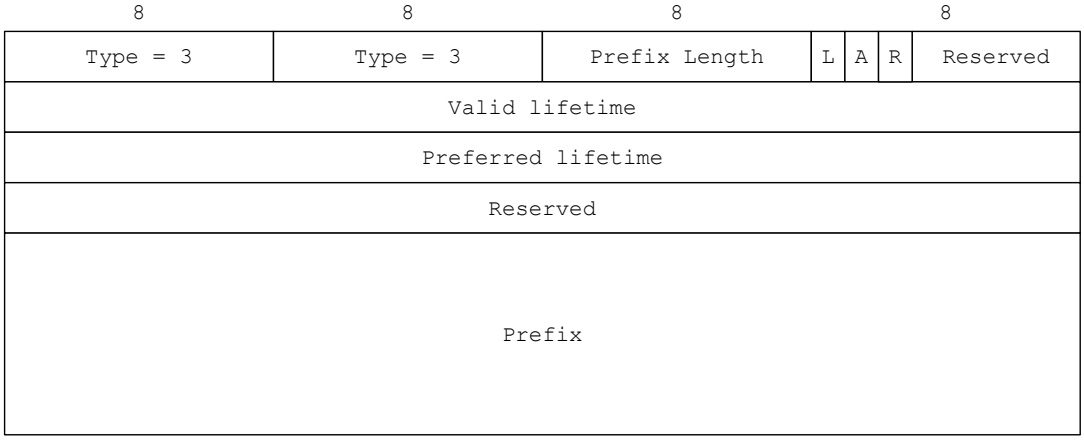


Figure 3.3: Prefix Information Option

Meaning of other flags and fields can be found in relevant RFCs – RFC 4861 [62], RFC 5175 [63] and RFC 6275 [64]. The typical scenario of address assignment using SLAAC in Ethernet networks can be then briefly described as follows:

- Firstly, a host must create a link-local address and assigns it to an Ethernet interface. The link-local prefix is known (`fe80::/10`, a more specific `fe80::/64` is used for Ethernet), the end user identifier (EUI) is derived from MAC address (older Linux, MAC, iOS operating systems) or generated randomly (Windows OS, Windows Phone). The host combines the link-local prefix with the EUI and receives a 128-bit IPv6 link-local address. The link-local address is assigned to the interface and marked as tentative because the host does not know if the link-local address is unique on the network. The discovery of uniqueness of the link-local address is done by Duplicate address detection (DAD) process. To be able to run DAD process, the host must also join *allhosts* and *solicited-node* multicast groups.
- If DAD ends successfully for the link-local address, the system marks the address as valid and use it for receiving a Router Advertisement (RA) message or sending a Router Solicitation message. If the RA message contains Prefix Information Option (PIO) and it is allowed to use the announced prefix for stateless configuration (`A` flag is set), the host will create another IPv6 address combining received prefix and EUI. It depends on host's operating system which algorithm will be used to generate the EUI. The majority of operating systems are using Privacy extension nowadays, thus, the EUI is generated randomly. The host must then join to a corresponding *solicited-node* multicast group for the new IPv6 address and starts the DAD process to verify its uniqueness.
- The host adds the router, which sent the Router Advertisement message, to the host's default router list indicating that off-links packets could be forwarded via the router.

Basically, the router sending the RA message is treated by the host as a default gateway.

The benefit of stateless address configuration is a very fast address configuration even in a large network. Only one Router Advertisement packet is usually necessary for any number of hosts in a LAN to configure their IPv6 addresses, default gateway and MAC address of the default gateway. A host, however, needs to know also addresses of DNS resolvers. These addresses were originally available only by using stateless DHCPv6 server, but the options for DNS configuration were added to the SLAAC in RFC 6106 [65] in 2010.

3.3.2 Stateful address configuration

The stateful configuration in IPv6 is done by the DHCPv6 protocol. In general, the DHCPv6 protocol is very similar to the DHCP protocol used in IPv4, however, the details of DHCPv6 are very different from DHCP in IPv4.

DHCPv6 features two basic modes – stateless and stateful. The first mode, stateless DHCPv6, can be seen as a simple layer on the top of the SLAAC process. The purpose of stateless DHCPv6 server is to extend the SLAAC autoconfiguration and pass other information to a client based on simple, request – response interaction of the client with the DHCPv6 server. The benefit of this approach is that the DHCPv6 server does not have to keep any state. Two messages are used – Information-request message and Reply message. The configuration options which DHCPv6 server can assign are listed at IANA webpage [66].

The stateful DHCPv6 can be employed to assign IPv6 addresses to a host or other configurations. Stateful DHCPv6 usually exchanges four messages to pass the necessary information – this is a similar approach as DHCPv4. If the client and the server support rapid commit option, just two messages are needed. The main differences between DHCPv4 and DHCPv6 can be summarized into the following points:

- Using a link-local address for requesting an IPv6 address from DHCPv6 server is a cleaner way, how to implement the client. DHCPv4 has to use system specific implementation because it is usually a problem to sent a packet through an interface if the interface does not have assigned any IP address.
- The DHCPv6 messages are multicasted to `All_DHCP_Relay_Agents_and_Servers` multicast group. If the network devices support MLD snooping, all DHCPv6 messages go directly to the server; if not, the client's Network Interface Card (NIC) drops the packet, because the client is not joined in the multicast group. DHCPv4 uses broadcast, thus, the client must parse every message.
- DHCPv6 can configure multiple addresses and multiple interfaces in a single exchange.

Two special flags are used for signaling if there is a stateful or stateless DHCPv6 in the network; the managed flag (`M`) and the other flag (`O`). These flags tell the client that it should ask for more information related to the connection parameters through DHCPv6. If the `M` flag is set, a client should request information using stateful DHCPv6 protocol. If the `O` flag is set, a client can ask for necessary information using stateless DHCPv6. If both flags are set to zero, the end-users stations know that there is no DHCPv6 server available in the network.

We can see the one of the biggest differences between DHCPv4 and DHCPv6: the DHCPv6 protocol is not a standalone protocol as in IPv4, but it is closely tight to the NDP protocol, specifically to the Router Advertisement message sent by routers in a network. Unfortunately, it is not clearly defined, whether a client should wait for receiving RA message and strictly follow the information inside the RA packet or not. Hence, the behavior of operating systems is very different, e.g., some operating systems do not wait for RA message and always try to obtain an address from DHCPv6 server. Some implementations take flags in RA message as a hint. Others strictly obey their presence. We will cover these issues and issues related to address assignment later in this chapter.

DHCP Unique Identifier (DUID)

Another fundamental change in DHCPv6 protocol is using different identifiers compared to DHCPv4. The DHCPv4 server uses client's MAC address as an identifier. DHCPv6 uses a concept of DHCP Unique identifier (DUID). The main idea behind the new identifier is to free the clients from dependence on hardware and a specific network interface. The advantage is that a change of a network adapter or a connection through another interface (such as WiFi instead of Ethernet) does not mean that the end-user station would start to behave like „someone else". According to RFC 3315 [67], the DUID identifier is designed to be unique across all DHCP clients and servers, and stable for any particular client or server. Furthermore, DUID must be treated as opaque values. Currently, there are four DUID identifiers defined.

- **DUID-LLT:** Link-layer Address Plus Time: This DUID is based on a combination of time and link layer address of any network interface that is connected to the DHCP device at the time that the DUID is generated. This is the default identifier in the majority of operating systems.
- **DUID-EN:** Vendor Based on Enterprise Number: The DUID is assigned by the vendor. It consist of the vendor's registered Private Enterprise Number and a unique identifier assigned by the vendor.
- **DUID-LL:** Link-layer Address: Link layer address of any one network interface is used.
- **DUID-UUID:** Universally Unique IDentifier [68] is used for this DUID.

3.4 Transition technologies

"We have more transition mechanisms than IPv6 packets."
– Randy Bush, *RIPE 62*

We have already discussed the problems with the transition to a new, incompatible protocol in the previous chapter. We came to the conclusion that network engineers have several options that can be used to overcome the incompatibility between protocols.

- **Dual-stack:** The most straightforward way for new nodes to remain compatible with the older nodes is by providing a complete implementation of both protocols and run both protocols simultaneously.

- **Tunneling:** A transition mechanism that provides a connection for the new nodes over the old protocol is another approach.
- **Address family translation:** There could also be a translation service or gateway between incompatible protocols.

This section will cover different types of transition techniques proposed for IPv6 protocol. We will cover dual stack and tunneling mechanisms in this section. Address family translation between same or different address families will be covered in the next chapter.

There are several transition mechanisms available to help moving networks from IPv4 to IPv6. Several of them were abandoned or deprecated, several of them are still used in production networks, even though that tunneling, especially over Internet, is perceived as a problematic. If a transition mechanism tunnels IPv6 or IPv4 packets over Internet, it introduces problems, e.g., different MTU size leads to fragmentation or blackholing traffic, a firewall can deny unknown protocols, etc. Despite these issues, there are situations where a tunneling mechanism is a best deployment option available. A list of transition mechanisms can be divided as follows:

- **IPv6-in-IPv4 tunneling:** This approach allows IPv6 packets to be transmitted over an IPv4 network. It is especially relevant during the initial phases of deployment to full, native IPv6 connectivity because it allows reusing old network protocol. The general idea is to embed an IPv6 packet in the payload of an IPv4 packet as described in RFC 2473 [69]. The protocol number in the IPv4 header is set to 41. Several mechanisms were designed to use this approach; e.g., 6to4, 6in4, 6rd or 6over4. These mechanisms are considered as stateless tunneling mechanisms, as there is no extra configuration burden for an end user or a network administrator. There is also an approach where a shim header is inserted between IPv4 and IPv6 headers. Examples of a shim header can be GRE or MPLS headers. These mechanisms, e.g., 6PE/6VPE, GRE tunnels or configured tunnels according to RFC 4213 [70] are considered as stateful because a static configuration is necessary. There are also stateless transition mechanisms that use a slightly different approach. Teredo was designed to be a NAT-aware tunneling mechanism. It uses IPv4 protocol for tunneling, but the IPv6 payload is encapsulated in UDP and Teredo specific headers. It provides an IPv6 connection for users behind IPv4 NAT. ISATAP is another example of IPv6-in-IPv4 tunneling. It is similar as 6over4, but requires a particular configuration of network, where it is deployed. Hence, it is not a stateless protocol as 6over4, but stateful. We will cover these tunneling techniques in more detail later in this section as several of these transition mechanisms are still deployed today.
- **IPv4-in-IPv6 tunneling:** Several transition mechanisms were designed to transmit IPv4 packets over IPv6 networks. Examples are DS-lite [71], MAP-T [72] and MAP-E [73], A+P [74] or 4RD [75]. These mechanisms allow to deploy IPv6 only network in ISP core. Eyeball subscribers are then connected to the IPv6 only network, and their IPv4 traffic is tunneled over the IPv6 only core. This approach is relevant in situations where an ISP wants to deploy only single protocol in its network core to limit OPEX. These mechanisms take benefit of the longer IPv6 address and often incorporate several fields from IPv4 and transport protocol headers directly to the IPv6 header.

- **Other type of tunneling:** There are transition mechanisms that use a different approach. LISP, BGP tunneling, or tunneling mechanisms as L2TP or VxLAN can be used during the transition from IPv4 to IPv6. These mechanisms create an overlay network that can be used for interconnection of IPv6 networks over IPv4 or L2 protocols. A deeper understanding of the functionality of these protocols is not required to understand principles in this thesis.

The tunneling transition mechanisms briefly introduced above have a severe impact on ISP accounting process. We highlight the fact that a flow based approach, e.g., using NetFlow protocol, is the current de-facto standard deployed for user accounting in ISPs networks. However, the flow accounting creates statistics only from the topmost header. If IPv6 is tunneled in IPv4 header, flow based accounting creates a flow according to IPv4 addresses and protocol numbers as depicted in Figure 3.4. In this case, the ISP loses the information about services and protocols transmitted inside the tunnel.

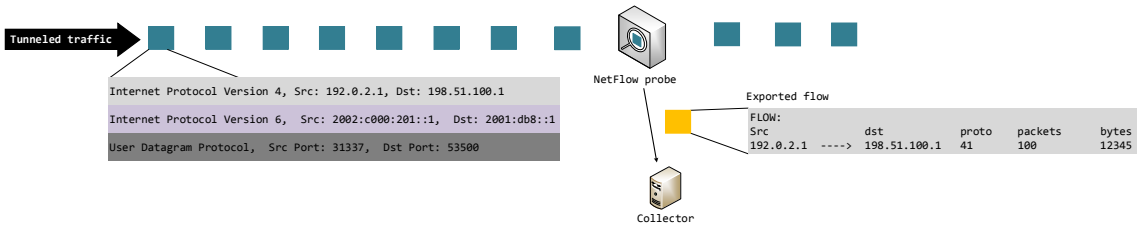


Figure 3.4: IPv6 tunneling over IPv4 protocol

How to overcome the limitation? Is it possible to deploy a system that accounts a user even if a transition mechanism is used? We will answer these questions later in this chapter. To understand what is necessary for user accounting, we firstly describe several tunneling mechanisms in detail. We select tunneling mechanisms that are used in production networks today and transition mechanisms that are not widespread so far, but can have an impact on users accounting, if deployed. Several transition mechanisms work similarly, thus, we pick just one representative for each group.

3.4.1 Connection of IPv6 Domains via IPv4 Clouds (6to4)

"6to4 is indeed an invention of the devil."
– Geoff Huston, *RIPE 71*

6to4 is a transition mechanism specified in RFC 3056 [76] for migration from IPv4 to IPv6. It allows any site with a globally unique IPv4 address to create a unique IPv6 address prefix. The transition mechanism was very popular because a user was able to obtain IPv6 connectivity even if user's ISP was not willing to deploy native IPv6 or a transition mechanism. The combination of 6to4 and RFC 3068 [77] (An Anycast Prefix for 6to4 Relay Routers) was designed to provide out-of-the-box working IPv6 connectivity for all users with a public IPv4 address.

Using 6to4, any site or an end-host with a public IPv4 address can create a unique IPv6 prefix according to the following rules. The globally unique 32-bit IPv4 address is added to the 6to4 prefix (`2002::/16`) registered by IANA. Thus, the resulting prefix is a single `/48` and can be used exactly as any other valid IPv6 prefix. Let us consider the following example. An end-site uses `192.0.2.1` as a public IPv4 address. The IPv4

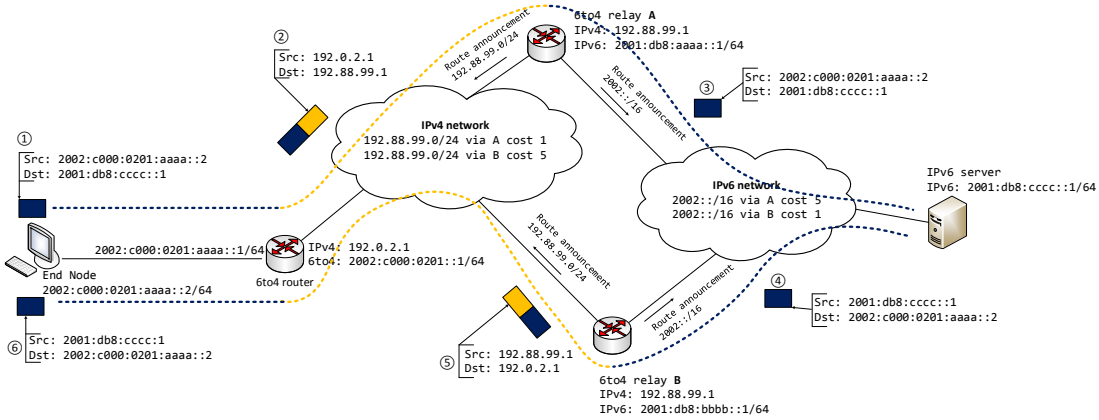


Figure 3.5: 6to4 transition technique

address is combined with the 6to4 prefix, so the resulting prefix is `2002:c000:0201::/48`. The following 16 bits form a site-level aggregation identifier for creating a local addressing hierarchy and identifying a subnet. The last 64 bits are used as EUI (End Unit Identifier)[7] for end nodes in the 6to4 network.

The overall example of 6to4 functionality is depicted in Figure 3.5. If an end-site wants to contact another IPv6 site (step 1), site's 6to4 router encapsulates the IPv6 packet to the IPv4 header and transmits it over IPv4 Internet to a relay router (step 2). The relay router has a native IPv6 connectivity together with 6to4 interface. The relay is responsible for decapsulating the 6to4 traffic and transmitting over its native IPv6 interface (step 3). There is a problem, however. How to obtain relay's IPv4 address? A user can specify it manually, but to avoid the need for static configuration, RFC 3068 offered a solution. It specified an IPv4 anycast address (`192.88.99.1`), thus, providers willing to provide 6to4 service to their clients could advertise the anycast prefix. Any 6to4 router could just send its outbound 6to4 traffic (an IPv6 packet encapsulated in an IPv4 packet) to the anycast address specified in the RFC. Routing protocols ensure that the traffic is routed to a nearest 6to4 relay as is shown in the figure.

The returning traffic must be processed by a relay as well. Here, providers who provide 6to4 service announce 6to4 prefix (`2002::/16`). Similarly to the outbound traffic, routing protocols ensure that returning 6to4 traffic is routed to a nearest 6to4 relay (step 4). The relay must encapsulate the IPv6 response to an IPv4 packet (IPv4 destination address is extracted from the 6to4 address) and send it through IPv4 Internet (step 5). The site's 6to4 router strips IPv4 header and sends the IPv6 packet to the client (step 6).

We have to highlight the fact that inbound and outbound relays do not have to be same. In fact, they are usually different which introduces asymmetry in the packet path as shown in the figure.

6to4 – Rise and fall

What gained 6to4 popularity and why the protocol was (and still is) one of the most used transition mechanism? It is probably because of the widespread support in CPE devices and Windows operating systems (6to4 is supported since Windows Vista). Together with anycast 6to4 relay specified in RFC 3068 it was very easy to obtain an IPv6 connectivity

[7] Sometimes it is also called Interface Identifier (IID) as mentioned in the previous section

for all home users behind a router with a public IPv4 address. However, the 6to4 traffic asymmetry and firewall issues introduced a problem with stability and reliability of 6to4 protocol. Geoff Huston made several measurements showing that approximately 13% of all 6to4 connection toward his measurement site failed [78]. Other studies and analyzes showed similar numbers, e.g., Tore Anderson's study [79] or Emile Aben's study [80]. Due to these problems an advisory and recommendations about 6to4 deployment were published in RFC 6343 [81]. However, the failing rate of 6to4 remained bad. Thus, after several long and heated debates in IETF v6ops working group, anycast prefix for 6to4 relay was deprecated in RFC 7526 [82]. Hence, it is not recommended to include the anycast 6to4 transition mechanism in new implementations. If included in any implementation, the anycast 6to4 mechanism must be disabled by default. It is expected that usage of 6to4 transition technique and amount of 6to4 traffic decrease in future.

There are other mechanisms using similar tunneling method as 6to4, e.g., 6in4, 6rd or 6over4. We do not describe these mechanisms in detail in this thesis. Although there are small differences, e.g., static tunnels instead of dynamic tunnels in 6in4, ISP's IPv6 prefix instead of a generic IPv6 prefix in 6rd, or IPv4 multicast requirements in 6over4, the overall idea is very similar to 6to4 tunneling. If the reader is eager to know more about these protocols, we recommend RFCs with protocols specifications – 6rd is defined in RFC 5569 [83] and RFC 5969 [84], 6over4 is covered by RFC 2529 [85] and 6in4 is defined in RFC 4213 [70].

3.4.2 Teredo

Teredo is a transition technique where IPv6 traffic is tunneled over IPv4 and UDP. The main reason for this approach is overcoming limitations of classic tunneling methods proposed for IPv6 tunneling over IPv4 (6to4, 6in4, etc.). These methods do not work when an end node is behind a NAT device because NATs typically do not allow arbitrary protocols – usually only protocol numbers 1 (ICMP), 6 (TCP) and 17 (UDP) are allowed. Classic tunneling methods use, however, protocol number 41 (IPv6), thus, they are often blocked by NATs or firewalls. Another problem is that a device behind a NAT has a local address that cannot be used in the 6to4 scheme.

Teredo overcomes these limitations and allows IPv6 connectivity for devices behind NATs or firewalls. Teredo is a quite complex transition mechanism as it has to cope with different types of NAT, keep the connection over NAT open, etc. We will highlight only parts of Teredo service protocol that are relevant to this thesis. If the reader wants to familiarize with Teredo in more detail, we can recommend the RFC 4380 [86] or excellent publication by James Hoagland: *The Teredo Protocol: Tunneling Past Network Security and Other Security Implications* [87] or Microsoft's *Teredo Overview* [88].

Teredo mechanism consists of three basic components: clients, relays, and servers. Teredo clients use UDP tunnels over IPv4 for sending and receiving Teredo IPv6 traffic. Teredo relays serve as routers and bridge IPv4 and IPv6 Internets for Teredo clients. Relays accept tunneled traffic from clients, decapsulate it, send it through the native IPv6 connection and vice versa. Teredo servers help clients to set up tunnels to IPv6 nodes. Servers are necessary for determining client's own Teredo address and type of client's NAT. Servers have a dual stacked connection as relays, but do not serve as a general relay (do not pass users traffic). We will only describe an example where a Teredo client is behind a cone NAT (the less restrictive type of NAT – for more information see 4.1). Other type of Teredo communication use Teredo server and relay interaction to open NAT holes, etc.,

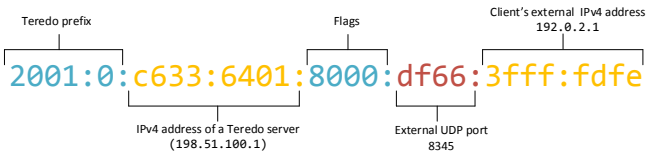


Figure 3.6: Example of Teredo address

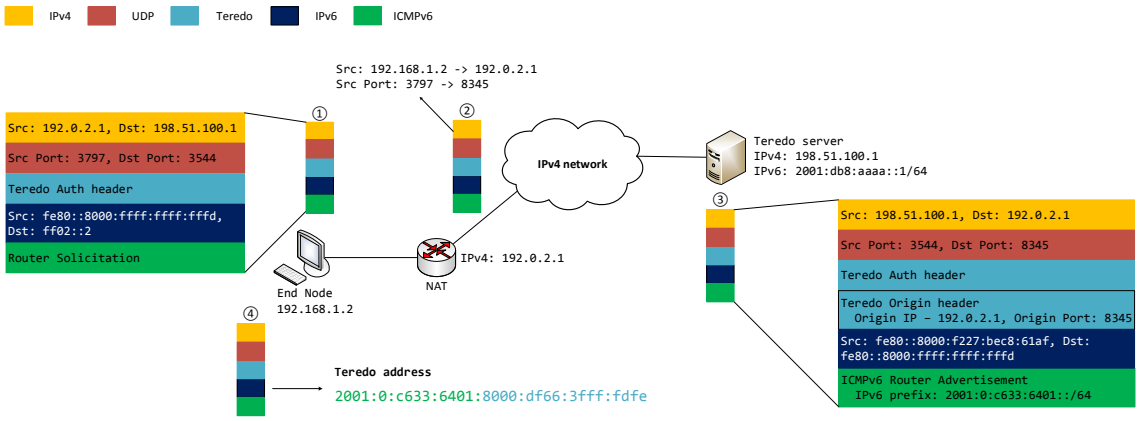


Figure 3.7: Teredo qualification procedure

but these are not relevant to this thesis.

Teredo qualification procedure

If a Teredo client wants to connect to an IPv6 server, the first step the client must do is to determine its Teredo address. The Teredo transition mechanism uses a rather complex set of rules (a qualification procedure), how to find and create an IPv6 Teredo address. The address consists of several parts as depicted in Figure 3.6.

The Teredo prefix (`2001::/32`) is reserved by IANA, flags are used for a NAT type detection and unique/global bits for the Teredo address. An observant reader may notice that UDP port `8345` and IP address `192.0.2.1` do not correspond with values in the Teredo address (`8345` is `0x2099` and `192.0.2.1` is `0xC0000201` actually) because original values are XORed – original UDP port is XORed with `0xFFFF` and original IPv4 address with `0xFFFFFFFF`. The reason for this obfuscation lies in the behavior of some NAT middlewares. Some implementations of NAT examine all packets for „embedded addresses“ present in application payloads. If they detect a bit sequence that looks like a private IPv4 address of the client, they will replace the bits with bits corresponding with an external address. The behavior could help protocols that embed IP addresses inside the application payload (FTP, SIP, etc.), but the approach „rewrite everything“ is sometimes harmful. The obfuscation, thus, protects against this behavior. The overall process for detection and creation of the Teredo address is depicted in Figure 3.7.

How does a client create its Teredo address? The first thing the client have to do is to determine what is a public IPv4 address of its NAT. To find out, the client must initiate a connection to a Teredo server which address is usually preconfigured in client's software. The client sends an ICMPv6 Router Solicitation message tunneled in UDP, Teredo and IPv4 headers to its Teredo server (step 1 in Figure 3.7). The message passes through the

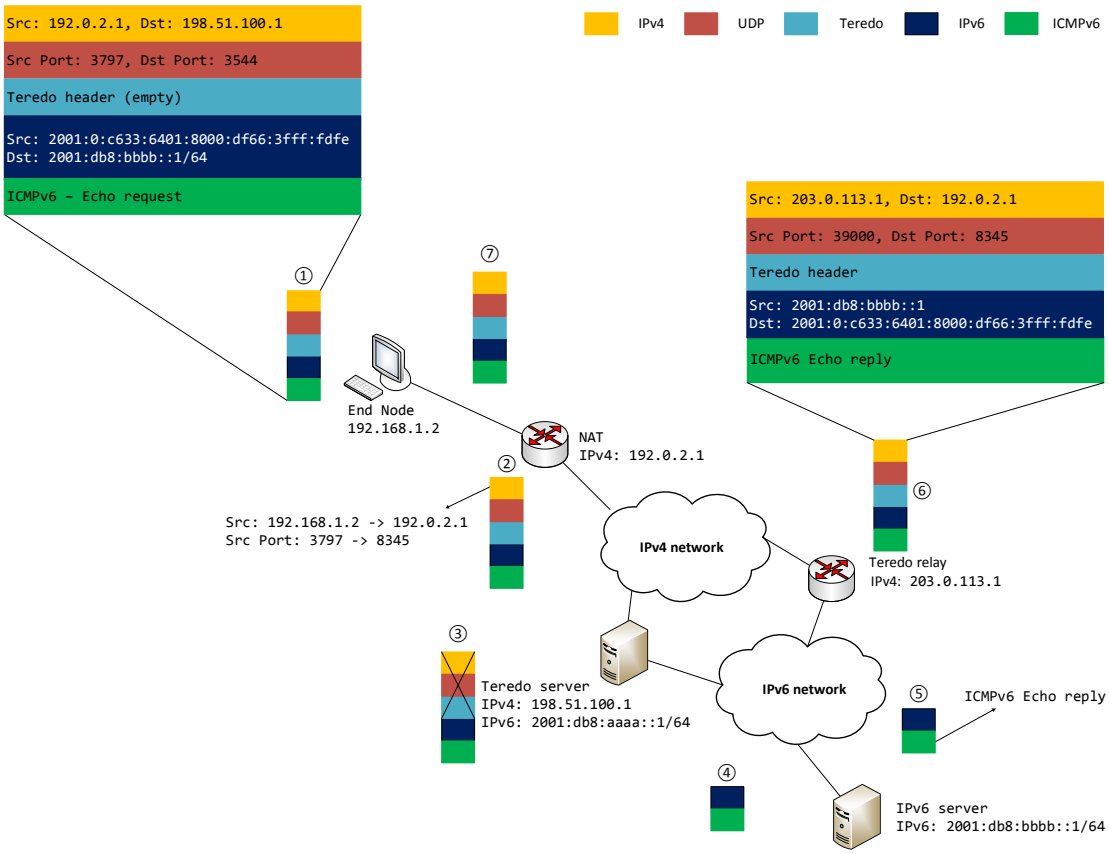


Figure 3.8: Teredo initial communication

NAT. The NAT translates client's private IPv4 address to a public one (step 2) and sends the message to the Teredo server. The server replies with an ICMPv6 Router Advertisement message. The RA message contains an IPv6 prefix that combines Teredo prefix (`2001::/32`) with the IPv4 address of the Teredo server. The server also inserts an Origin data block to the Teredo header. The Origin data block contains client's external IPv4 address (IPv4 address of client's NAT) and an external port number (step 3). Using the IPv6 prefix and data from Teredo header (Origin data block), the client is able to configure it's own Teredo address (step 4).

Teredo Initial communication

After Teredo qualification procedure, the client obtains a Teredo address and is able to contact an IPv6 host. Before the real traffic between a Teredo client and an IPv6 host can happen, the client must first discover the IPv4 address and UDP port of a Teredo relay. The initial communication is depicted in Figure 3.8.

To find the IPv4 address of a Teredo relay, the Teredo client performs a *direct IPv6 connectivity test*. In this test, the client sends an ICMPv6 Echo Request message to the IPv6 node that it would like to contact (step 1 in Figure 3.8). This message is sent to client's Teredo server. Client's NAT translates the IPv4 source address of the packet (step 2) and the Teredo server strips IPv4, UDP and Teredo headers and sends the packet via its native IPv6 connection to the IPv6 node (steps 3 and 4). The IPv6 node responds

with ICMPv6 Echo reply (step 5). Routing protocols ensure that the reply is routed via a nearest Teredo relay. The Teredo relay encapsulates the IPv6 packet with Teredo, UDP and IPv4 headers and sends the packet toward client's NAT that forwards the packet to the client (steps 6 and 7). All subsequent packets sent between the Teredo Client and the IPv6 node take this path via the Teredo relay.

Teredo – Rise and fall

Similarly to 6to4, the Teredo is an auto-tunneling protocol. It means that without a user intervention, Windows OS can create a Teredo interface, setup Teredo address and use it for a connection. However, the Teredo mechanism was always used as a last resort way how to contact an IPv6 node. The reason is that Windows OS does not query the DNS for an IPv6 address if the Teredo address is the only IPv6 address configured on the host. In other words, Teredo is used only for a connection to an IPv6 literal address (e.g., `http://2001:db8::1/index.html`). The performance of the Teredo can be quite poor as it takes a long time to set up a Teredo tunnel. Furthermore, RTT of the Teredo connection is much worse than RTT of the associated IPv4 connection for a lot of Teredo clients as described in Geoff Huston's article *Testing Teredo* [35].

Where the Teredo works well is peer-to-peer protocols. The reason is that for a file sharing, the Teredo protocol increases the number of available peers as it penetrates NAT. Furthermore, file sharing protocols create a connection directly to an IP address (no DNS is involved), thus, the Teredo can be used even in Windows. The performance penalty of the Teredo protocol is eliminated by the nature of P2P protocols – the protocols use a large number of peers and redundant connections.

However, the usage of the Teredo mechanism is slowly decreasing. George Michaelson showed several measurements about Teredo usage in his presentation *The decline and fall of Teredo* at IEPG[8] meeting. He observed a drop in the number of Teredo connection from approximately 100 000 connections per day in January 2013 to only a few connection per day in July 2014. The main reason is that Microsoft stopped their Teredo Relays in 2014 and 2015. The Teredo clients are still able to obtain an Teredo address and use a different relay. However, the Teredo servers preconfigured in Windows OSes (e.g., `teredo.ipv6.microsoft.com`, `win8.ipv6.microsoft.com` or `w10.ipv6.microsoft.com`) has not been available since January 2016. The consequence is that Teredo is not available for Windows clients if a user does not set some other public Teredo server manually.

The only exception is a gaming platform Xbox One. According to Microsoft, Xbox One uses Teredo for increasing the number of available peers [89]. In this case, Teredo is used without relays – only for Teredo-to-Teredo traffic. This approach is less error prone and more stable, thus, Xbox One uses it to increase the number of available peers [90]. This usage is also not visible using APNIC's measurement tools as Teredo is not used for downloading content.

3.4.3 Dual-Stack Lite

Dual-Stack Lite (DS-Lite) is an example of a transition technology that uses IPv4-in-IPv6 tunneling and it is specified in RFC 6333 [71]. The main benefit of DS-Lite approach is that ISPs do not have to deploy dual stack in their core and can use only IPv6 protocol, thus, their OPEX expenses can be decreased. DS-lite requires an additional support on

[8] Internet Engineering and Planning Group

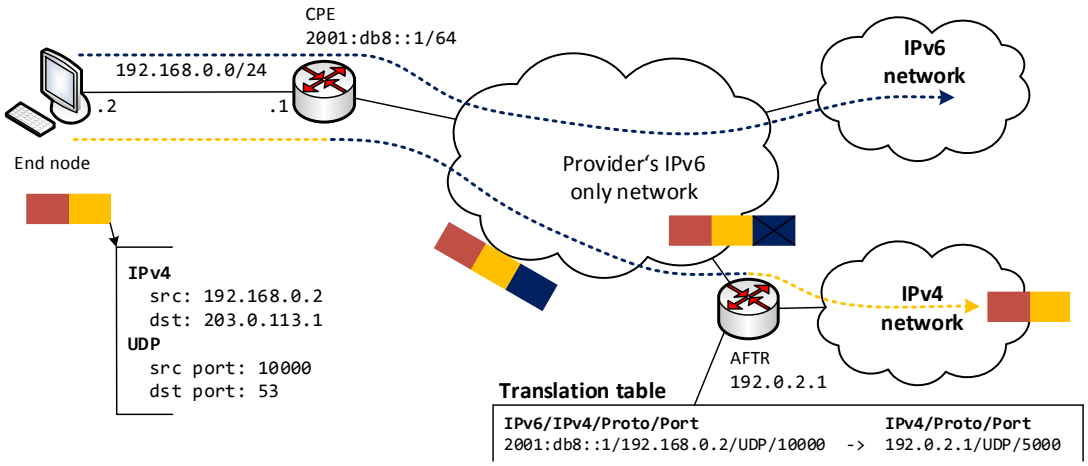


Figure 3.9: DS-Lite communication

user's CPE[9]. The CPE assigns IPv4 addresses for the LAN clients (typically RFC 1918 is used) as well as global IPv6 addresses. However, the CPE is not provisioned with an IPv4 address on its WAN interface and the CPE also does not employ address translation for IPv4 LAN traffic. The CPE uses its global IPv6 connectivity to deliver IPv4 packets to the ISP's AFTR (Address Family Transition Router) device. The overall function of the DS-Lite technology is shown in Figure 3.9.

The CPE encapsulates user's IPv4 traffic into an IPv6 tunnel which is terminated on service provider's AFTR where the IPv4 traffic is translated. We can see in Figure 3.9 that source IPv4 address `192.168.0.1` is translated to global IPv4 address `192.0.2.1` of the AFTR gateway and UDP port 10000 is translated to UDP port 5000 (the translation of UDP port may not happen in every case). Note that the AFTR router must store also an IPv6 address of user's tunnel endpoint (`2001:db8::1`). The reason is that the customers' private addresses overlap in practice, thus the IPv6 address of user's endpoint is used to distinguish customers. DS-Lite benefits are that translation is done only once in provider's network and the network core can be IPv6 only. The drawbacks are the necessity of DS-Lite support on CPE devices, more complicated AFTR (a conventional NAT device cannot be used, the mapping must contain an IPv6 endpoint as well) and scalability. There is no IP aggregation in DS-Lite, thus, if an ISP has one million subscribers, there are one million of tunnel endpoints on AFTR. The ISP can load balance the number of users per several AFTRs, but it can be an expensive solution. The DS-Lite transition mechanism is used by several ISPes around the globe, e.g., Orange Polska (Poland), Orange SK (Slovakia), Forthnet (Greece) or Transix (Japan). There are several operators in Germeny as well.

3.4.4 MAP – Mapping of Address and Port

Similarly to DS-Lite, MAP is a transition technology that uses IPv4-in-IPv6 tunneling. MAP takes an advantage of a larger address space of the IPv6 protocol and uses several bits of the IPv6 address to store necessary information. MAP eliminates a stateful behavior of classical NAT on the ISP border and transports IPv4 traffic over ISP's IPv6-only core. To

[9]The DS-Lite terminology does not use CPE as a name for user's router, but B4 – Basic Bridging BroadBand device. However, we will use CPE in our text.

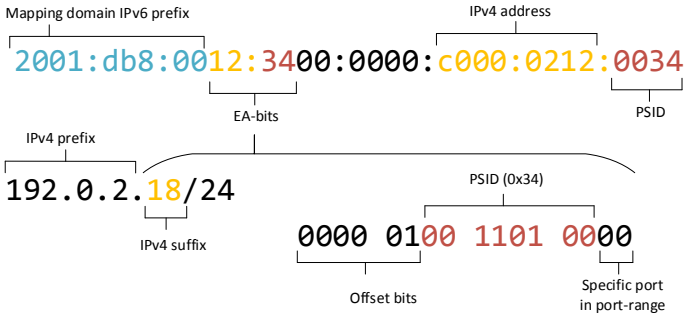


Figure 3.10: MAP – example of an IPv6 address

be able to do that the MAP mechanism requires support on user's CPE. MAP combines an approach from A+P (Address + Port – RFC 6346 [74]) where a single IPv4 addresses is shared among several users without using a stateful NAT. The stateless behavior is guaranteed by an assignment of a unique range of ports to each host. User's CPE device then ensures that outgoing IPv4 traffic has a source port set within user's unique port range. The CPE also assures that IPv4 traffic is sent over IPv6 to ISP's border router. Due to a limited number of ports per user, the MAP mechanism allows accommodating a large number of users behind a shared IP address. The shared ratio is variable and the algorithm supports exclusion of a port range (the first 1024 ports are typically excluded).

The CPE can use either a translation (MAP-T) or encapsulation (MAP-E) approach. MAP-E encapsulates IPv4 traffic in an IPv6 header which contains all necessary information for stateless translation at provider's boarder router. MAP-T does not encapsulate the original IPv4 header with an additional IPv6 header like MAP-E but instead it performs the mapping of the original IPv4 address and port into an IPv6 address and port combination. The structure of an MAP IPv6 address is rather complex. An example is shown in Figure 3.10.

The IPv6 address of MAP CE is `2001:db8:0012:3400:0000:c000:0212:0034`. The address consists of several parts – a provisioned MAP IPv6 prefix (`2001:db8:00/40`), Embedded Address bits (EA-bits) – these bits can have a variable length and they identify an CPE and a port set. The length of EA-bits is 16 in our example (`0x1234`). Shared IPv4 address (`192.0.2.18`) and Port-set ID (PSID - `0x34`) can also be embedded in the IID (Interface Identifier) section. Example in Figure 3.10 shows an IPv6 address of CPE that uses shared IPv4 address `192.0.2.18` and 63 port ranges where each range can contain four ports, e.g., 1232 – 1235, 2256 – 2259, etc. The sharing ratio in this example is 1:256 – 256 IPv4 addresses, 65 536 users, 252 ports for each user (first 1024 ports are reserved and cannot be used by users). CPE can obtain necessary information using, e.g., DHCPv6 protocol or SLAAC. The overall example of MAP-E approach is shown in Figure 3.11.

The CPE with address `2001:db8:0012:3400:0000:c000:0212:0034` ensures that outgoing IPv4 communication will follow the mapping set on the device. Example in Figure 3.11 shows that outgoing IPv4 traffic with source UDP port 10000 is translated to UDP port 1232 (the first port in the first port range). The IPv4 packet is encapsulated in IPv6 protocol and sent through ISP's IPv6-only core. The border router (BR) decapsulates the IPv6 and sends the IPv4 traffic using its native IPv4 connectivity.

The benefit of MAP is stateless ISP's core because BR provides only stateless mapping of incoming IPv4 addresses and ports to IPv6 tunnel. The additional benefit of MAP is

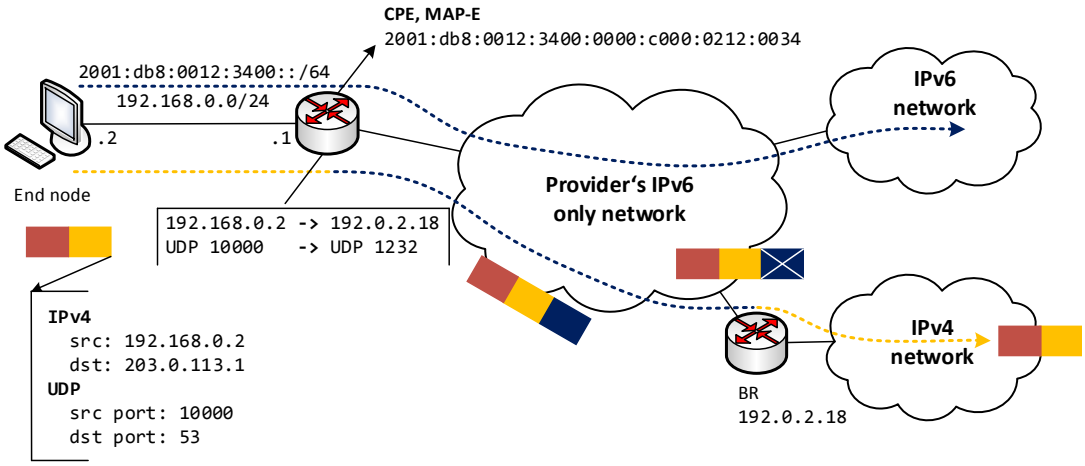


Figure 3.11: MAP-E – example of a communication

a scalability because a hierarchical design is possible. The drawbacks are the necessity of MAP support on CPE devices and inability to transfer other protocols than TCP, UDP or ICMP. The support is also not very widespread among CPE devices. We are not aware of any large MAP deployment in a production network. There is an expired draft *Operational Experience of MAP-E* [91] from JPNE company, but it is not clear for us if the MAP-E deployment is in a production or testing network.

3.5 Challenges in user accounting

Previous sections described the current status of IPv4 accounting and several transition techniques used in today's ISPs networks for migration from IPv4 to IPv6 protocol. This section tries to summarize these techniques and describes the differences between user accounting of IPv4 and IPv6 protocols. We should highlight the fact that requirements for user accounting vary between ISPs. An ISP operating a cable network has different requirements compare to an enterprise or an academic networks. As we have mostly experience with academic, enterprise networks and eyeball ISPs, we focus on these types of networks in the rest of this section. We will, however, alert the reader if there are noticeable differences with other types of network.

3.5.1 Dual-stack

The dual stack approach is currently a recommended approach for IPv6 deployment. There is an ongoing discussion in IETF mailing lists whether IETF should propose a different approach, e.g., an IPv6 only network where the IPv4 protocol is deployed as a service on the top of the IPv6 network. Although there are ISPs that run parts of their networks as IPv6 only – mainly mobile carriers in the US (T-Mobile is known to operate IPv6 only mobile cellular network), there is no consensus so far whether approaches promoting IPv6 only network should be recommended as the default ones.

The dual stack approach requires IPv4 and IPv6 to run simultaneously. We know how to account users on IPv4 and we presented main methods for users accounting in IPv4 in section 3.1. If an ISP deploys dual stack, what will change? Are the current approaches

used for accounting IPv4 users valid? The short answers are „*No*“ or „*It depends.*“. There are several differences in the IPv6 address assignment that obsolete the current techniques for accounting IPv4 users. The main differences are several IPv6 addresses per interface and different address assignment techniques. We will provide an in-depth overview of these issues in the rest of this section.

SLAAC

An ordinary user using current operating system has several IPv6 addresses per device. Let's take a typical SLAAC implementation as an example. User's device always has a link-local address as this address is mandatory. Using Router Solicitation and Router Advertisement messages, the user's device obtains a global IPv6 address. This address is generated according to EUI-64 algorithm or randomly in case of Windows implementation. Even though the address is random, another random address is generated. This address is a temporary address, and it is the address that the system prefers for outbound communication. The address is valid for one day or one week (depends on the system configuration) then the system generates a new one. The older addresses are deprecated, but they can still be used for incoming communication for a while. Thus, a device with a long uptime can use several IPv6 addresses simultaneously.

Note that the older IPv6 RFCs used only single unique address per prefix. A host could have several IPv6 addresses, e.g., link-local, global, etc., but every IPv6 prefix was combined with a host part to form a unique IPv6 address. It was proposed that the host part will be based on EUI-64 or Modified EUI-64 values. The interface can have own EUI-64 identifier, but it is not common. Thus, RFC 4291 introduced a Modified EUI-64 format that is based on interface's MAC address. The vast majority of OSes used this algorithm in practice.

However, leaking information from L2 (MAC address of the interface) to a network address is perceived as privacy and security problems. An attacker able to deploy several monitoring probes across the Internet can track a user when the user moves from one IPv6 network to another because the IID never changes in practice. It is not common that a user swaps network interface cards as it was in the beginning of 1990's. Currently, the NIC is embedded in a device and cannot be removed or changed. The consequence is that the host part remains the same and only the network prefix changes in practice. Based on these observations, privacy and security concerns, EUI-64 and Modified EUI-64 based IID were abandoned and currently operating systems generate IID randomly using Privacy Extensions (RFC 4941), Semantically Opaque IIDs (RFC 7217) or a combination of both. There is currently (beginning of 2016) an ongoing effort to standardize the default IID algorithm in draft *Recommendation on Stable IPv6 Interface Identifiers* [92].

Let us consider the standard installation of Windows and Linux operating systems to better understand the address assignment process. Figure 3.12 shows addresses configured on each system after the first boot. The Windows operating system configures four different IPv6 addresses – a link-local and three global unicast addresses. We can see that neither of these addresses is based on device's MAC address. IPv6 address `2001:67c:1220:80e:e98c:aac:5b07:6d60` is created randomly and never changes. Notice that the IID of the stable random IPv6 address (`e98c:aac:5b07:6d60`) is the same as the IID of the link-local address. Since Windows 7, the Windows operating system will reuse the random interface ID that was generated in the link-local IPv6 address in the Global Unicast and/or ULA addresses. In other words, the same IID will be used if the device is connected to a different network. IPv6 address `2001:67c:1220:80e:bb:a4e9:5e45:c261` was ob-


```
MAC:  08-00-27-6b-2a-ce
Global IPv6 addresses:
     2001:67c:1220:80e:bb:a4e9:5e45:c261   (DHCPv6)
     2001:67c:1220:80e:e98c:aac:5b07:6d60  (Stable private)
     2001:67c:1220:80e:51ec:7373:fdf4:777a (Temporary)
Link-local IPv6 Address:
     fe80::e98c:aac:5b07:6d60%11 (Stable private)
```


```
MAC:  68-b5-99-ea-d4-8a
Global IPv6 addresses:
     2001:67c:1220:80e:98:3d70:925b:e28b   (DHCPv6)
     2001:67c:1220:80e:3f89:4943:87c5:7cc  (RFC 7217)
     2001:67c:1220:80e:ace7:613c:1aff:53ff (Temporary)
Link-local IPv6 Address:
     fe80::6ab5:99ff:feea:d48a (EUI-64)
```

Figure 3.12: Different IPv6 addresses configured on Windows and Linux operating systems

tained from DHCPv6 server and the last one `2001:67c:1220:80e:51ec:7373:fdf4:777a` is a temporary address that changes regularly.

The situation in Linux operating system is different. There is a link-local address and three global unicast addresses as well. Notice, however, that the link-local address is created based on EUI-64 algorithm, thus, the address leaks device's MAC address. IPv6 address `2001:67c:1220:80e:3f89:4943:87c5:7cc` is a random IPv6 address created according to RFC 7217. It means that if the device is connected to a different network (there is a different IPv6 prefix, SSID, etc.), the address will be different. The DHCPv6 address and temporary address have the same meaning as in the Windows example.

We see that different operating systems use different mechanism to create their IPv6 addresses. To find out how operating systems comply with recommendations about generating privacy addresses, we run an analysis in the BUT network to observe the number of unique IPv6 addresses per user. We analyzed an operating systems behavior of approximately 6 000 dual stacked users in years 2013, 2014 and 2015. Results are presented in Figure 3.13.

You can observe the increasing number of IPv6 addresses per user as new updates of OSes obsolete EUI-64 based algorithms and roll Privacy Extensions by default. The average number of unique addresses was 3.62 addresses per user in 2013, 4.01 addresses per user in 2014 and 4.57 addresses per user in 2015. The median remained the same – 3 unique addresses per user. Why is the average increasing and why devices use more and more addresses? Are temporary addresses according to Privacy Extension the only reason for the increase of the number of IPv6 addresses? Yes, it is the primary reason. However, there are other reasons as well. A device having a stable network connection during a day, for example, a desktop computer, will typically not exceed three IPv6 addresses per day (link-local, random and temporary random address). On the other hand, a device that regularly connects and disconnects from the network, generates sometimes a unique IPv6 address for every connection to the network. The increase in the number of addresses is, thus,

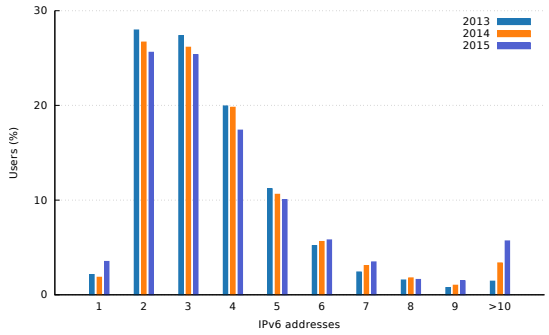


Figure 3.13: Number of IPv6 addresses per user, years 2013, 2014, 2015

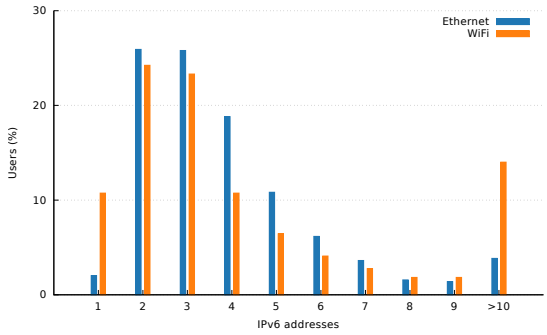


Figure 3.14: Number of IPv6 addresses per user in WiFi and Ethernet networks

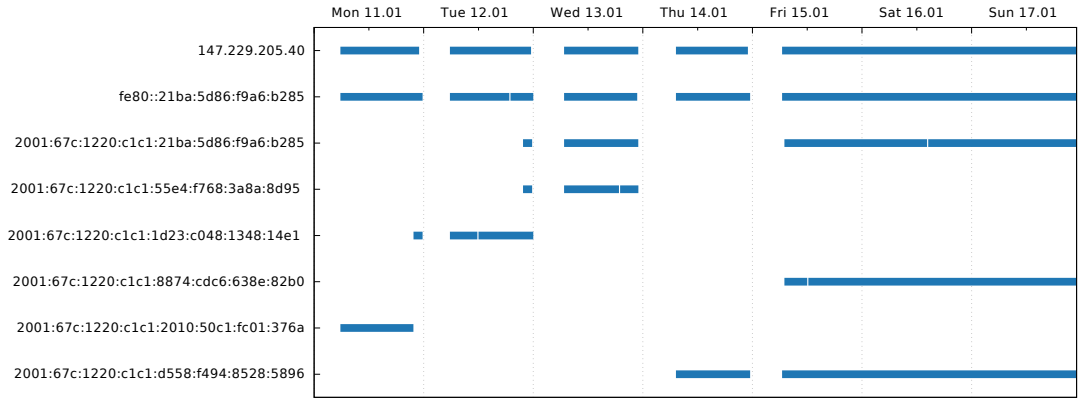


Figure 3.15: IPv6 addresses of a device during period of one week

driven by the rising number of mobile devices in our network - smartphones, tablets, etc. These devices use temporary and random generated addresses by default. The difference between mobile devices and other operating systems is that mobile devices create a random temporary IPv6 address with every association to an access point. We have identified several devices that used more than 100 unique IPv6 addresses per day – the record holds a HTC device with 197 unique IPv6 addresses per day! Figure 3.14 confirms that there are more IPv6 addresses per device in WiFi compared to wired (Ethernet) network.

Figure 3.15 shows a behavior of an operating system on a wired network over a longer period of time. The computer obtained an IPv4 address from DHCPv4 server, generated a link-local IPv6 address and several unique global temporary IPv6 addresses. You can observe that IPv4 and link-local addresses are stable and active during the time when the device was turned on. Global IPv6 addresses were used randomly during the whole week. There were three unique IPv6 addresses simultaneously in use since Friday 15.1.2016. The device used these addresses for the entire weekend for transmitting and receiving traffic.

The behavior of operating systems presented above creates a serious problem for a standard accounting process. If SLAAC is used in a network, the IPv6 address simply **cannot** be used as an identifier of a user's device. However, the problem is that all requests from law enforcement agencies, CSIRT teams, NOC[10] teams or companies complaining about unauthorized distribution of movies, music, etc. cannot use a different identifier

[10]Network Operations Center

than IPv6 address! The reason is simple. They do not have any other information from their side of the connection. They can extend their complaint with port numbers of L4 protocol and time of the incident, but this information is still not sufficient for a network provider to identify the user. We have seen these kinds of issues and complaints and we have received complaints containing IPv6 addresses only. Hence, the accounting process must be updated to reflect these issues.

DHCPv6

Stateless address autoconfiguration introduces several issues for user accounting process. Is there a way how to solve these problems? Why cannot an ISP use DHCPv6 protocol for IPv6 address assignment similarly to IPv4? After all, the DHCPv6 protocol is stateful and it should not be a problem to log all addresses assigned by the DHCPv6 server. This is a valid question. However, there is no easy answer. In the rest of this section, we introduce several reasons why we did not choose DHCPv6 for address assignment in our environment and why it is problematic to use DHCPv6 in general.

One of the problems is that DHCPv6 support is not mandatory for IPv6 end nodes according to RFC 6434 [93]. One could argue that RFC 6434 is an Informational RFC. It means that it does not represent an Internet community consensus or recommendation as RFCs published as standards-track documents (proposed standard, draft standard, standard and Best Current Practice). However, RFC 6434 can be used (and is used in reality) to point out that an IPv6 end node should support this or that behavior. RFC 6434 uses key words syntax (MUST, SHOULD, etc.) from RFC 2119. If RFC 6434 says that an IPv6 end node SHOULD support DHCPv6, it means that there may exist valid reasons for rejecting the support, but these reasons should be very carefully evaluated. What does it mean in practice? There can be devices without DHCPv6 support. If DHCPv6 is the only protocol used for address assignment in a network, there can be devices that will not be able to connect to the networks. Thus, an ISP is not motivated to deploy DHCPv6, as the ISP must run SLAAC anyway. Running two protocols that provide the same thing is an operational burden. It is a real problem.

Android platform does not support DHCPv6 protocol. The feature request for DHCPv6 support on the Android platform was filled in 2012. Unfortunately, Android developers declined to implement the DHCPv6 support [94]. They presented several reasons why they decided not to implement DHCPv6 protocol. Their main objection was that DHCPv6 can limit the number of available IPv6 addresses for a device. If an Android device wants to share the Internet connection (network tethering), the device must have several IPv6 addresses to be able to distinguish between a native IPv6 connection and connection that should be tethered to a different device. The another approach is using IPv6 NAT. Android developers do not want to implement IPv6 NAT, thus, they rely on protocols that can assure several IPv6 addresses for an end node. This can be achieved using stateless autoconfiguration as there is basically „unlimited“ number of addresses for an end node in `/64`. A device can generate an IPv6 address from obtained prefix according to its needs. However, the same is not true for DHCPv6. The DHCPv6 protocol is designed to provide several IPv6 addresses in one request if a device asks for them, but the device does not know what the limit is. Developing future applications can be, thus, cumbersome. Several people objected that the DHCPv6 protocol can allocate an entire IPv6 prefix using DHCPv6 Prefix Delegation, but there is no experience and guidance how an end node should handle an entire IPv6 prefix. So far the end node always operates only with IPv6 addresses and

not with the entire IPv6 prefix. Although these reasons are not perceived as strong reasons for declining DHCPv6 support by many people, the reality is that Android platform does not support it and probably will not support it in the near future neither. It means that all networks without a strict control over connected devices (public hotspots, campus networks, etc.) cannot use DHCPv6 if they do not want to cut off Android devices and upset their users. Considering the fact that Android has a large mobile segment share (80%), it makes the problem even bigger.

There are other reasons why DHCPv6 is not used for address assignment of end nodes. DHCPv6 protocol is not a standalone protocol as in IPv4. It cannot provide all necessary information for an end node to connect to a network. It cannot provide information about a default gateway and IPv6 prefix. Thus, it must run together with NDP protocol. A Router Advertisement message sent by routers is the only way how an end node can obtain information about the default gateway and prefix. Why is there such a requirement? A layer violation. It is believed by several people in the networking community that DHCPv6 protocol, as a protocol of application layer, should not contain information from network layer. Thus, default gateway information or the prefix length information cannot be included in the DHCPv6 message as it would not be a „pristine“ design. Ironically, Router Advertisement message, as a message of network layer, contains information about DNS servers (RFC 6106 [65]) or information about IPv6 address or a domain name of captive portal (RFC 7710 [95]) which are, by definition, services of application layer. The pristine design was, thus, broken in case of NDP, but people will argue that it is still necessary for DHCPv6. There were several proposals by ISPs to soften the refusal of default gateway in the DHCPv6 protocol. All these proposals were rejected. In the end, an ISP must run DHCPv6 protocol together with NDP protocol to assign an address to end nodes. ISPs, however, run their networks for business reasons and maintaining two protocols while there could be only one, increase their OPEX expenses. The practical consequence is that stateless address autoconfiguration is usually used in end networks, e.g., wireless, campus or public WiFi networks. DHCPv6 is used mainly for IPv6 prefix delegation to user's CPE. This topic - DHCPv6 versus Router Advertisement is a reoccurring topic on several networking mailing lists. You can be sure that there will be at least dozen of these discussions in NANOG, 6man or v6ops lists during a year without any outcome. Several people compare these discussions to a religious war – it is about peoples' opinions, not about technical issues the protocols have. I have seen many of these discussions with the same dead end result.

What if all these problems go away? What if Android developers change their minds and implement DHCPv6 client support? What if people in IETF change their minds and allow default gateway and prefix length information in DHCPv6 message? Will it change anything? Maybe, but there is another obstacle in DHCPv6 protocol that hinders the use of the protocol for address assignment – DHCP Unique Identifier (DUID).

There was a strong opinion in DHCP community that identification of an end host by using MAC address of host's NIC is wrong. For example, if a host changes its NIC, it is a different host for DHCPv4 server. Thus, a different identifier was chosen. The DHCP Unique Identifier was designed to be unique across to all DHCP clients and servers, and stable for any specific client or servers. There can be different types of DUID – link layer address + time, vendor specific identifier or link layer address only. In almost all current implementations, the first type is used – a link layer address + a time value when the DUID was generated. DUID is, however, a software value. What is controlled by software is very hard to keep stable in practice and DUID is no exception. We have faced the following

operational problems.

- **Reinstallation:** If a user reinstalls his/her operating system, DUID is changed. This is a problem for address accounting as the accounting system must be notified that a new DUID belongs to the same device.
- **Dual boot:** User can run several operating systems on a device. In this case, every operating system has a different DUID. Again, it is a complication for accounting process.
- **Cloning:** If an operating system is cloned, the DUID remains the same – it is a problem especially in virtual environments, where one virtual machine is cloned from a template. It leads to the situation where DHCPv6 server denies requests for allocating an address to a new virtual machine.
- **Firmware updating:** CPE devices are small embedded boxes with an embedded operating system – usually a Linux. These devices are updated by reflashing an old firmware with a new one. It can lead to generating a new DUID as it is basically a new installation of operating system. CPE will, thus, receive a new IPv6 prefix as DHCPv6 Prefix Delegation is based on DUID identifier.
- **Forward knowledge:** It is currently not possible to determine the DUID for a device beforehand. It is easy to obtain a MAC address of a device, as the MAC address is usually printed as barcode back on the device. The common workflow for configuration of user's CPE in ISPs networks is following: Network administrator reads the MAC address or scans the barcode on CPE device and stores the value to a central configuration system. The system updates DHCP configuration based on CPE's MAC address and the CPE is shipped to a customer. Only thing what a customer have to do is to plug the device into the socket and the device download all necessary settings from DHCP server. This workflow is broken for DHCPv6. The same approach cannot be used for DHCPv6 as DUID is a software based value.
- **Stability:** Several implementations of DHCPv6 clients change DUID value when the device is restarted. Although it is an implementation problem, fixing the problem is sometimes impossible for the ISP as the device is not under its control.

There are arguments that a link layer address of NIC can be easily changed as well which creates the same problems as unstable DUID value. Although it is true that a link layer address can be modified, we believe that the situation is different. If a user has to register his device before using it on the network and the user changes the link layer address, another registration is required. The user is not motivated to do that as he cannot gain any benefit. On the other hand, it is common that users reinstall their laptops, update the operating systems, e.g., from Windows 7 to Windows 10, or use more operating systems. These actions create different DUIDs which trigger a new registration process. However, the device is still the same, thus, the user is only puzzled with the whole situation.

A robust accounting process can, in theory, handle problems with DUID instability, e.g., Interface ID option (similar to Option-82 in DHCPv4) can be inserted in DHCPv6 messages by access switches. However, the feature is usually not supported on current devices. Even if it was, does it mean that DHCPv6 is a valid option if we want user accounting in IPv6 networks? We described in section 3.1, how ISPs account users in IPv4 networks. The

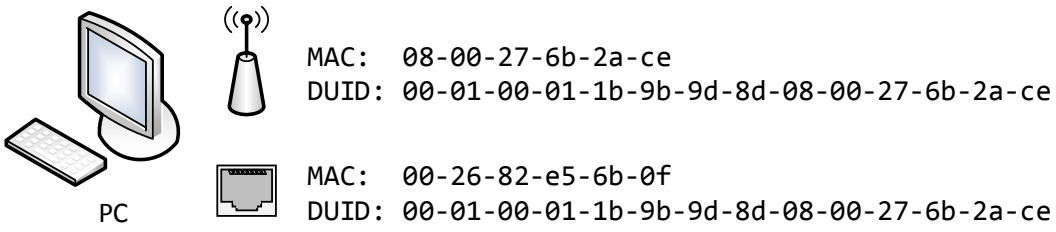


Figure 3.16: DUID and network interface cards

administrator can obtain all the necessary information for user identification (switch port to link layer address to IP address mapping) since the administrator controls all parts of the network and protocols that carry all the necessary information. The benefit is that no-user-to-admin interaction is necessary which is cost effective and convenient for users. The binding information between port – device and IP address is the link layer address of user's device. Can we replicate the same approach with DHCPv6? There are possibilities, but not in today's network with current equipment. Why?

Let us consider Figure 3.16. The figure shows a device with two interfaces – wireless and wired. During installation of an operating system, DUID is created. In the example, the following DUID is created `00-01-00-01-1b-9b-9d-8d-08-00-27-6b-2a-ce` (time + link layer address). One could be in temptation to use the link layer address from the DUID similarly as in IPv4 networks, but it is not a valid approach. RFC 3315 clearly states that clients and servers must treat DUIDs as opaque values and must not interpret the DUID in any other way. Of course, we can break this rule in practice, but it will not help us either because the choice of a network interface for creating DUID is entirely arbitrary. The consequence is that it is not possible to link the information from DHCPv4 server with logs of DHCPv6 server to obtain the full picture of all assigned IP addresses for a device.

Fortunately, there are other approaches. If DHCPv6 server is in the same segment as a client, the server can obtain the link layer address from the Ethernet header. Implementations of this approach exist, but it means there must be a DHCPv6 server in every LAN or there must be a device in the network that can monitor all LANs and VLANs. This approach does not scale very well. The best solution is probably RFC 6939 *Client Link-Layer Address Option in DHCPv6* [96] that adds the necessary information (client link layer address) to DHCPv6 packets. To avoid updating software of all DHCPv6 clients, the link layer option is inserted to the DHCPv6 packet by DHCPv6 relay. Using this approach, user's DHCPv4 and DHCPv6 logs can be tied together based on the link layer address. There is a problem, though. Network equipment, mainly switches acting as DHCPv6 relays, must support RFC 6939. It is, however, not a commonly implemented feature yet.

Several people argue that using a link layer information as an identifier is not a valid approach. They claim that IEEE 802.1X authentication framework with usernames and passwords should be used instead, and all the necessary information for user accounting should be extracted from IEEE 802.1X server logs. 802.1X protocol is an IEEE standard for media-level (Layer 2) access control. It can permit or deny a network access based on an identity of the end user. If it is used in switched networks, it opens a port for a user if user's credentials are valid or keeps the port closed otherwise (similarly in WiFi network where it allows or denies a user access during authentication phase). The address assignment

process is, however, a layer above 802.1X, as 802.1X does not provide a mechanism for IP address assignment. In other words, there must be a DHCP server that assigns an address to the user. 802.1X allows accounting, thus a network access server (a switch in wired networks or an access point in wireless networks) can export information about assigned IPv4 addresses (`Framed-IP-Address` attribute), but it is not supported on all platforms. RFC 6911 defines similarly attributes for IPv6 networks, but implementations are missing in current hardware.

We have described several challenges that ISPs are facing if they decide to assign address with the DHCPv6 protocol. Some of them are problems of implementations, thus, we can expect that they will vanish in future. Other problems emerged because of the different protocol design (different identifier, prefix length and default gateway are not implemented). These issues will remain as there is a strong opinion in DHCP workgroup not to change the underlying protocol behavior. The consequence is that DHCPv6 is currently not a popular address assignment method for the end networks where SLAAC prevails. The DHCPv6 protocol is mainly used for IPv6 prefix delegation where ISPs push entire IPv6 prefix to customer's CPE. There is an ongoing debate considering that the end node (e.g. a laptop) also receives a whole IPv6 prefix, but the discussion is only at the beginning.

3.5.2 Tunneling transition techniques

"Tunnels are evil."

– Geoff Huston, *RIPE 71*

Tunneling techniques are an inevitable part of the transition to the IPv6 protocol. These techniques must be present, and they are used in practice as there is a demand for connecting IPv4 networks over IPv6 and vice versa. There are tunneling protocols as well in the IPv4 world. However, there is one important difference between IPv4 and IPv6 tunneling. Almost every tunneling protocol in the IPv4 world must be set up by a user or a network administrator manually. It is in contradiction to IPv6 because tunneling mechanisms in IPv6 are created automatically. Examples of these techniques are 6to4 or Teredo described in the previous sections.

The consequence is that user's IPv6 traffic can be tunneled inside IPv4 without a user or network admin intervention. This is a problem from a security perspective, as packet filters cannot typically filter tunneled protocols. It is a problem of robustness as tunnels create overhead, decrease the MTU and can cause traffic blackholing. It is an issue of user accounting as well. Standard accounting techniques, such as NetFlow, cannot handle the inner traffic – see previous Figure 3.4. A network administrator loses an insight into the traffic which can cause a problem with backtracking security incidents or data retention requests. If there is an abuse report saying that there was an attack from a 6to4 IPv6 address, the network administrator cannot decide if the abuse message is valid or not. The report consists only of the 6to4 IPv6 address, but there is no such address in captured NetFlow data as NetFlow data do not contain tunneled protocol.

Is the IPv6 tunneling different to tunneling approaches used in IPv4 networks? If a user sets up a VPN, the problem is the same, isn't it? Tunneling in IPv6 is truly very similar, but there are some differences. Firstly, the tunneling mechanisms were developed for promoting IPv6 deployment. Sometimes it is the only way how can an end user obtain IPv6 connectivity. It is, thus, not recommended to block these tunnels. This is different from the IPv4 world as it is much harder to enforce a strict security policy in a network.

The consequence is that ISP cannot block IPv6 tunnels and cannot account inner protocol either.

Secondly, there is a problem with scalability. Although VPNs are common in IPv4, not everybody uses it. However, there could be a large number of users using IPv6 a tunneling approach as these mechanisms are preinstalled on Windows operating systems and active by default. Hence, the accounting solution for IPv6 must be able to cope with a higher amount of users compare to the IPv4 world.

Thirdly, there is still a lot of misunderstandings about IPv6 protocol in ISP and enterprise worlds. Sometimes network admins are not aware of these tunnels. This problem can be solved by proper education and training, however, this is often expensive and according to our experience, not easy at all! The consequence is that IPv6 is deployed in many networks without any monitoring or accounting.

Despite above mentioned drawbacks, there are some benefits. Tunneling techniques often embed IPv4 address in the IPv6 address field. If a network admin has a visibility into the tunneling traffic, the embedded address can be used to correlate IPv6 and IPv4 traffic. Unfortunately, it is often necessary to do it manually as current accounting software cannot handle this correlation.

Some tunneling techniques, e.g., MAP or DS-Lite, must be set up by a network admin, thus, the admin has control over its deployment. Tunneled traffic is then decapsulated on provider's equipment, hence, it can be possible to log the traffic for accounting purpose. Here we often encounter performance limitation as decapsulation and creation of NetFlow records impose a high load on equipment's CPU.

3.6 Solving the challenges in user accounting

How can we solve challenges mentioned in the previous sections? Is it possible to create an accounting system equipped with all the necessary information for user accounting? Is it feasible to deploy the system in a reasonable large network to test the system scalability? This section provides answers to these questions. An accounting system that we developed to solve previously mentioned issues will be presented in next sections. Firstly, we describe how it is possible to handle accounting of tunneling techniques, later we describe how we overcame limitations of current accounting techniques in a dual stacked network. All methods presented in the following sections were tested in a real, production BUT campus network. The core of the network uses 10 and 40 Gbps links, thus, the system is tested at reasonable speeds.

3.6.1 Tunneled traffic accounting

We mentioned several challenges and issues related to tunneled traffic earlier in this chapter. To overcome these issues we developed two solutions. The first one is a hardware accelerated probe that is able to process packets on highly utilized links. The second solution is a software probe with a slightly lower performance that is still able to account packets on BUT core links (if there is a standard Internet packet size distribution[11]).

[11] see Frame Size Distribution at AMS-IX exchange point – https://ams-ix.net/technical/statistics/sflow-stats/frame-size-distribution

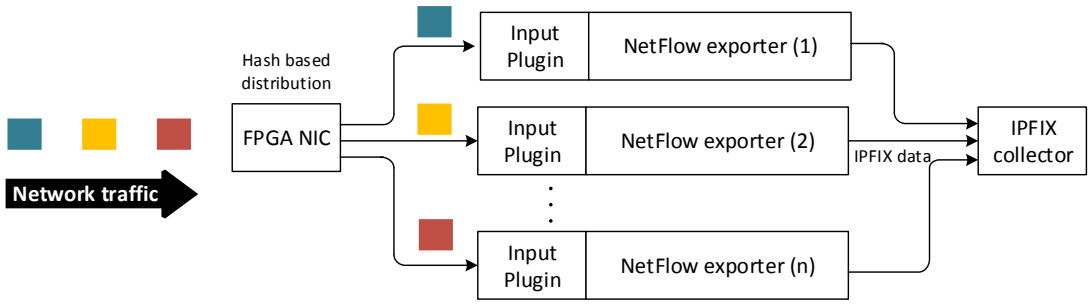


Figure 3.17: Architecture of the probe. Packets are captured by the NIC card and distributed up to 16 exporter instances.

Hardware probe

The hardware probe was developed by our colleagues Martin Elich and Pavel Čeleda. The probe uses a hardware acceleration allowing to parse IPv6 transition tunnels at line rate. These probes were deployed at the CESNET2 network and result published in our paper *Monitoring of Tunneled IPv6 Traffic Using Packet Decapsulation and IPFIX* [97]. The probe is able to achieve high packet processing speed with no need to use packet sampling. The speed is achieved by distribution of packet to different processors.

The architecture of the probe consists of three layers as depicted in Figure 3.17. The first layer is a specialized hardware (NIC with FPGA module). The purpose of the layer is capturing packets and distributing the packets to different instances of flow exporter. An example of a simple distribution method could be a periodic cycle through given number of channels with every received packet. It is a simple and effective way that provides almost even load on all exporter instances. However, the method is not suitable for a flow generation because every instance of the exporter has own flow cache. Distribution of packets among different instances breaks the flow into more noncontinuous flows. This behavior is unwanted, so more advanced distribution, which meets requirements for correct flow export, is needed.

The developed solution uses a packet header parser directly on the NIC card. The parser extracts the following fields for flow identifications: source and destination IP addresses (128 bits per address), source and destination ports (16 bits per port), protocol number (8 bits), IP version (4 bits) and card's input interface. If the field is not present in the packet header, or the header cannot be parsed to find the field, all bits of the field are set to 0. The output of the parsing unit is a sequence of fixed length bits which is passed to a hash unit. The hash unit computes CRC hash with the length of $log_2(number\ of\ channels)$. Each packet is sent to one of the channels according to its hash (the hash is used to address a channel).

The second layer reads packets from the NIC and processes them. The NetFlow exporter is FlowMon Exporter developed by Flowmon Networks. The FlowMon Exporter can export NetFlow and IPFIX data and provides plugin support via defined API. My colleague Martin Elich designed and implemented a plugin for monitoring of IPv6 tunneled traffic. The plugin reads packets from the NIC card and processes them. Each packet is parsed and analyzed until IPv4 or IPv6 header is detected. Otherwise, the packet is discarded. IPv6 packets are passed directly to IPv6 packet parser. All IPv4 packets are processed by the rest of

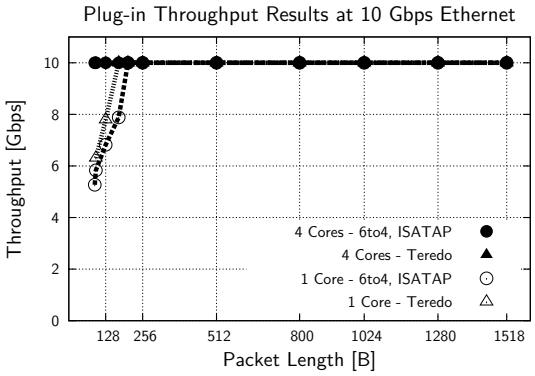


Figure 3.18: Throughput of the plugin at 10 Gbps Ethernet

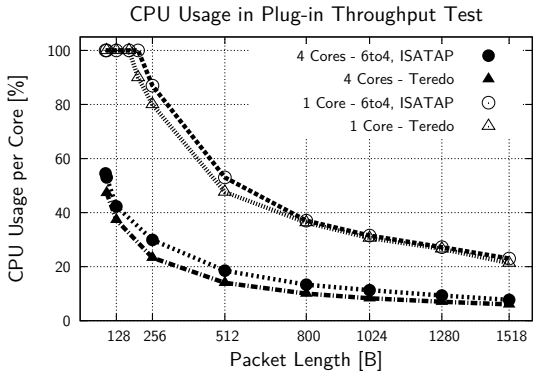


Figure 3.19: CPU usage during the test on single CPU core and multicore system

the filters to detect a presence of tunneling. Detection supports the following tunneling mechanisms: Teredo, 6to4 and ISATAP. The mechanism of the decapsulation is described later in this section.

Performance

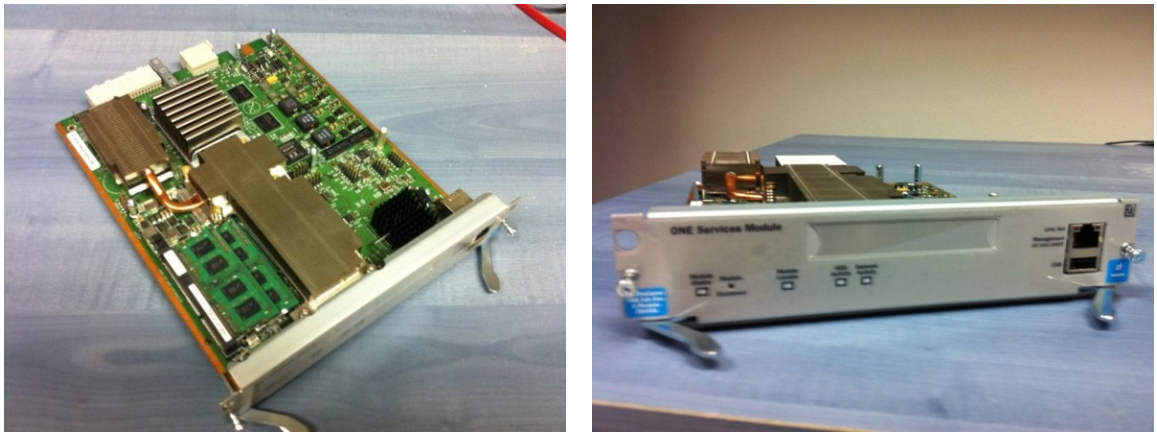
Packet processing performance was measured on 2.0GHz quad-core CPU. We measured the overall throughput of 10 Gbps Ethernet network link and CPU usage. The throughput was measured for Teredo and 6to4 packets. The plugin performance for ISATAP traffic is the same as for 6to4 traffic, as the encapsulation is the same. A single instance of the FlowMon Exporter with loaded plugin was able to process packets with a size larger than 192 bytes at line rate. Four instances of the FlowMon Exporter with loaded input plugin were able to process all packets at line rate with medium to low CPU load on every core. Results are shown in Figures 3.18 and 3.19.

Contribution

The hardware probe was designed by my collegues Martin Elich and Pavel Čeleda. The original idea came from Tomáš Podermański. My contributions in this case were an analysis of captured NetFlow data (50 %), creation of statistics and participation in writing the paper (50 %) [97]. I did the presentation at the Traffic Monitoring and Analysis conference.

Software probe

The hardware probe has an advantage of creating statistics even on a link fully saturated with small packets. There are some disadvantages, though. Firstly, the hash for distribution to different CPU cores must be computed on the card. This can be a problem in some situations, e.g., a packet is fragmented, thus, the card cannot find an upper layer protocol or there is a new L2 technology that is not supported by the card, e.g., VxVLAN or QinQ encapsulations. Secondly, the price of the solution is very high. The network card used in the hardware accelerated probe is much more expensive than ordinary network cards used in servers. We were in the situation where there was a demand for accounting of tunneled traffic on our campus network, but we could not afford several hardware accelerated probes. Fortunately, with the help of our colleagues, we were able to develop a software probe that



(a) Top view **(b)** Side view

Figure 3.20: Software probe deployed on module for HP 5406 switch

is cheaper and provides sufficient performance to monitor user traffic on 10 Gbps links. We have to highlight the fact that the software probe is not able to cope with 10 Gbps link that is fully saturated with small packets. However, this is not an usual case in practice. The average packet size of Internet traffic is much higher – usually around 500 bytes [12]). Thus, we can achieve a sufficient performance just by using a specialized software probe.

The probe can be deployed on a service card module for HP 5406 switch (see Figure 3.20). The module contains two internal 10 Gbps links connected directly to switch's backplane and out of band management port for external configuration and monitoring. The module is de-facto an embedded server, and we were able to deploy a standard Linux distribution on it. As an exporter, we used FlowMon, but any other exporter could be used as well, e.g., nProbe.

A benefit of using a service card that is directly inserted into the switch's backplane is that we can configure the switch to copy traffic from selected ports to the service module. This feature gives us a possibility to monitor even inter VLAN traffic which is usually not available for a standalone probe. We developed an input plugin that is very similar as the one for hardware probe. There are some differences, though. Software probe uses network pseudo devices `rawnetcap` or `pcap` for capturing packets (these pseudo devices are provided by Flowmon NetFlow exporter). The `pcap` device is a standard network pseudo device created using libpcap library. The `rawnetcap` pseudo device is similar to `PF_RING`. It obtains packets directly from a network card. This approach bypasses kernel network stack allowing much higher capture speed with less CPU overhead. On the top of these pseudo devices, we developed an input plugin for the exporter that is able to detect Teredo, 6to4, 6rd, ISATAP and AYIYA (Anything in anything) encapsulations. These mechanisms were chosen as they are used without a network admin intervention. Mechanisms such as MAP, or DS-Lite must be deployed by ISP. The ISP controls the encapsulation/decapsulation which makes accounting easier.

The detection of 6rd, ISATAP and 6to4 tunnels is similar as they use the same encapsulation – IPv4 header is directly followed by IPv6 header. The plugin detects this encapsulation and passes a IPv6 packet to the IPv6 packet parser. The parser decides which tunneling mechanism is used according to the structure of the IPv6 address, e.g., if

[12] https://ams-ix.net/technical/statistics/sflow-stats/frame-size-distribution

the IPv6 address is from the `2002::/16` prefix, 6to4 encapsulation is used, etc.

The detection of AYIYA tunneling is more difficult than the detection of 6to4 and ISATAP traffic. The AYIYA tunneling uses UDP port 5072 in general, but the port can also be different. Furthermore, it is necessary to distinguish between AYIYA traffic and other traffic on port 5072. To detect AYIYA traffic, we use several fields in AYIYA header. Probably the most important is the Epoch Time field. This field is used as a protection against the reply attack, and if the epoch time differs too much between client and server, the tunnel cannot be established. Thus, we can use it as a helper for detection of AYIYA traffic. It is possible to use a precise timestamp on the probe and compare the timestamp with the Epoch Time field. If the AYIYA header is validated and the value of the Epoch Time field fits in the small interval around probe's timestamp, we can pass the next layer protocol in AYIYA header to appropriate parser – usually it is the IPv6 parser. The parser extracts all necessary information about the inner flow and stores it in exporter's flow cache.

The most challenging task is a detection of Teredo traffic as Teredo uses several different headers and arbitrary ports. The initial client communication is with Teredo server and it usually uses UDP port 3544. However, the communication between Teredo client and Teredo relay can flow on arbitrary ports. Furthermore, the detection is limited by the fact that the probe must be stateless – if we store stateful information, there would be a serious performance impact. Thus, we have to process every packet independently. The probe processes every unicast UDP packet trying to detect Teredo encapsulation. We search any of the Teredo specific headers, either Origin, Authentication or their mix, and Teredo specific IPv6 address field. If all headers and fields are consistent with the Teredo specification, we mark the packet as a Teredo packet. The information about the outer protocol is stored as well. The detection method can be perceived as a „Duck test[13]“, but it works in practice.

Using either software or hardware probe, we gain the visibility into tunneling traffic, thus accounting and backtracking of security incidents are quite easy tasks. We created several statistics about the inner traffic – which protocol is used, how the traffic looks like, etc. We did not include this information here as it is not relevant to the thesis. However, if the reader is eager to know more, we can recommend a paper of our colleagues – *An Investigation Into Teredo and 6to4 Transition Mechanisms: Traffic Analysis* [98].

Although usage of tunneling techniques, such as Teredo and 6to4, fall significantly during last few years, the monitor solution is available, and we can adapt it for techniques that can be used in future.

Contribution

I developed the input plugin for the software probe and integrated the probe into the central BUT monitoring system. Data captured by the probe were used several times in presentations at different conferences and lectures by myself or my collegues. The software probe source code was published [99], thus it can be incorporated into the commercial Flowmon probe.

3.6.2 Dual stack accounting

In the previous section, we introduced software and hardware probes. These probes can be used for user accounting even if a tunneling transition mechanism is used. The inner traffic

[13] If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.

can be still mapped to an IPv4 address of the user, thus, an admin can always find who is responsible for the traffic.

What will happen, if an ISP choose to deploy dual stack? In that case, user's traffic flows natively over IPv6 and the admin cannot correlate IPv6 traffic with the user as IPv4 and IPv6 protocols are completely independent. Previous section 3.5.1 summarizes several problems connected with user accounting in dual stack network: different addressing schemes (SLAAC and DHCPv6), temporary privacy addresses, etc. How to solve these issues to be able to account a user?

Firstly, we have to create a protocol agnostic identifier – an identifier which is not tied to any networking protocol. Why? It simplifies the implementation, queries and database. If every user has a unique identifier and all traffic is linked to this identifier, it is convenient for an accounting process to create statistics based on the user's identifier. We can also perceive the protocol agnostic identifier as an abstraction for the accounting process. The accounting process does not care if IPv6, IPv4 or other protocol is used as all these protocols are somehow linked to the identifier. Secondly, we have to create the link or mapping of all user's flows (IPv4, IPv6, ...) to the user's identifier. However, how to create such a mapping? What can be used as a common identifier for both IPv4 and IPv6 flows?

One solution can be to force a client to create such a mapping and publish all necessary information. For example, an ISP admin can demand that user's device must register itself to ISP's DNS by creating a dynamic DNS entry for every IPv4 and IPv6 addresses that the device generates. For example, Windows operating system can register its IPv6 address to enterprise's Active Directory. This solution has, however, several obstacles and can be used only in specific environments. The reasons are the following:

- The solution can only be used in enterprise networks where an admin can force such a requirement. If there is a mix of operating systems or an admin does not have control over end nodes (campus networks, hotel networks, airport networks, etc.), the solution will not work. For example, Windows stations must be enrolled in Active Directory domain to be able to register their global IPv6 address. This cannot be used in campus environments where students use their own devices and different operating systems.

- A specific support on end node is required. A client that is able to register itself to DNS is not available in many default installations of current operating systems (Android, Linux, *BSD, etc.).

- Temporary IPv6 addresses are typically not registered into Dynamic DNS. For example, Windows OS does not register temporary IPv6 addresses and link local IPv6 addresses to Dynamic DNS. Unfortunately, these addresses are precisely the addresses a network admin would like to know who they belong to.

The previous solution has one advantage – the functionality is present in the end clients. It agrees with the end-to-end principle, but unfortunately, there are networks where the solution does not fit. Is there another possibility? Can we find all the necessary information without the necessity to push a particular software to every client? Let us consider Figure 3.21 – a typical scenario how an end node is connected to a network.

The figure shows an end node connected through an access switch to ISP's internal network and through a gateway to the Internet. We can find several pieces of information on different devices. The access switch knows a MAC address and a switchport of the

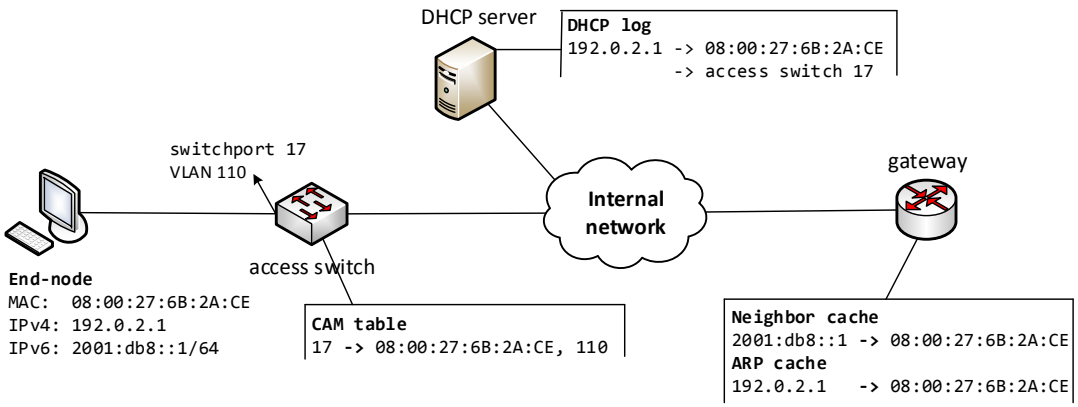


Figure 3.21: A simple network topology showing which information are available in different parts of the network.

connected end node. There could also be information about the IPv4 address if First Hop Security is deployed (e.g., DHCP Snooping or SAVI[14] frameworks). The IPv4 address is also present in logs of a DHCPv4 server. The logs can also contain information about the switchport if the access switch inserts information about it using DHCP Option-82. The gateway holds mapping between IP address and MAC address. These mappings are stored in ARP cache table for IPv4 address and Neighbor cache table for IPv6 addresses.

We can see that information is repeating – mainly MAC address and switchport. Thus, we can use this information to glue necessary pieces together. We should, however, highlight the fact that the MAC address should not be used as the key! A user can change hardware (a common incident) or change its MAC address in operating system (less frequent, but it can happen). Thus, the identifier (UserID) must be different – it can be a random number, user's login or something else. It does not matter what, except that the identifier must be unique in the whole ISP network.

To put all the information together, we can reuse the infrastructure that is used in IPv4 world as described in 3.1. The accounting system must be extended with additional functionality that holds necessary information for IPv6. There should also be a possibility to insert additional information that is network specific, e.g., a Radius login or a PPP identifier.

The data structure for user identification can be the following tuple: *(UserID, Timestamp, L2 address, L3 address)*. If we collect and store this information, we can answer simple questions, e.g., *„Who used this IPv6 address on 1.1.2016?“* or *„Which addresses are connected with user John Doe?“*. Unfortunately, we cannot answer questions like: *Which servers were contacted by the user?* or *Can you confirm or deny that there was a communication between these two users?* This does not fulfill necessary law requirements for data retention, thus, additional information must be collected. Fortunately, we can reuse NetFlow/IPFIX protocol and extend the data structure with all the necessary information for every flow. Using NetFlow we can obtain the following tuple: *(Timestamp, source L3 address, destination L3 address, source port, destination port, protocol)*. If we extend the NetFlow tuple with UserID, we have the required information.

[14]Source Address Validation Improvement

This solution provides all the required information and can be deployed easily as several key components are already set for IPv4 accounting. However, several interesting questions remain. Mainly:

- How to collect all necessary information about IPv4, IPv6 mapping, MAC to switch-port mapping, etc?
- Does it scale? Does the solution have necessary performance?

We will try to answer these questions in the following sections.

Collecting the necessary information

There are several ways how to collect all the information. It depends on the type of the network, management tools that are used in the network, knowledge of admins, etc. We are going to introduce several possible approaches and highlight their benefits and drawbacks. The final solution probably varies with the network requirements, design and devices. A network administrator must choose a method that works for him best.

Intercepting management protocols

One possibility how to obtain a mapping between IPv6 and MAC addresses is to monitor ICMPv6 traffic. This approach was introduced by our colleagues Libor Polčák, Martin Holkovič and Petr Matoušek in their paper *Host Identity Detection in IPv6 Networks* [100]. Their method makes use of Duplicate Address Detection mechanism (DAD) that must be triggered if a device configures a new IPv6 address. The DAD mechanism assures that the address is unique and can be used in the network without a conflict. The DAD validates the uniqueness of an IPv6 address by sending Neighbor Solicitation message – basically, the DAD process asks if someone in the network knows that IPv6 address. If nobody responds to some time, the DAD process marks the IPv6 address as unique. The system proposed by our colleagues, thus, monitors traffic in a LAN and tracks all Neighbor Solicitation messages that carry information about the mapping between IPv6 and MAC addresses. Furthermore, as Neighbor Solicitation messages are sent to a multicast address, the system has to also monitor MLD queries. If a new join to a solicited-node multicast group is captured, the system joins the group as well. This ensures that even if MLD snooping is deployed in the network, the system should be able to intercept Neighbor Solicitation message.

Benefits:

- The system does not require any support on end nodes.
- In ideal conditions, address detection is very fast – there is only a minimal delay. This is convenient if there is a need to use the information immediately, e.g., a firewall rule is created based on the detected IPv6 address, or a lawful interception is initiated to store the traffic of the given IPv6 address.

Drawbacks:

- The system must have a visibility to all VLANs present in the network. This is often a problem even for a medium size ISP who usually has tens or hundreds of VLANs.

- ICMPv6 messages are not reliable. They can be lost, especially in WiFi environment. If a message is lost, the system will not learn the mapping.
- It is not known how the system behaves on a larger network. It was tested only on a rather small network (`/24`).
- If MLD snooping is deployed in the network, there could be race conditions. Even though the system joins the multicast group immediately after the interception of end node's MLD join, the system can still miss the Neighbor Solicitation message. The operating system on the end node sends a MLD join and a DAD request at the same time in most situations as the operating system expects that there will be no conflict (Optimistic DAD, see RFC 4429 [101]). Hence, the monitoring system can miss the Neighbor Solicitation message as system's port is not yet excluded in the MLD table of the switch. This drawback is only present in some networks with particular network equipment. Current switches with recent firmware usually do not provide MLD snooping capabilities for solicited-node addresses and broadcast the traffic instead, thus, the drawback is not important in this case.

SNMP protocol

Another solution for collecting all the necessary information from switches and routers could be SNMP protocol. The SNMP `GetBulkRequest` message can be used to query necessary tables for IPv4 to MAC mapping, IPv6 to MAC mapping and MAC to VLAN mapping. If there are L3 switches or routers with recent firmware and support for RFC 4293 [102], it is possible to collect L3 to L2 mappings only from one SNMP table – `ipNetToPhysical` table. There are other tables necessary to query – `dot1qTpFdbTable` for mapping between VLAN, MAC address and switchport and `dot1qVlanCurrentTable` and `dot1qVlanFdbId` on some switches to find information about VLAN ID. There are several either commercial or open-source software that can be used to gather this information, e.g., NAV[15] or netdisco [16].

Benefits:

- The SNMP protocol is widely supported by network devices.
- Network vendors often add additional info and statistics to SNMP MIBs that are not available otherwise.
- If a recent firmware is used, the network device can send a message with necessary information (a push-based approach), e.g., a SNMP trap or a Syslog message. This eliminates the delay of traditional pull-based approach.

Drawbacks:

- The SNMP protocol sometimes does not work well in large networks. We tested SNMP to download all the necessary information, but the system failed. The main problem is that the implementations of SNMP agent on switches and routers do not perform well when answering SNMP queries. There are various reasons for this behavior. Usually,

[15] https://nav.uninett.no/
[16] https://metacpan.org/pod/App::Netdisco

it is caused by combination of slow CPU on the device and the fact that the response to SNMP request is sorted by OID value. If a large table is queried – which is the case of `ipNetToPhysical` and `dot1qTpFdbTable` tables, the device's CPU is overloaded, and the generating of SNMP responses can affect the normal behavior of the device. Usually, the forwarding and routing processes are not affected, but the control and management planes of the device can respond slowly. If there are other scripts for configuring the device using SNMP, overloading the device has severe consequences. For example, we use a system for automatic ACL configuration, changing the speed of users interfaces or changing the VLAN of a port. The system uses SNMP write access for changing the configuration. When the switch is overloaded by generating large SNMP answers, the configuration is not pushed properly to the device.

- The mapping is obtained with a delay as SNMP is usually used in pull-based mode. The typical period for querying devices differs, but 10 or 15 minutes interval is typically used for large tables. The push-based approach that eliminates the drawback is often not supported (none of our switches or routers support push-based approach).

Custom based tools

We overcame limitations of SNMP protocol and management protocol interception by writing a custom script that uses device's command line interface via SSH protocol. The script connects to the device using SSH credentials, enters necessary commands and downloads the outputs. The script does not burden the CPU as the CLI implementation is much more efficient – the output does not have to be sorted, the device does not have to create SNMP packets, etc. Although we admit that there could be vendors that implement SNMP properly on their devices we had severe problems even with equipment from big vendors. According to discussion on several mailing lists, we are not alone, and others use a similar approach if query for a large amount of information.

Benefits:

- Similarly to SNMP, SSH access is widely supported. The solution, thus, can be deployed even in multi-vendor environment.
- It is effective even with large tables. The device's CPU processor is not overloaded even when we pull several thousands of records.

Drawbacks:

- Implementation is more complicated compared to SNMP protocol. There must be a specific parsing grammar for every vendor as different vendors use different commands and have different output syntax.
- It is still a pull-based approach. The mapping is obtained with a delay. Although it is possible to query devices reasonable often, e.g., every 1-2 minutes, it is still not as efficient as push-based approach.

Time interval consideration

The time dependency of gathering different data is crucial when accessing caches on a device. Caches hold information needed to build dependency between L3 and L2 addresses.

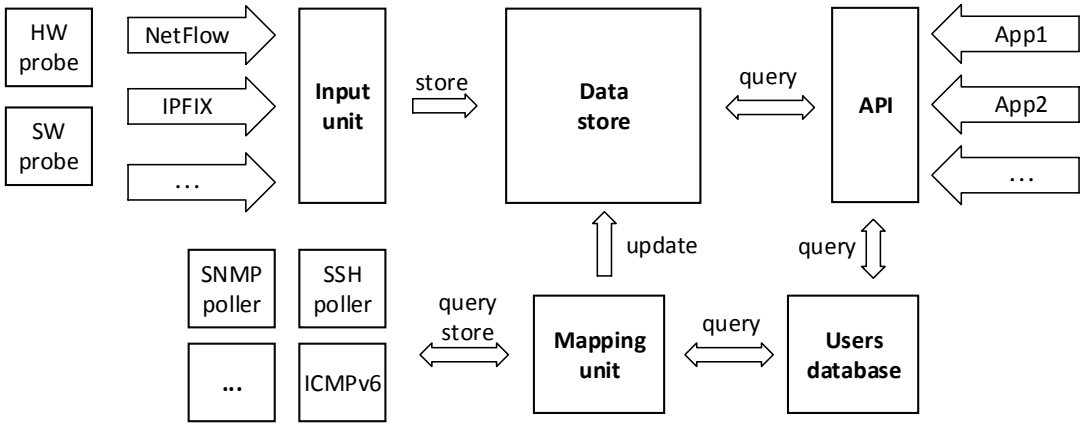


Figure 3.22: Scheme of the accounting system used at BUT network

Since IPv6 addresses change in time and have limited validity, in case an entry is lost, there is no way to detect the mapping. To ensure that all the information is stored properly, the polling interval has to be less that ND cache expiration timeout. Otherwise, some entries from ND cache can expire without being downloaded. Timeout for Neighbor Cache vary between vendors from one hour to several hours, e.g., Cisco and HP use four hours by default. It depends on requirements and use cases. If only what we want is user accounting we can use slightly less than four hours interval. If we wish to have the mapping earlier, the interval must be shorter. We query devices on our network every 15 minutes.

Putting all the information together

Previous sections described which information is necessary and how an administrator can obtain this information. Now, we have to put all the information together to get the overall picture. Figure 3.22 shows an abstract view of the accounting system that is used at our university. The central point of the system is data store. We use a binary data format created by nfdump toolkit[17]. The reason is that the format is more efficient compared to traditional relational databases [103]. Although there are several limitations with the binary format, e.g., a lack of writing capabilities or missing indexes, we will show how it is possible to overcome these problems while maintaining the benefits.

The *Input unit* accepts NetFlow or IPFIX data from several sources and stores them to the data store. Any NetFlow exporter can be used. We used software probes introduced in the previous section to obtain visibility inside IPv6 transition tunnels. The SSH or SNMP pollers are used to provide information for the *Mapping unit*. The *Mapping unit* stores all the necessary information about the mapping between L2 and L3 addresses together with temporal information (start and end timestamps). It can use any standard rational database as a backend for storing this information. We recommend PostgreSQL as it has native support for IP address format (both IPv4 and IPv6). The *Mapping unit* communicates with *Users database* which holds UserID and other information about the user (e.g., name, contact, information about payment, etc.). If necessary, the *Mapping unit* can be combined with *Users database* to a single system.

[17] http://nfdump.sourceforge.net/

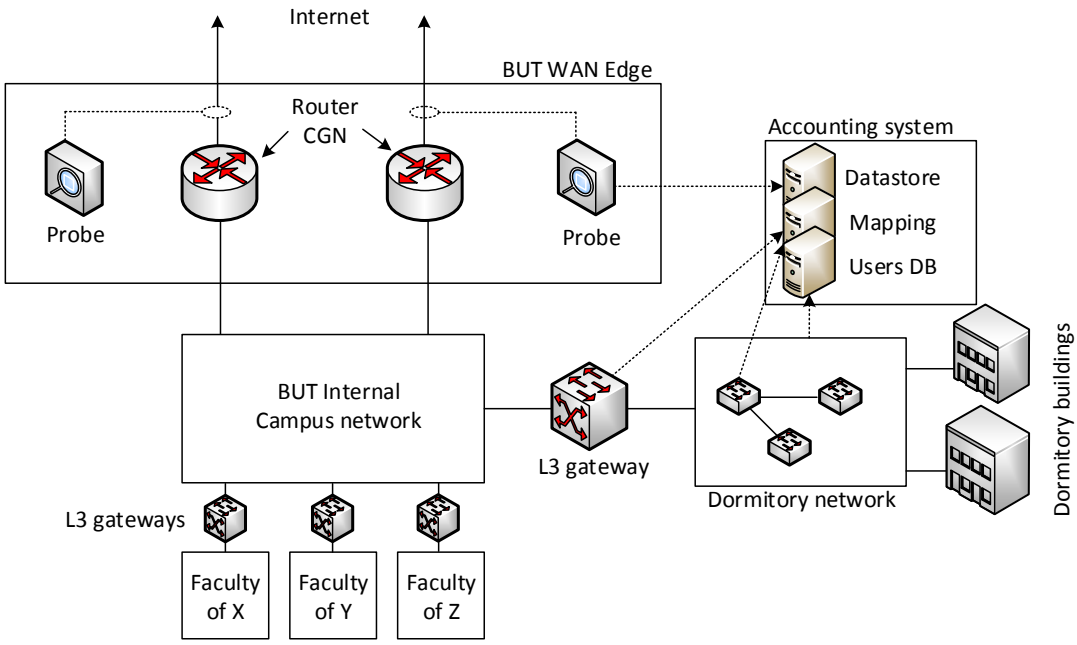


Figure 3.23: Real deployment of the accounting system used at BUT network

The overall process works as follows: The pollers regularly update the mapping database. The *Mapping unit* uses collected information and extends and regularly updates the NetFlow data in the data store. The result is *Extended NetFlow data* structure (the standard NetFlow data extended with additional information). It is possible to query the data for accounting purposes, backtrack security incidents or create data retention reports. Any field can be used for a query, e.g., an application can ask for statistics about specific IPv4/IPv6 address, MAC address, switchport or all traffic belonging to a user based on UserID.

Figure 3.23 depicts an actual deployment of the accounting system at BUT network. The figure is still a simplified view as the BUT network is much more complicated in reality, e.g., we are not presenting the underlying L2 interconnections, and there is also quite complex policy based routing in place. However, it should be clear to see how the system is deployed. We can divide the BUT campus design to different modules. BUT has several faculty buildings in different parts of Brno. The BUT Internal Campus network, thus, spans through city of Brno and interconnects all faculties and research buildings. Each faculty is connected via L3 gateway. There are two primary routers providing connectivity to the Internet using 40 Gb/s uplinks. These uplinks are monitored using NetFlow probes.

To obtain all the information, we monitor each L3 gateway for IP address to MAC address mapping. This mapping information is stored in the mapping database. NetFlow data is stored in the form of binary nfdump format on the central collector. If there is a necessity to monitor users' switchports, we collect the information using SNMP/SSH poller or from DHCP logs. Note that DHCP logs can only be used for IPv4 as there are currently problems with working implementation of RFC 6939 for IPv6 [96]. Although implementation exists, it is available only for selected platforms.

There are scripts that go through the data regularly and update or extend the data as necessary (MAC fields, UserID fields, switchport ID, etc.).

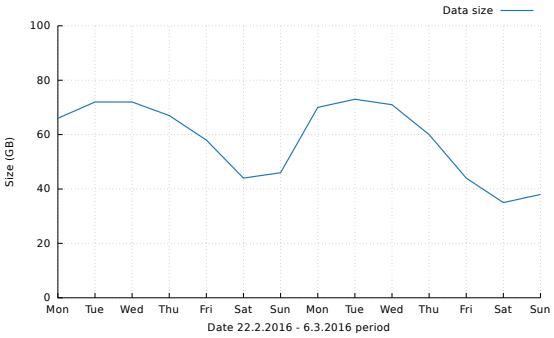


Figure 3.24: Size of NetFlow data in 14 days period

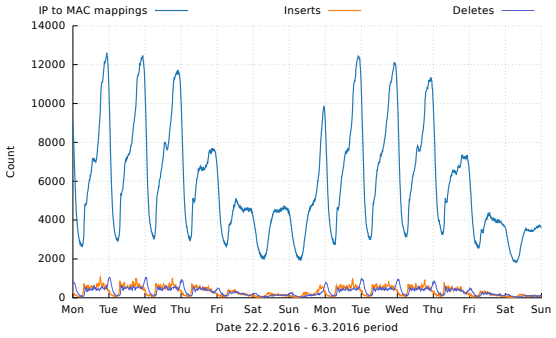


Figure 3.25: Number of open mappings, inserts and deletes during 14 days period

An observant reader having practical experience with NetFlow data processing may wonder, how it is possible to update the binary nfdump format with additional information. We use a special library for this purpose – `libnf`[18]. The `libnf` library has been developed by our colleague Tomáš Podermański. Basically, the library is a shim layer above nfdump sources. It provides necessary abstraction and allows to access each field of nfdump binary record. The benefit of this approach is that it is possible to develop an application that uses the library API. If the underlying data format of nfdump binary structure is changed, the application does not have to be rewritten. It is also possible to add different file format. The `libnf` library provides API for C language. It is also possible to use Perl API[19]. Using the library, the *Mapping unit* is able to go through the NetFlow records stored on a disc and updates all necessary information.

Another question from an observant reader can be: How the UserID is stored? The NetFlow version 9 standard and file format of nfdump do not contain any UserID field. It is true, however, there are several fields available that can be used for the purpose – e.g., `username` field from NSEL[20] extension. The `libnf` library supports writing to the field, thus, we can accept standard NetFlow data and extend it later on the collector. The ideal solution would be to use an IPFIX collector and define an own enterprise ID. Unfortunately, there is a lack of good open source IPFIX collectors. There is a promising `IPFIXcol`[21] [104], thus, the situation can change in future.

System statistics and performance

The amount of data necessary to process is higher than storing only simple NetFlow records. We show, however, that the overhead is not that significant and the accounting system can still run on a commodity hardware. What is the amount of information necessary to store?

Figure 3.24 displays the amount of NetFlow data stored each day. We see that the amount of data peaks around 70GB/day during a week and drops to 40GB/day during weekends. Note that the probes collect traffic twice – before entering to a NAT router and after leaving the NAT router. This is necessary for NAT accounting, and we will show in the next chapter why it is necessary and how the amount of data can be decreased. Figure 3.25 shows a number of open mappings in a temporal database and number of inserts

[18] http://libnf.net
[19] http://search.cpan.org/~tpoder/Net-NfDump/lib/Net/NfDump.pm
[20] NetFlow Secure Event Logging – http://goo.gl/j4l72o
[21] https://github.com/CESNET/ipfixcol

and deletes. We can see that there are around 12 000 active mapping (both IPv4 and IPv6) during the evening. The numbers of inserts and deletes are rather stable – around 1000 during a day, less during night. Note that after the update of a central data store is triggered, the database can be cleared, thus, the size of the database that holds mapping is not important. It is not necessary to keep the data in the database as the mapping information is stored in the extended NetFlow data structure. The overhead depends on the UserID size, e.g., if we choose a 64-bit integer to represent a UserID, the overhead is 4 or 8[22] bytes per flow.

[22] the size of 64 bit integer is implementation defined

3.7 Summary

The first part of this chapter – sections 3.1 and 3.2, described the process of address assignment and user accounting in today's IPv4 networks. We discussed address assignment techniques for IPv6 protocol in section 3.3 and introduced transition techniques (section 3.4) that are used in today's network to overcome incompatibility between IPv4 and IPv6 protocols. Section 3.5 describes challenges that must be addressed if we want to have a robust accounting system both for IPv4 and IPv6 – mainly problems with different address assignment techniques, temporary addresses and automatic IPv6 tunnels. These issues were identified over several years of our experience with IPv6 deployment.

We introduced a system that is able to account users even in dual stacked networks or network with a transition technology deployed. We described hardware and software probes that are able to detect IPv6 transition techniques and account the traffic inside the tunnel. The hardware probe has better performance, the software probe is cheaper and more flexible. Both can be used to monitor 10 Gbps links.

Data from these probes are collected on a central data store and extended with additional information. This information is gathered from various sources, e.g., L3 gateways, server logs or by passive monitoring techniques. We discussed benefits and drawbacks of these solutions as well as the performance of the solution and amount of data necessary to store.

This chapter presented several contributions. Firstly, we believe that detailed description of challenges for user accounting process is important as we are not aware of a publication that covers all these issues together. Secondly, there is a lack of information about the behavior of operating systems in a large network. We covered this topic and present several observations how operating systems behave, how many IPv6 addresses can the network administrator expect, etc. Thirdly, we introduced specialized probes that are able to cope with different transition techniques. The standard NetFlow probes or routers do not provide such functionality. Fourthly, we presented an accounting system that can process and store all this necessary information. The system is application-aware, meaning there is an API that can be used for future applications. The system is built using open source technologies that were extended to provide necessary functionality. It means that it is freely available to everyone and can be deployed cheaply. Furthermore, it is proven that the system can run in rather large and complex network as BUT campus. The system was used several times in practice to track IPv6 security incidents and malevolent users.

There were several papers presenting parts of the system – [97], [10], [105], [106] and [107]. This chapter summarizes these results and presents them in a compact form.

4

Address translation and user accounting

"Any problem in computer science can be solved with another level of indirection."
– David John Wheeler

We discussed several different statistics and measurements of migration from IPv4 to IPv6 protocol in chapter 2. The conclusion of the chapter is that support for IPv6 protocol grows slowly but steady and there are more and more clients connected over IPv6 and requesting IPv6 content. In chapter 3 we solved problems with user accounting in networks where dual stacked or other transition technology is deployed. However, the IPv4 protocol grows as well and data in chapter 2 show that the speed of the IPv4 growth is higher than the growth of IPv6 (e.g. there are more new IPv4 ASN and IPv4 prefixes in BGP DFZ).

How is it possible that IPv4 Internet still grows if RIRs' IPv4 pools are depleted? One of the reason for ongoing growth of IPv4 Internet is that large providers still have IPv4 addresses from previous allocations. As discussed in chapter 2, large fractions of IPv4 address space remain unrouted, and of those address blocks that are routed, again only a fraction is actually in use. These observations are also confirmed in paper *A Primer on IPv4 Scarcity* by Philipp Richter et.al. [108]

Another reason for ongoing growth of IPv4 Internet is that IPv4 markets emerged. The number of IPv4 market transfers skyrocketed in 2015 and unused IPv4 addresses are sold or leased to companies that need them. The IPv4 market seems to work well and it provides access to additional IPv4 resources after the free pool is depleted. Transfers of IPv4 space using market also reallocate number resources more efficiently by moving them out of unused or underutilized allocations toward organizations who need them to grow [109]. Considering the fact that you can lease `/23` for 500€ annually [110], ISPs can easily redistribute the 1€ fee per month per IP address among their customers.

However, the main reason why IPv4 Internet is still able to grow, ignoring the number of available IPv4 addresses, is the Network Address Translation (NAT) technology. NAT allows effective address sharing of one IPv4 address between several clients, thus, it is possible to connect much higher number of devices than the number of available IPv4 addresses. On one hand, the NAT technology is in contrast with the original design of the Internet architecture where each IP address was defined to be globally unique and globally reachable. This is not the case if NAT is deployed in the network as devices uses private addresses and connect to the public Internet via NAT gateways. On the other hand, the current applications can usually overcame the limitation of the NAT technology.

Even though the network translation is currently almost ubiquitous, e.g., 90% of residential customers from a major European ISP use NAT gateways to connect to the Internet [111], residential user accounting has not been an issue for ISPs as they assigned public IPv4 addresses to their customers. It means that every customer had a public IPv4 address and if there was a data retention request demanding information about an IPv4 address, ISP just identified a particular customer for which the IPv4 address was assigned. Unfortunately, the situation is now changing. ISPs are growing in the number of users despite the fact that public IPv4 addresses are scarce resources. This is an unpleasant situation for ISPs as they would like to grow, but they do not have addresses and IPv6 does not help them either as the protocol is still not widely deployed. So far the ISPs solve the problem by adding another level of indirection (another level of NAT). The drawback of the solution is that user accounting is much harder as IPv4 address in a data retention request does not belong to an end user, but to ISP's NAT box. In the rest of this chapter, we show how it is possible to solve this problem.

A historical perspective

We already described the history of transition from IPv4 to IPv6 in chapter 2. We believe, however, that it is interesting to study the history of NAT technology in more detail as the unexpected success and the ubiquitous adoption of NATs had not been foreseen when the idea first emerged.

The first RFC document describing the NAT functionality is RFC 1631 by Paul Francis and Kjeld Egevang [112]. There were, however, several other papers and proposals before the RFC 1631 was published as several researches worked in the same field independently. RFC 1631 was published in 1994 and it is based on Paul Francis and Tony End paper *Extending the IP internet through address reuse* [113] published in 1993. RFC 1631 states that Paul Francis had the concept of address reuse from Van Jacobson, but Paul denied it on his personal webpage [114]. Van Jacobson, however, worked in that field as well and published a draft called *LNAT – Large scale IP via Network Address Translation* in 1992 with very similar technique. There is also RFC 1287 [115] *Towards the Future Internet Architecture* published in 1991 that discussed three possible directions how to extend IP address space. One of the directions presented in RFC 1287 is the following:

„Replace the 32 bit field with a field of the same size but with different meaning. Instead of being globally unique, it would now be unique only within some smaller region (an AD[1] *or an aggregate of ADs). Gateways on the boundary would rewrite the address as the packet crossed the boundary.“*

This is basically the functionality of NAT. In the same year (1991), Paul Francis published paper *The IP Network Address Translator (Nat): Preliminary Design* [116], which is probably the first paper describing the behavior of NAT. The original NAT proposals did not include port translation and was designed only for rewriting IP headers. The port translation included in NAT is, however, the „killer“ feature of the technology as it is an improvement that increased the utility of NAT many times over. The port translation enhancement was probably first invented by Rayan Zachariassen, Glenn Mackintosh and Steve Lamb [114] in 1993. The first public document describing behavior of NAPT is informational RFC 3022 [117].

The original intent of NATs was a short-term response to address exhaustion and all proposals highlighted the fact that NAT breaks certain applications, end-to-end communi-

[1] Administrative domain

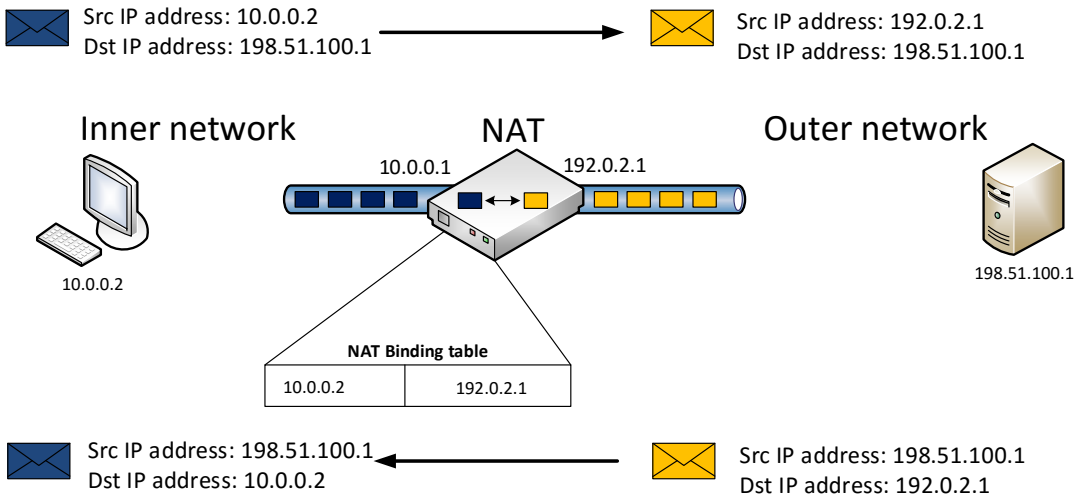


Figure 4.1: Operation of basic NAT: The IP address of the device in the inner network is rewritten to the IP address of the outer network interface

cation, etc. A longer-term solution was seen as the proper way how to handle the explosive growth of IPv4 Internet. The IETF launched IPng working group that led to IPv6 protocol. We already covered the history and outcomes of the IPng working group in the beginning of chapters 2 and 3.

4.1 NAT, NAPT and CGN

We can divide the network address translation technique to three different mechanisms that are used today – Network Address Translation (NAT), Network Address and Port Translation (NAPT) and Carrier Grade Network Address Translation (CGN). These mechanisms are described in the rest of this section.

NAT

The original proposal of NAT as described in the RFC 1631 [112] rewrites IP addresses of a inner network to the IP addresses of an outer network. The behavior is depicted in Figure 4.1. The mapping in the binding table is created either statically by network administrator or dynamically by NAT. The consequence of the IP address rewriting is that the block of inner network addresses can be reused in a different part of a network. This NAT behavior is referred as Traditional NAT or Basic NAT [117].

NAPT

The much more deployed mechanism today is NAPT - Network Address Port Translation, which is the Basic NAT extended with the identifiers from the transport protocol (UDP/TCP ports). Although NAPT is more precise abbreviation for the address and port translation, we will call it NAT in the rest of this chapter. If we refer to address translation without port translation, we will use Basic NAT term. Using IP addresses together with ports allows the NAT device to serve more hosts per one IP address, because a host is

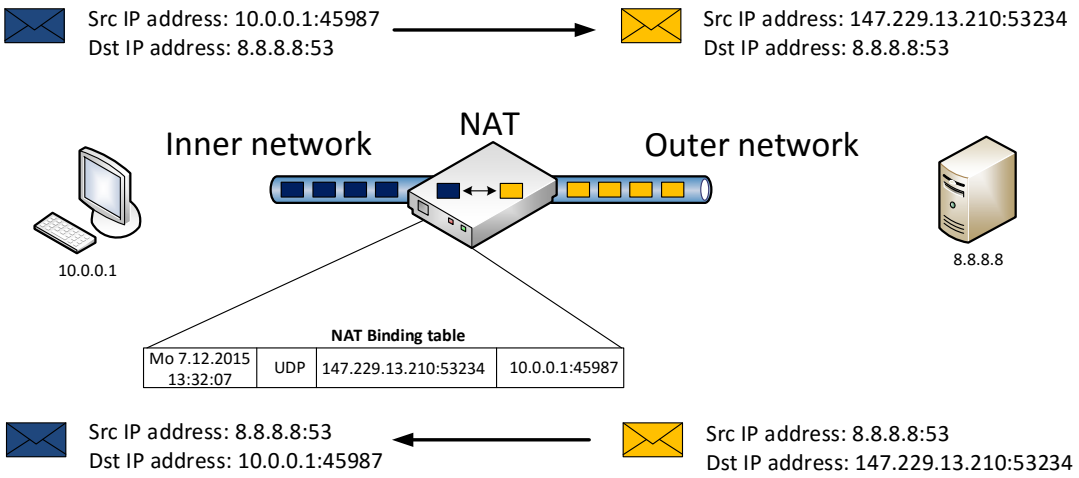


Figure 4.2: Operation of NAPT: The combination of IP addresses and ports are used to create the binding between the inner and outer network

not identified by its IP address, but by a tuple. It depends on NAT implementation, how the tuple is created. Some NAT implementations use only combination of IP address and port, other create a tuple with a protocol type, source and destination addresses, ports and binding time. The main difference between these approaches is the number of users that can be „squeezed“ per one IPv4 address. If only IP address and port are used, the number of clients per IPv4 address must be limited as there are only 65535 available ports (it is less in practice as ports < 1024 are usually not used). Modern NAT implementations use 5-tuple as shown in Figure 4.2. This approach allows to put a very large number of users per one IPv4 address as depleting ports is usually not a problem in this case – there could be 65535 simultaneous connection to the same port on the same destination.

CGN

Carrier Grade NAT, also called Large Scale NAT or NAT444, is an IPv4 address preserve technique used by ISP's to serve IPv4 connection to the clients. CGN technique is deployed by many ISPs. Considering the Czech Republic, CGN is used by a large number of small WISPs[2] that provide wireless Internet access in small towns and rural areas. Large ISPs usually have more IPv4 addresses from previous allocations, thus, the technique is used less frequently. However, the situation changes in these days and even big providers start to deploy CGN today.

The CGN is basically NAT (NAPT) in the ISP network. User's CPE does not get a public IPv4 address, but a private IPv4 address from RFC 1918 or RFC 6598[3] spaces [118, 119]. This cascading of NATs allows ISPs to serve more IPv4 clients with their current IPv4 address allocations. The whole translation process is illustrated in Figure 4.3.

CGN has several benefits to ISPs. The first one is obvious – ISP can provide Internet access to more customers. Furthermore, CGN allows ISP to charge a small fee per public IPv4 address. Although this approach is not popular in the network community, it is

[2]Wireless Internet Service Provider

[3]IPv4 shared address space (`100.64.0.0/10`) delegated by IANA for addressing users' CPE devices

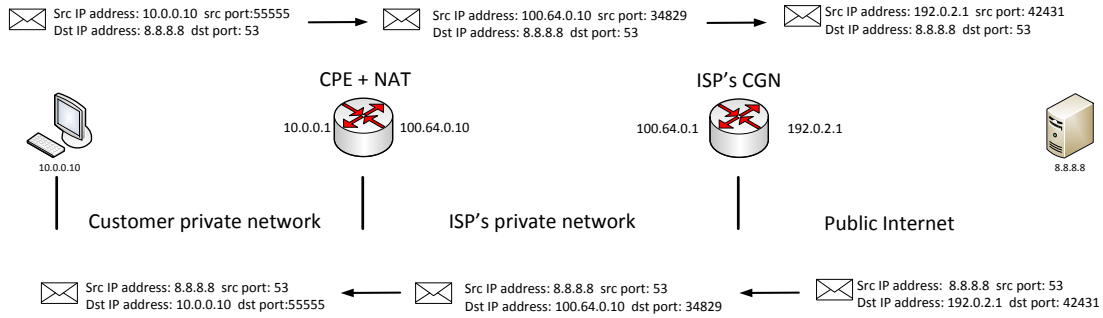


Figure 4.3: Operation of CGN and NAPT. CGN translates public IPv4 address to RFC 6598 address. User's CPE translates between RFC 1918 and RFC 6598 addresses

common. The drawbacks of CGN are similar as drawbacks of NAT. Moreover, the ISP must solve additional drawbacks with asymmetric traffic, load-balancing and redundancy.

CGN can be deployed as a standalone mechanism. However, several ISPs deploy CGN together with dual stack or other IPv6 transition technique. The benefit of this approach is that native IPv6 traffic bypass CGN. Hence, the CGN maintains lower number of stateful sessions and it translates less IPv4 traffic. As described in Chapter 2, around 20 % of traffic can be shifted from CGN by deploying IPv6.

4.1.1 NAT binding behavior

The creating of NAT binding is usually the same across all implementations – the binding is created by incoming interior SYN or UDP packet. Accessing the binding from the outer network and releasing the NAT binding from the binding table is, however, different among implementations. The reason for this lies in the fact that the behavior of NAPT is not standardized. The IETF chose not to standardize NAPT implementation and its operations as IPv6 was perceived as more appropriate and long term solution. The work on the IPv6 protocol, however, took much more time then expected. This, together with the decision of not standardizing of NAPT, led to the current situation where there are several different implementations among vendors as the NAPT functionality was demanded by their customers and IPv6 was not an option.

The different implementation of NAT has led to an attempt to categorize NAT behaviors. The effort was focused mainly to distinguish different ways of accessing NAT bindings from outside networks and different approaches that NAT uses to release the bindings. The terms symmetric NAT, full cone, restricted cone or port-restricted cone first appeared in RFC 3489 [120] that specified STUN protocol (Simple Traversal of UDP Through Network NAT). These terms were further studied in RFC 4787 [121], RFC 5382 [122] and RFC 7858 [123]. These different categories can be summarized as follows:

- **Symmetric NAT:** The binding table stores a local address to a public address mapping and ties the mapping with the destination address for the whole lifetime of the binding. In other words, it is not possible for other host than the destination to send data to the internal host. It is the most restricted model of NAT operation. This behavior is usually default for the TCP protocol as it is similar process how TCP creates a connection.

- **Full cone NAT:** Any exterior IP address and any exterior port can initiate a con-

nection to an internal host if there is a previous mapping for internal IP address and port in NAT binding table. A full-cone NAT behavior is desired especially for UDP VoIP application as it is the least restrictive, thus, any exterior host can dial internal VoIP phone.

- **Restricted cone NAT** and **Port Restricted cone NAT:** These types of NAT add an additional restriction to the behavior of Full cone NAT. A device behind restricted cone NAT is accessible by the destination host only after the binding is created. The destination can use any ports as the NAT binding is restricted only to destination's IP address. Port restricted cone NAT acts as Restricted cone NAT but applies restriction also to ports. Restricted cone NAT accepts connections only from the IP address and port it sent the outbound request to.

Additional categories based on different binding behavior was published in *Anatomy: A Look Inside Network Address Translators* by Geoff Huston [124]. The paper extended the original terms used in STUN protocol and classified the various ways in which NAT bindings and filters were maintained. Relevant part for this thesis is Geoff Huston's classification of port binding behavior. The paper proposed the following terminology:

- **Port iteration:** Every new session receives a new port number. Port numbers start typically from 1024 (lower ports are reserved) and are incremented by one. For example, connection from IP address `10.0.0.1` and internal port `54321` is translated to IP address `192.0.2.1` (public IP address of NAT interface) and port `1024`. The next connection that uses port `12345` is rewritten to port `1025` etc. Example of this approach is NAT implementation on Cisco IOS devices. This behavior is not described in Geoff Huston's paper, but it is quite common, thus we extended the terminology.

- **Port preservation:** NAT attempts to preserve the local port number, if possible. It means that if there is a connection from internal IP address `10.0.0.1` and internal port `54321`, the NAT will try to use the same port for external mapping. If there are two connection with the same port, NAT preserves port for one connection and use a different port for the second connection.

- **Port overloading:** NAT uses port preservation at all times. The consequence is that a different host, establishing a new connection with a port that is already been used in binding, will usurp the existing binding.

- **Port multiplexing:** NAT attempts to preserve the port number as in the port preservation example. The difference is the whole five tuple is used which increases the chance that the source port will be preserved during the translation. If the NAT is using a single external address and two internal clients initiate a connection with the same source port to two distinct servers, the external view is two packets with the same external address and source port. This is possible because the destination IP address is different. If there are two internal connections with the same source IP addresses and ports going to the same destination and port, the NAT must rewrite source port of one of the sessions.

Different approaches are also used for releasing the binding from NAT binding table. NAT cannot keep the binding between internal and external addresses indefinitely as these bindings would consume large amount of resources. Thus, NAT releases the binding if there

are no further packets that use the binding within a certain time period. The consequence is that a local application running on a local host have to use some form of keep-alive operation to maintain a NAT binding open. This is especially true for UDP traffic where UDP based protocols, such as for audio and video streaming, routinely send UDP keep-alive packets roughly every 15 seconds [125]. If the transport protocol is stateful, such as TCP, the binding is based on a transport session. The binding timer is refreshed based on the transport session state and releasing of the binding is based on packets with `RST` or `FIN` flags set. However, NAT can use timer even for maintaining the bindings created by a stateful protocol. The main reason is a protection against Denial of Service attacks (an attacker can forge TCP packet with `RST` flag to release NAT bindings) and for waiting for out-of-order packets – there is a timeout after receiving a packet with `RST` or `FIN` set to double the maximum segment lifetime in the case of TCP. This timeout ensures that there are not any packets on the wire that belong to the binding before the NAT is allowed to release the binding. The following table summarizes the different approaches of creating, releasing and accessing bindings.

Table 4.1: Design parameters of NAT - bindings [2]

	TCP	**UDP**
creating NAT binding	interior SYN packet	interior UDP packet
accessing NAT binding	symmetric	symmetric full cone restricted cone port-restricted cone
releasing NAT binding	timer interior RST or FIN exterior RST or FIN	timer

4.2 Problem statement – NAT accounting

In the previous chapter, section 3.1 describes necessary information for accounting in IPv4 networks. The section discusses the reasons why providers store different mappings and information about assigned IP addresses. The following points briefly repeat information presented in 3.1:

- **IP address:** The IP address is stored and used as an identifier from the data retention point of view [50] as well as from the ISP management point of view (information necessary for billing, etc.). ISP typically stores information about a contract with a customer, together with the assigned IP address and other information in relational database.
- **Timestamp to IP address mapping:** The mapping between timestamp and IP address is stored as well if the ISP provides dynamic address assignment. Typically, logs from DHCP server (if the address is assigned using DHCP protocol) or BRAS server (in case PPP protocol is used) are stored.
- **Network layer to data link layer mapping:** The mapping between network and data link layer is necessary as well if dynamic address assignment is used. It depends

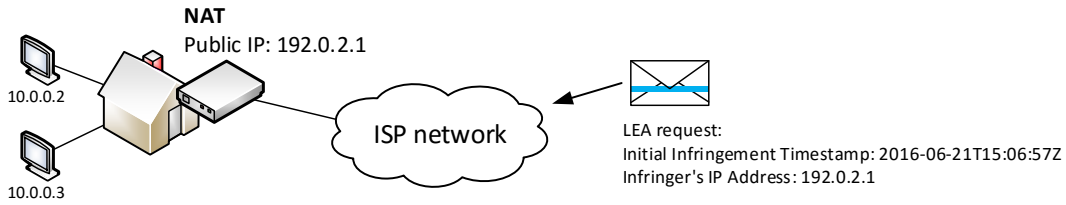


Figure 4.4: Lawful Enforcement Agency requests data originally generated by IP address `10.0.0.1`, but the only piece of information available from outside point of view is public IPv4 address of user's NAT.

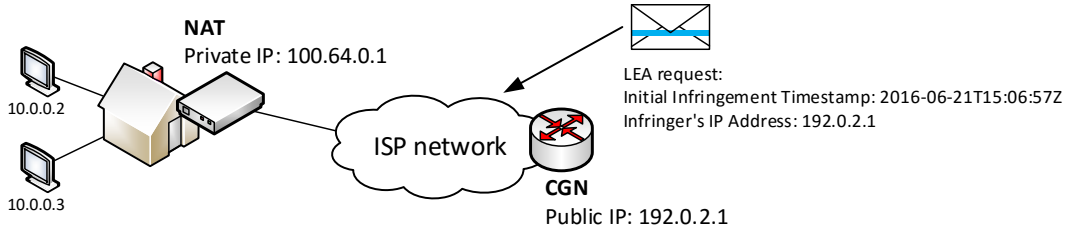


Figure 4.5: Lawful Enforcement Agency requests data originally generated by IP address `10.0.0.1`, but the only piece of information available from outside point of view is the public IPv4 address of provider's CGN.

on a design of an ISP network – it can be the mapping between the IP address and MAC address of customer's CPE, logs from RADIUS server or session-id or a username in case of PPP, PPPoe or PPPoA protocols.

All these pieces of information are, however, insufficient if some form of address translation is deployed in the network. Why is it a problem if the beginning of this chapter states that NAT is prevalent technique for address sharing in current networks? NAT is truly very commonly deployed, but mainly at the edge of ISP network – see Figure 4.4.

If the ISP receives a data retention request, the ISP is able to pair IPv4 address (`192.0.2.1`) in the request with the user that signed a contract with the ISP. It does not matter too much in practice that there are several computers/users connected in the end network as the customer who signed the contract with the ISP is seen as responsible for the whole traffic. From ISP point of view, it is LEA's responsibility to correctly identify a real device or a user from the captured networking metadata that ISP provides.

If the CGN is deployed in provider's network, the IP address in data retention requests is, however, maintained by the ISP as it is the address of CGN public interface – see Figure 4.5. ISP, however, cannot share all data generated by the CGN public interface as there are several customers behind CGN and sharing the date would break their privacy. Thus, there must be some kind of logging option present in ISP network to allow pairing between the public address of CGN with the private address assigned to the customer.

4.2.1 NAT logging

There are several options how to enable logging in networks where NAT or CGN are deployed. Typically, these options require support on NAT device itself. The requirements for a NAT logging depend on NAT's allocation strategy and maintaining of bindings. If NAT is pre-configured and static assignments are used, it is enough to store the NAT configuration only. Typical deployment is, however, NAT that use dynamic assignments. This approach requires more details in the logs as every NAT event needs to be logged. An event in a NAT device can be viewed as a state transition, e.g., the creation and deletion of NAT sessions and bindings can be seen as examples of NAT events as they result in resources (addresses and ports) being allocated or freed. There is also a need for a protocol to transport logs to a central logging server. Two protocols are mostly used – Syslog and NetFlow.

Syslog

NAT device can support logging via Syslog protocol. Every NAT translation created on the NAT device is then logged and sent to the Syslog server. The format is not standardized, thus, every vendor can export different information. To solve this issue, there is a standardization effort in BEHAVE workgroup [126]. The following example shows the Syslog export from a Cisco IOS router where NAT translation is enabled.

```
13:16:10.543: %IPNAT-6-NAT_CREATED: Created tcp 10.0.0.1:23800 100.64.0.1:1024 192.0.2.1:80 192.0.2.1:80
13:17:52.251: %IPNAT-6-NAT_DELETED: Deleted tcp 10.0.0.1:23800 100.64.0.1:1024 192.0.2.1:80 192.0.2.1:80
```

The format of the log is the following: the timestamp, a type of NAT event (created/deleted), an inside IP address and port, an outside address of NAT device (IP address after translation) and the destination IP addresses – in Cisco terminology the outside local and outside global addresses. Thus, the example can be read as a attempt of device with IP address `10.0.0.1` to connect to a server with IP address `192.0.2.1` using HTTP protocol. The IP address `10.0.0.1` is translated to IP address `100.64.0.1`. Notice, that the original port (23800) is translated to 1024. This behavior can be seen as an example of creation of the binding using port iteration approach as described in the previous section.

Using Syslog for logging has some limitations as described in the following list.

- Syslog uses a non-structured text-based protocol – the message can be parsed by a parser that uses a regular expression. If the format of the message is changed, the parser must, however, be reimplemented.
- Two events (creating and deleting of the session) are logged for every session. It means that ISP must retain large amount of data. It can lead to overwhelming of Syslog server.
- Implementation of the Syslog logging uses CPU of the NAT device. If there are many sessions, there could be a large operational impact of the NAT as the CPU is overloaded by generating Syslog messages.

NetFlow extensions – NEL, NSEL and bulk port allocation

Flow based monitoring is a widespread method to store metadata about network communication. It is used both for network management and data retention purposes and can be

```
▽ Flow 1
    SrcAddr: 100.64.10.1 (100.64.10.1)
    Post NAT Source IPv4 Address: 147.229.64.10 (147.229.64.10)
    DstAddr: 8.8.8.8 (8.8.8.8)
    Post NAT Destination IPv4 Address: 8.8.8.8 (8.8.8.8)
    SrcPort: 5
    Post NAPT Source Transport Port: 1
    DstPort: 1
    Post NAPT Destination Transport Port: 1
    Ingress VRFID: 0
    Protocol: 1
    Nat Event: 1
    Observation Time Milliseconds: Jun  7, 2015 20:17:52.000000000 CEST
  Padding (2 bytes)
```

Figure 4.6: Export NEL logging information from Cisco CSR router.

used for NAT accounting as well. NAT logging based on NetFlow or IPFIX uses binary encoding, thus, it is more efficient compare to Syslog, especially in environments where there is a high volume of sessions. Unfortunately, there is currently no standard IPFIX or NetFlow template for encoding NAT binding options. Similarly to Syslog, BEHAVE workgroup in IETF has tried to standardize IPFIX template, but it is still a work in progress [127]. There are, however, several vendor proprietary extensions used in production networks for NAT logging. Cisco NEL (NetFlow Event Logging, sometimes called HSL – High-Speed Logging) or NSEL (NetFlow Security Event Logging) can be examples of such extensions. Figure 4.6 shows information exported using Cisco NEL from Cisco CSR Virtual router.

We can interpret the exported data as follows. CGN created a NAT binding for a connection from private address `100.64.10.1` to Google public DNS server with the address `8.8.8.8`. The private address was translated to the public IP address `147.229.64.10`, the protocol was ICMP – a simple ping request. Note that there is no information about the number of packets or transferred bytes, thus, data exported using NEL must be combined together with classical NetFlow data that are exported simultaneously. The following example of NetFlow records shows both NEL data (first two rows) and classic NetFlow data (rows three and four) as seen on a collector.

Date first seen	Event	Proto	Src IP Addr:Port		Dst IP Addr:Port	X-Src IP Addr:Port		X-Dst IP Addr:Port	Bytes
18:17:52.398	CREATE	ICMP	10.10.10.1:5	->	8.8.8.8:0.1	192.168.1.10:1	->	8.8.8.8:1	0
18:18:56.343	DELETE	ICMP	10.10.10.1:5	->	8.8.8.8:0.1	192.168.1.10:1	->	8.8.8.8:1	0
18:17:52.398	INVALID	ICMP	192.168.1.10:0	->	8.8.8.8:8.0	0.0.0.0:0	->	0.0.0.0:0	500
18:17:52.765	INVALID	ICMP	8.8.8.8:0	->	192.168.1.10:0.0	0.0.0.0:0	->	0.0.0.0:0	460

The previous examples of logging NAT events using NetFlow protocol can be used both for a classical NAT or CGN. Some CGN implementations offer another NetFlow extension – a bulk port allocation and bulk logging. These features allocate a block of ports for translation instead of allocating individual ports. The bulk port allocation works as follows:

1. The client initiates a connection through the CGN. The CGN device allocates multiple global ports of a single global IP address instead of a single global IP address and global port. The default number is 512 global ports.
2. CGN exports information about the allocated block using NetFlow protocol.
3. A new connection initiated from the same client use a free port from the previous bulk allocation. CGN does not log any information about the connection.
4. Based on the volume of translations, additional blocks of ports can be allocated.

```
▽ FlowSet 1
    FlowSet Id: (Data) (263)
    FlowSet Length: 32
  ▽ Flow 1
      SrcAddr: 100.64.10.4 (100.64.10.4)
      Post NAT Source IPv4 Address: 147.229.64.100 (147.229.64.100)
      Ingress VRFID: 0
      Protocol: 6
      Nat Event: 1
      Observation Time Milliseconds: Jun 16, 2015 15:52:43.674000000 CEST
      Port block start: 1024
      Port block step size: 8
      Number of ports in block: 512
```

Figure 4.7: Bulk logging using NetFlow protocol exported from Cisco ASR router.

To increase the security, the Bulk Port Block Allocation feature does not allocate ports sequentially but use a scattered port-set method. CGN uses a starting port number (e.g., 2000), a step value (e.g., 8) and the number of ports to allocate (e.g, 512). The starting port number is used for the first connection of a client. CGN adds the step value to the starting port number to obtain a port for another connection for the client. In our case, another connection will used port 2008. The whole process repeats until all 512 ports are allocated. Using the bulk port allocation with the bulk logging decreases the amount of data exported by CGN to a log server. The whole process is illustrated in Figure 4.7. We can see that a device with IP address `100.64.10.4` tries to established a connection through a CGN. The connection is translated to public IP address `147.229.64.100` and CGN allocates 512 ports started with port 1024 for the client. The port step size is 8. If the device does not create more than 512 simultaneous connections, only one log event is exported to a central collector. A similar approach is described in RFC 7422 [128]. The difference is that the RFC uses an approach where ports are presets in the CGN configuration, thus, CGN does not export any logging information.

4.3 A new approach for NAT accounting

The previous section describes several issues that are introduced by CGN or NAT in a network. We described several approaches that can be used for logging all necessary events. However, several unresolved issues remain. Firstly, all approaches for logging of NAT events require support directly on the NAT device. If the NAT device does not support Syslog, NEL, NSEL or bulk port logging, there is currently no way how to obtain all the necessary information. Secondly, logging features are CPU sensitive. If there is a large number of sessions, the CPU can be overwhelmed by the logging process. Thirdly, there is no easy way how to correlate exported binding data from the NAT with NetFlow data exported by a standalone NetFlow probe. Let us consider the following example in Figure 4.8 to better understand the last issue.

Network administrator is able to configure both NAT logging and NetFlow accounting to export the data to the same collector. However, there is no easy way how to query the collected data. If there is a LEA request or the network administrator tries to backtrack a security incident of an IP address, NAT logs must be queried first to obtain information about the address translation (step 1). Equipped with the information, the network administrator can query NetFlow records to obtain actual connections for the IP address (step 2). A large amount of records must be processed to obtain all the necessary information.

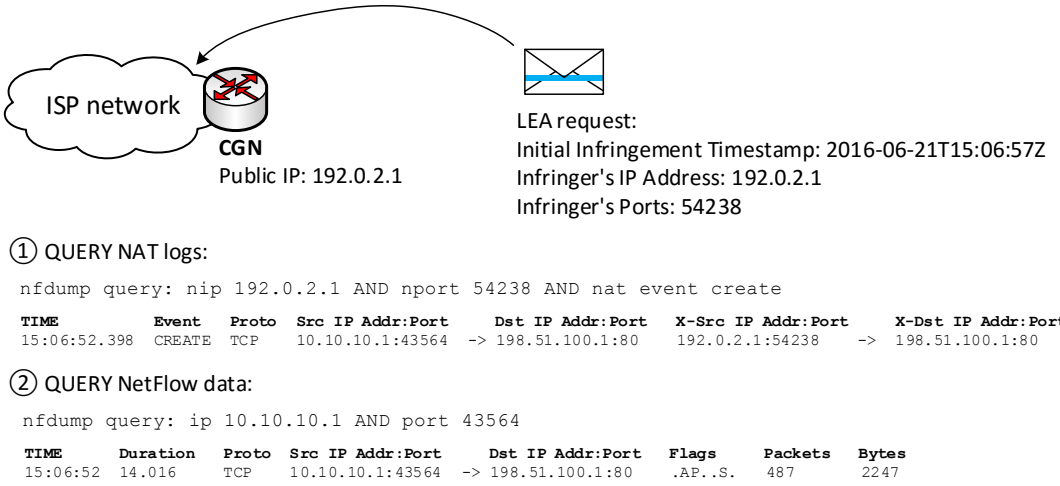


Figure 4.8: Different queries necessary to obtain all the information to fulfill the data retention request

This process is time consuming in practice and not easy at all.

We tried to solve these issues and this section presents a new approach for NAT accounting using standalone NetFlow probes. There is no need to change anything in network topology and there is no need for a NAT device to support logging as the logging is done outside of the address translation process. The whole idea is based on correlation of the traffic before and after the address translation. A NetFlow probe can be inserted in the network topology to monitor the inner and outer traffic as depicted in Figure 4.9.

Let us briefly describe example in the figure. A client with IP address `10.0.0.2` in the inner network tries to connect to a server with IP address `198.51.100.1`. The public address of the NAT device is `192.0.2.1`. Using a probe that exports NetFlow records before and after the translation, we obtain four flows on the collector. Two flows for outgoing direction – flow `10.0.0.2 -> 198.51.100.1` exported before the translation and flow `192.0.2.1 -> 198.51.100.1` exported after the translation. The same is with the returning (incoming) traffic. There will be two flows `198.51.100.1 -> 192.0.2.1` and `198.51.100.1 -> 10.0.0.2`.

Is it possible to create a monitoring system that is able to combine and correlate the flows? Firstly, let us create some properties that the system should have:

- Inner and outer flows must be correlated to obtain only two pieces of information – one flow for the traffic going from the inner network to the outer network and one flow for the returning traffic.
- Correlated flow information should contain all the necessary information – IP addresses and ports before and after translation, number of packets in the flow and number of transferred bytes.
- There cannot be any heuristic or only 90% of probability that the correlation is correct as the data could not be used as an evidence during the data retention process in that

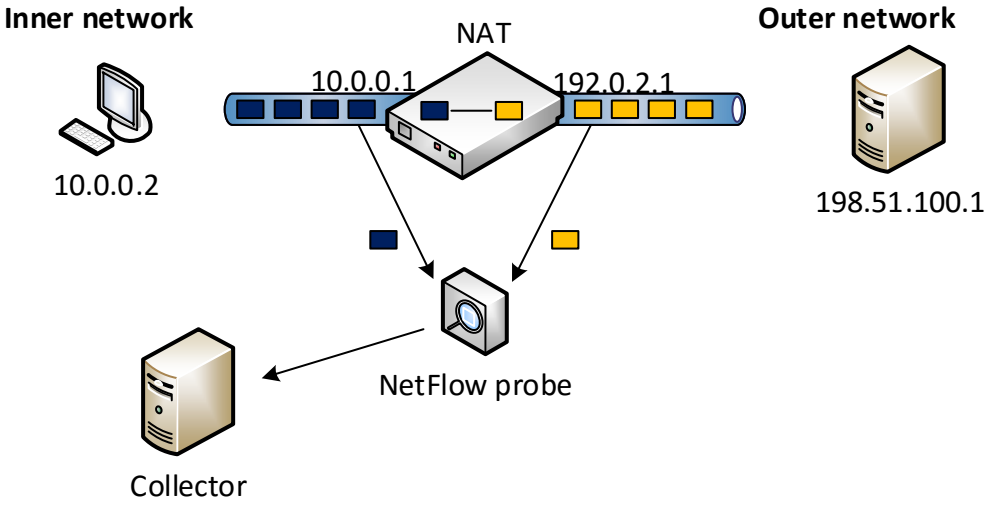


Figure 4.9: NetFlow probe can be inserted in a topology to monitor inner and outer traffic.

case. The correlation must be correct – no ambiguity is allowed. It is better to not create the correlation than create it wrongly.

- The NetFlow probe should be able to handle traffic at reasonable speeds. The solution should work for 10 Gbps speed on a commodity hardware.
- TCP, UDP and ICMP must be supported. Other protocols should be supported as well.
- The network topology should not be limited only for a one probe monitoring both the inner and outer links, but the solution should be flexible and allows more probes for better scaling.

4.3.1 Flows correlation

Before we start to discuss how we can correlate flows before and after translation, we have to take into consideration behavior of NetFlow probes. Firstly, the NetFlow probes are stateless – every packet is analyzed separately. The probe computes a hash from key fields[4] and stores the information in the flow cache. Another packet with the same hash updates the counters (number of packets/bytes) in the flow cache. Secondly, the flow creation process allocates and fills all necessary data structures in the flow cache. All subsequent packets simply increase the appropriate counters in the flow cache. Thus, if we want to add additional information to a flow, we should do that during the creation of the flow.

How is it possible to correlate the inner and outer flows? The first approach we tried was based on an assumption that it is possible to use two FIFO queues for flows correlation as depicted in Figure 4.10.

[4]In case of a standard flow definition, key fields are source and destination IP addresses, source and destination ports, interface, protocol and Type of Service

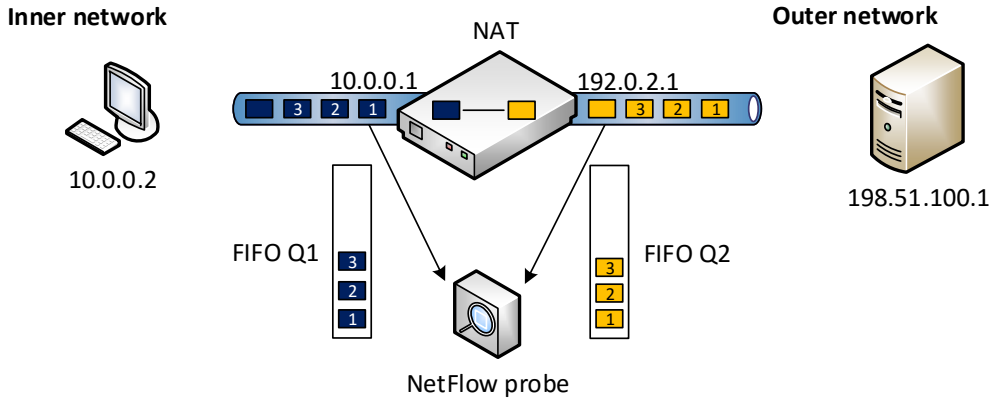


Figure 4.10: NetFlow probe with two FIFO queues for flows correlation.

We though that the first packet captured on the inner interface (packet 1) could be inserted into FIFO queue Q1 and matched with the first packet captured on the outer interface in FIFO queue Q2. Unfortunately, this approach can be used only if there is a small amount of traffic or if we have a strict control over Network Interface Card (NIC). The problem is that at higher speeds, the NIC does not use an interrupt to signal the kernel that there is a packet, but pushes several packets to the kernel in one batch. This lowers the amount of interrupts and increases the networking performance, but it also leads to a situation where the first packet on the inner interface does not have to correspond with the first packet captured on the outer interface, e.g., FIFO Q1 contains packets in order `(1,3,2)` and FIFO Q2 contains packets in order `(3,1,2)`.

Furthermore, there is problem with the implementation of such an approach. In practice, the NetFlow exporter is implemented as a standard Linux process that listens traffic on a NIC. The NetFlow probe must have two interfaces, hence, there must be two different processes. One process that monitors the inner NIC, one process for the outer NIC. The problem is: How to access FIFO Q1 from the outer process and vice versa? There must be some form of inter-process communication, either via shared memory or named sockets or pipes. However, such an approach limits the monitoring solution to only one probe.

Thus, another approach should be used. We could extract all the information that remains the same before and after the translation and use it to compute an unique identifier. Let us consider the same topology and traffic as in Figure 4.9. The probe can compute a unique value for the flow captured on probe's inner interface (`10.0.0.2 -> 198.51.100.1`) and repeats the same process for the flow captured on probe's outer interface `192.0.2.1 -> 198.51.100.1`.

As the computation can use fields that do not change, we should be able to compute the same unique identifier before and after the translation. Does the approach work in practice? Let us consider TCP protocol first. Figures 4.11 and 4.12 describe IP and TCP headers before and after the translation.

Considering the IP header in Figure 4.12, destination IP address together with the protocol field remain the same, thus these fields can be used to compute the unique identifier. Other fields in the IP header either change during the translation or they are not stable enough to be used, e.g., TTL field is decremented, and Identification remains the same, but

```
Internet Protocol Version 4, Src: 192.168.1.4, Dst: 147.229.9.14
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x8153 (33107)
    Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x5ac9 [validation disabled]
    Source: 192.168.1.4
    Destination: 147.229.9.14
Transmission Control Protocol
    Source Port: 37663
    Destination Port: 3128
    Sequence number: 1378765392    (relative sequence number)
    Acknowledgment number: 0
    Header Length: 40 bytes
    Flags: 0x002 (SYN)
    Window size value: 5840
    Checksum: 0xcb9f [validation disabled]
    Urgent pointer: 0
    Options: (20 bytes), Maximum segment size, SACK permitted,
                         Timestamps, No-Operation (NOP), Window scale
```

Figure 4.11: IP and TCP headers before NAT translation

```
Internet Protocol Version 4, Src: 10.10.10.220, Dst: 147.229.9.14
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x8153 (33107)
    Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 63
    Protocol: TCP (6)
    Header checksum: 0x0890 [validation disabled]
    Source: 10.10.10.220
    Destination: 147.229.9.14
Transmission Control Protocol
    Source Port: 37663
    Destination Port: 3128
    Sequence number: 1378765392    (relative sequence number)
    Acknowledgment number: 0
    Header Length: 40 bytes
    Flags: 0x002 (SYN)
    Window size value: 5840
    Checksum: 0x7866 [validation disabled]
    Urgent pointer: 0
    Options: (20 bytes), Maximum segment size, SACK permitted,
                         Timestamps, No-Operation (NOP), Window scale
```

Figure 4.12: IP and TCP headers after NAT translation

not every operating systems fill the value.

Considering the TCP header in the the same figure, only the checksum field is changed. However, NAT can rewrite the source port so we have to exclude this field as well. As the first packet initiating connection is a SYN packet, we cannot used any application payload as SYN packets does not have any payload[5].

Initially, we used only TCP sequence number as unique identifier as the TCP sequence number field is same before and after translation, it is randomly generated and has a reasonable size (32 bits). Using the TCP sequence number has also an advantage that the unique value is not tight with IP addresses, thus the probe does not have to distinguish inner and outer direction. We, however, encountered a problem if only the TCP sequence number field is used – collision. Although the sequence number is randomly generated on most operating systems, there could be a situation that two hosts generate the same value. More traffic means higher probability of collision. How much higher? It can be computed in the same way as Birthday problem [129]. The probabilities of collisions for 1000 and 10 000 concurrent TCP session, thus, are calculated using the following equations.

$$p(1000) = 1 - \binom{2^{32}}{1000} \times \frac{1000!}{(2^{32})^{1000}} \tag{4.1}$$

$$p(1000) = 0.0001162921... \tag{4.2}$$

$$p(10000) = 1 - \binom{2^{32}}{10000} \times \frac{10000!}{(2^{32})^{10000}} \tag{4.3}$$

$$p(10000) = 0.0115728899... \tag{4.4}$$

For 1000 simultaneous session, the probability is approximately 0.01 %. For 10 000 connection, the probability of collision increases to 1 %. Large networks can easily reach more than 10 000 flows/s, thus the probability of collision is quite high. As we want to be sure that there is no ambiguity for flows correlation, we extended the number of fields to compute the unique value. Currently, we use combination of destination IP together with TCP sequence number to compute the unique value. Using this approach, there could be 10 000 simultaneous connection to one server. If it is still not enough and there is more

[5] Although there is an attempt from Google and Apple to change this behavior and use payload even with the first SYN packet.

than 10 000 simultaneous connection to the same server, another options can be used – IP Identification (if set), or TCP options.

UDP and ICMP traffic is rather easier as these protocols carry application payload. As CGN does not change the payload, we can hash the payload to obtain a unique value that is the same before and after the translation. Similarly to TCP, another fields can be used as well – IP Identification (if set) and destination port number in case of the UDP protocol.

4.3.2 Implementation

We implemented the extension for the NetFlow probe and tested it in a production environment in BUT network. The architecture of the system is as follows. We used Flowmon probe for exporting NetFlow records extended with ID value. The Flowmon probe allows to use an API for a development of an own plugin. Several functions (hooks) are provided to alter parsing and processing of packets. We used an approach where all the necessary information is computed when a flow record is created. It means that computation of the unique ID value is done only once per flow. This leads to the decreasing of the CPU load on the probe and increasing of performance.

It is, however, not sufficient to compute the unique value before and after the translation and export the NetFlow record to a collector. Why? The collector typically stores incoming NetFlow records to a temporary file and rotates the file every five minutes. Flows with the same ID can be merged after the file rotation. This greatly increase the probability of ID collision as there are many more flows in the five minute file. For example, there was 5 million flows within a five-minute period on 12.6.2016 in the BUT NetFlow dataset. According to Equations 4.4, the probability of a collision in this case is 100 % if we use only the TCP sequence number field. If we combine the TCP sequence number with additional fields (IP address, TCP options), the probability of collision is reduced, but still significant[6].

Furthermore, if we compute the unique ID, add the value of the unique ID to the flow and export the flow extended with ID value to the collector, it means that the unique ID is computed for every incoming and outgoing packet. The consequence is that the probe computes ID value even for packets that do not go through the CGN/NAT translation process. This behavior is not desired as it increase the CPU load of the probe and the probability of the collision ID. Moreover, it is more complicated to merge flows on the collector using the ID value as there are inner flows with ID value set without the corresponding outer flows and vice versa. This situation happens when the inner/outer interface receives a packet, but the packet is dropped or not translated.

How to solve these limitations? Firstly, the ID value should be set only for packets that are translated by the NAT device. Secondly, the probability of collision must be decreased to a reasonable small value. Thirdly, the system should support different probes on inner and outer links.

It is possible to solve these issues by creating some kind of stateful information in the stateless monitoring process, but how? There must be an inter-process communication between devices, but without the drawbacks described in the example with FIFO queues. We tested several approaches – shared memory, communication with sockets, etc. In the end, we decided to use an external in-memory key-value cache on Flowmon probe for inter-process communication. The cache is implemented using Redis [130]. Redis is an open source, in-memory data structure store, that can be used as database, cache or message broker. It runs as a standalone process on the NetFlow probe and every NetFlow exporter

[6]The probability depends on the traffic distribution

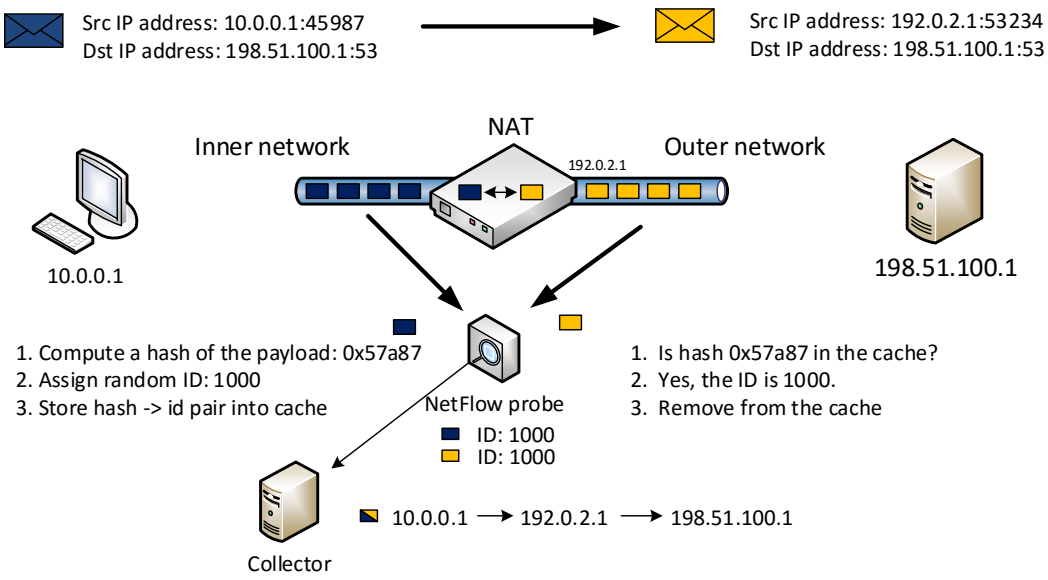


Figure 4.13: NetFlow probe extended with the in-memory cache. The probe computes unique ID value only for packets that are translated by the NAT box.

process connects to the Redis database to store or retrieve necessary information. The whole monitoring process works as follows.

Each time the incoming packet is seen on probe's inner interface, the NetFlow exporter computes the unique value from the fields described previously and uses the value as a key for the Redis cache. The value for the key is the unique ID field as discussed above. In practice, it is a 64, 128 or 256-bit random number (the length depends on the network size and it is configurable). If the packet should be translated by the NAT box, the probe intercepts the packet on the inner interface as depicted in Figure 4.13. The probe computes the key (step 1), assigns the unique ID (step 2) and stores the key-value pair in the cache (step 3).

The translated packet is intercepted by the probe on the outer link. The probe computes the same unique key as for the inner flow and search the in-memory cache. If the key is found, the NetFlow exporter running on probe's outer interface obtains the corresponding value from the cache, stores the value to flow's internal data structure and invalidates the record in the in-memory cache. This leads to a situation that NetFlow probe exports two flows – before and after the translation and both flows contain the same ID value. The 64-bit random value provides enough entropy that collision is very unlikely[7] even in a large network. An advantage of this approach is also the fact that only traffic that is translated by CGN can be tight together on a collector. There is an expiration timeout set to the key in the cache that invalidates the key automatically after a certain period of time – we use a second, but the timeout is configurable. The timeout ensures that the cache does not growth endlessly. As we use in-memory database that provides access via UNIX sockets, we are not limited to one probe only, but it is possible to use standalone probes for inner and outer link monitoring. Without any optimization, the in-memory database is capable

[7]Probability of collision is $6,7 \times 10^{-7}$ if there are 5 million flows

to store 100 000 key-value pairs per second. The size of the database does not have any performance impact as it is assured that the time complexity for retrieving and storing a key-value pair is always $\mathcal{O}(1)$. As the cache is accessed only in the case of flow creation, it means that without any optimization, the NetFlow probe is able to process 100 000 flows/s. It was reported, that a cascade of in-memory caches can handle billions of requests per second [131].

Collector processes NetFlow data and extracts unique IDs. These numbers are compared with each other and if there is a match, flows are merged together. One of the biggest advantages of this approach is that there is only one record that describes the whole translation process and also includes statistics such as number of packets and bytes. Hence, all the necessary information can be obtained only by one query. Figure 4.14 shows an example of NetFlow data before and after the merge on a collector.

Date first seen	Src IP Addr:Port		Dst IP Addr:Port	Bytes	Pkts
2016-06-04 12:25:13.840	8.8.8.8:53	->	192.168.1.4:36220	192	2
2016-06-04 12:25:13.844	10.10.10.220:37663	->	147.229.9.14:3128	487	8
2016-06-04 12:25:13.869	192.168.1.4:37664	->	147.229.9.14:3128	2247	41
2016-06-04 12:25:13.844	147.229.9.14:3128	->	192.168.1.4:37663	781	5
2016-06-04 12:25:13.870	147.229.9.14:3128	->	192.168.1.4:37664	57123	49
2016-06-04 12:25:13.870	10.10.10.220:37664	->	147.229.9.14:3128	2247	41
2016-06-04 12:25:13.870	147.229.9.14:3128	->	10.10.10.220:37664	57123	49
2016-06-04 12:25:13.844	192.168.1.4:37663	->	147.229.9.14:3128	487	8
2016-06-04 12:25:13.844	147.229.9.14:3128	->	10.10.10.220:37663	781	5
2016-06-04 12:25:13.826	10.10.10.220:36220	->	8.8.8.8:53	128	2
2016-06-04 12:25:13.825	192.168.1.4:36220	->	8.8.8.8:53	128	2
2016-06-04 12:25:13.839	8.8.8.8:53	->	10.10.10.220:36220	192	2

Src IP Addr:Port		Dst IP Addr:Port	X-late Src IP:Port		X-lateDst IP:Port	Bytes	Pkts
192.168.1.4:37664	->	147.229.9.14:3128	10.10.10.220:37644	->	147.229.9.14:3128	2247	41
192.168.1.4:37663	->	147.229.9.14:3128	10.10.10.220:37663	->	147.229.9.14:3128	487	8
147.229.9.14:3128	->	10.10.10.220:37663	147.229.9.14:3128	->	192.168.1.4:37663	781	5
147.229.9.14:3128	->	10.10.10.220:37664	147.229.9.14:3128	->	192.168.1.4:37664	57123	49
192.168.1.4:36220	->	8.8.8.8:53	10.10.10.220:36220	->	8.8.8.8:53	128	2
8.8.8.8:53	->	10.10.10.220:36220	8.8.8.8:53	->	192.168.1.4:36220	192	2

Figure 4.14: Example of merging inner and outer flows to one flow that contains all the necessary information

4.4 Summary

This chapter describes Network address translation process and issues that the mechanism presents for user accounting. Several possibilities for NAT logging were discussed – Syslog, NEL, NSEL and bulk port allocation and bulk logging. These methods can be used, but have some disadvantages. The NAT device must support the logging mechanism and CPU processor on the NAT device must be powerful enough to handle the logging process even if there are a lot of sessions. Furthermore, the logging methods must be combined with traditional NetFlow data to obtain the whole picture about user's activities.

We introduced a novel approach for NAT logging that eliminates these issues. Logging is performed from an external, standalone NetFlow probe that intercepts traffic before and after the translation. The probe computes the same unique ID for packets before and after translation and extends NetFlow records with the unique identifier before exporting them to the collector. The unique ID is used on the collector for flows correlation. The output of the flow correlation is one flow with all the necessary information. There are several benefits of this approach. Firstly, the NAT box does not have to support logging. It saves NAT's CPU and it is possible to account users even in networks where it was impossible. Secondly, it is much easier to query and obtain data retention logs as all the information is in one record. Thirdly, a large amount of disk space is saved as duplicated information is eliminated.

The approach for NAT logging were discussed in our paper – [10]. Implementation of the proposed solution was supported by CESNET grant 546R1/2014. Feasibility of the approach was discussed in GÉANT Best Practice workgroup. Currently, the implementation runs in production at BUT dormitory campus network.

5

Conclusion

The thesis deals with the issues of user monitoring and accounting especially in next generation networks. We focused primarily on IPv6 protocol as other proposals of new networking protocols or architectures are still in a development stage and not widely deployed. The thesis is divided into three main chapters. Chapter 2 presents analysis and statistics about the transition from IPv4 to IPv6 and provides the necessary knowledge about the IPv6 transition progress for the rest of the thesis. Chapter 3 describes challenges for user accounting presented by dual stack networks and networks where some kind of transition mechanism between IPv4 and IPv6 is deployed. The chapter introduces a central system that is able to account user even in these networks. Chapter 4 extends the accounting system with a support for network address translation mechanism. All these chapters together solve issues with user accounting created by the transition between two incompatible networking protocols. The following parts of this chapter summarize the main observations and contributions presented by the thesis.

Analysis of the IPv4-IPv6 transition

The thesis closely describes the transition to IPv6 in Chapter 2. The chapter is divided to three sections where each of the sections analyses the transition process from a different angle.

The first section of the chapter analyses routing infrastructure. The global BGP table is examined for current trends in IPv6 adoption. Several interesting facts emerged. The support for IPv6 is slightly lower among new companies (32-bit ASNs). The growth of transit only ASes supporting IPv6 drops significantly in 2014 – 2015 period. Slower growth of IPv6 support in the routing core means smaller path diversity and robustness. We did a correlation of BGP analysis with NetFlow data from BUT and CESNET networks. It is a novel approach, as BGP analysis is usually presented without any relationship with the real network traffic. We found out that the the ratio between the number of unique `/64` prefixes in one `/48` prefix is around 1.5 – 2. The ratio should be much higher in developed IPv6 networks – we see the ratio around 10 for developed networks (Facebook, Google, O2 Czech, etc.).

The second section of the chapter focuses on content availability and quality analysis. The section presents several figures showing the IPv6 support among the web, mail and DNS services since 2012. We compared our results with other projects measuring the IPv6 adoption and found out, that these projects overestimate the IPv6 content penetration. The main reason is that other projects use much smaller dataset compare to ours. The

section also discuss a quality of IPv6 connection to dual stack websites. The conclusion is that IPv4 and IPv6 perform similarly in most cases. We also found out that occurrences where one protocol performs better than other are mainly caused by differences in routing paths. Our measurements show that there is a substantial number of sites (around 5 %) we are not able to connect to.

The third section describes support for IPv6 protocol among users' devices and IPv6 traffic volume. The section presents long-term statistics of user IPv6 support in BUT campus network and finds out that IPv6 support is very high – around 80 % of devices (laptops, PCs and mobiles) actively use IPv6. Furthermore, the section discusses IPv6 traffic volume and flow ratio. The traffic volume oscillates around 20 % and flow ratio around 10 %. There is no big difference between traffic volume in 2016 and three years ago. This is caused by the fact, that main content providers enabled IPv6 in 2012. There was not any bigger movement in IPv6 support for small websites that comprises the rest of the network traffic in recent years.

The main contributions of Chapter 2 are the presented observations and statistics about the IPv4-IPv6 transition. All these analysis and statistics use large and unique datasets. Content analysis uses our own dataset much larger then others. All the statistics are publicly available online and are regularly updated.

User accounting and transition technologies

Chapter 3 describes the process of address assignment and user accounting in today's IPv4 networks. Address assignment techniques for the IPv6 protocol are described as well, together with transition techniques that are used in today's networks to overcome incompatibility between IPv4 and IPv6 protocols. The chapter discusses issues and challenges that must be addressed if we want to have a robust accounting system both for IPv4 and IPv6. It introduces a central accounting system that is able to account users even in dual stacked networks or network with a transition technology deployed. The system uses hardware or software probes that are able to detect IPv6 transition techniques and account the traffic inside the tunnel. Data from these probes are collected on a central data store and extended with additional information. This information is gathered from various sources, e.g., L3 gateways, server logs or by passive monitoring techniques.

The main contributions of Chapter 3 are the following. *(i)* Detailed description of challenges for users accounting process. *(ii)* Observations and practical experience with behavior of operating systems in IPv6 networks, how many IPv6 addresses can the network administrator expect, etc. *(iii)* Specialized probes that are able to cope with different transition techniques. The standard NetFlow probes or routers do not provide such functionality. *(iv)* An accounting system that can process and store all the necessary information. The system is application-aware, meaning there is an API that can be used for future applications. The system is built using open source technologies and it is freely available to everyone. Furthermore, it is proven that the system can run in rather large and complex network such as BUT campus. The system was used several times in practice to track IPv6 security incidents and malevolent users.

User accounting and address translation technologies

Chapter 4 describes the network address translation process and issues that the mechanism introduces for user accounting. The chapter focuses on different types of NAT logging that could be used for extending the central accounting system presented in Chapter 3, e.g.,

Syslog, NEL, NSEL, bulk port allocation and bulk logging. A novel approach for NAT logging that eliminates issues presented by address translation is introduced. The NAT logging process is performed from an external, standalone NetFlow probe that intercepts traffic before and after the translation. The benefit of this approach is that NAT does not have to support logging which save NAT's CPU. Furthermore, NetFlow probe extends NetFlow records with an unique identifier that is used on a collector for correlation and merging flows.

The main contributions of chapter 4 are the following. *(i)* In-depth description of different address translation approaches. *(ii)* A novel approach for NAT logging. The approach saves NAT's CPU and allows NAT accounting even if the NAT box does not have support for it. Furthermore, it is possible to obtain a complete view on the translation process in one flow. Hence, it is possible to easily trace back security incidents and create statistics for user accounting.

5.1 Future work

The thesis solves several problems of user accounting in IPv6 networks and networks with address translation. Furthermore, it presented several statistics and different views on the IPv6 transition progress. However, the job is not yet done. In the nearest future, we would like to solve a few remaining issues with the measurement of IPv6 transition. Mainly, NetFlow data correlation with BGP in cases where an ISP uses `/48` prefix per customer. We plan to run more tests for NAT logging approach introduced in Chapter 4. Even though the approach works, we would like to test it with more users and with different NetFlow probes.

What about other networking architectures? Recursive InterNetwork Architecture, Content Centric Networking or Named Data Networking are other examples of new networking architectures that are currently proposed. It is possible to predict the changes in the user accounting process for a future networking architecture? Unfortunately, I do not think so. Let us take IPv4 – IPv6 transition as an example. At the beginning of IPv6 deployment a lot of network administrators thought that the user accounting for IPv6 will be the same or easier. As a matter of fact, one of the goals of IPv6 was to simplify the network management. That was the theory. The practice is, however, very different. Considering the fact that every networking architecture that has been proposed so far is a clean slate design, I believe we will face similar issues and challenges in future. The thesis covered a lot of different methods for user accounting. Hopefully these methods and different approaches presented in this thesis will help with the future transition.

Bibliography

[1] Cisco Systems, Inc., "Introducing to Cisco IOS NetFlow," [online], Cisco, Tech. Rep., May 2012, [cited 08.8.2013].

[2] M. Kosters and G. Huston, "CGNs in IP. What are you going to do about it?" January 2013, [cited 11.11.2013]. [Online]. Available: http://www.potaroo.net/presentations/2013-01-16-cgn-kosters-joint-techs.pdf

[3] D. C. Mowery and T. Simcoe, "Is the Internet a US invention? — an economic and technological history of computer networking," Research Policy, vol. 31, no. 8–9, 2002, pp. 1369 – 1387. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0048733302000690

[4] J. Postel, "NCP/TCP transition plan," RFC 801, Internet Engineering Task Force, Nov. 1981. [Online]. Available: http://www.ietf.org/rfc/rfc801.txt

[5] A. Malis, "ARPANET 1822L Host Access Protocol," RFC 851, Internet Engineering Task Force, Apr. 1983, obsoleted by RFC 878. [Online]. Available: http://www.ietf.org/rfc/rfc851.txt

[6] G. Huston, "Is the Transition to IPv6 a „Market Failure?"," September 2009, [cited 19.4.2013]. [Online]. Available: http://www.potaroo.net/ispcol/2009-09/v6trans.html

[7] I. Livadariu, A. Elmokashfi, A. Dhamdhere, and K. Claffy, "A first look at IPv4 transfer markets," in Proceedings of the ninth ACM conference on Emerging networking experiments and technologies. ACM, 2013, pp. 7–12.

[8] J. Curran, "An Internet Transition Plan," RFC 5211 (Informational), Internet Engineering Task Force, Jul. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5211.txt

[9] M. Grégr, T. Podermański, and M. Švéda, "Measuring Quality and Penetration of IPv6 Services," ICNS 2014, 2014.

[10] M. Grégr and M. Švéda, "Challenges with Transition and User Accounting in Next Generation Networks," in ICNP 2014. Raleigh, NC, USA, US: Institute of Electrical and Electronics Engineers, 2014.

[11] G. Huston, "Testing IPv6 for World IPv6 Day," May 2011, [cited 19.4.2013]. [Online]. Available: http://www.potaroo.net/ispcol/2011-05/ip6test.html

[12] L. Colitti, S. Gunderson, E. Kline, and T. Refice, "Evaluating IPv6 Adoption in the Internet," in Passive and Active Measurement. Springer Berlin / Heidelberg, 2010.

[13] G. Huston, "Measuring IPv6 - Country by Country," July 2012, [cited 19.4.2013]. [Online]. Available: http://www.potaroo.net/ispcol/2012-07/v6report.html

[14] M. Karir, G. Huston, G. Michaelson, and M. Bailey, "Understanding ipv6 populations in the wild," in Passive and Active Measurement, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7799, pp. 256–259. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36516-4˙27

[15] E. Karpilovski, A. Gerber, D. Pei, J. Rexford, and A. Shaikh, "Quantifying the Extent of IPv6 Deployment," in Network Measurement. Springer Berlin, 2009.

[16] G. Huston, "BGP Reports," [cited 12.11.2014]. [Online]. Available: http://bgp.potaroo.net

[17] X. Zhou and P. Van Mieghem, "Hopcount and E2E Delay: IPv6 Versus IPv4," in Passive and Active Measurement. Springer Berlin / Heidelberg, 2005.

[18] X. Zhou, M. Jacobsson, H. Uijterwaal, and P. Van Mieghem, "IPv6 delay and loss performance evolution," International Journal of Communication Systems, vol. 21, no. 6, 2008, pp. 643–663.

[19] A. Berger, "Comparison of Performance over IPv6 versus IPv4," Akamai Technologies, 2011.

[20] A. Dhamdhere, M. Luckie, B. Huffaker, A. Elmokashfi, E. Aben et al., "Measuring the deployment of ipv6: topology, routing and performance," in Proceedings of the 2012 ACM conference on Internet measurement conference. ACM, 2012, pp. 537–550.

[21] M. Nikkhah, R. Guérin, Y. Lee, and R. Woundy, "Assessing IPv6 through web access a measurement study and its findings," in Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies. ACM, 2011, p. 26.

[22] J. Czyz, M. Allman, J. Zhang, S. Iekel-johnson, E. Osterweil, and M. Bailey, "Measuring IPv6 Adoption," in SIGCOMM 2014, Chicago, Illinois, USA, 2014, pp. 87–98.

[23] O. M. Crépin-Leblond, "IPv6 Matrix Project," [cited 22.12.2013]. [Online]. Available: http://www.ipv6matrix.org

[24] IPv6 Observatory, "Top-500 websites with AAAA records," [cited 22.12.2013]. [Online]. Available: http://www.ipv6observatory.eu/indicator/

[25] E. Vyncke, "IPv6 Deployment Aggregated Status," [cited 22.12.2013]. [Online]. Available: http://www.vyncke.org/ipv6status

[26] M. Leber, "Global IPv6 Deployment Progress Report," [cited 22.12.2013]. [Online]. Available: http://bgp.he.net/ipv6-progress-report.cgi

[27] A. Keranen and J. Arkko, "Some Measurements on World IPv6 Day from an End-User Perspective," RFC 6948 (Informational), Internet Engineering Task Force, Jul. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6948.txt

[28] H. Kaczmarek, "Internet IPv6 Adoption: Methodology, Measurement and Tools," July 2012, [cited 19.4.2013]. [Online]. Available: http://6lab.cisco.com/stats/information

[29] University of Oregon, "Route Views Project," [cited 21.1.2015]. [Online]. Available: http://www.routeviews.org

[30] University of Oregon, "The State of the Internet," [cited 21.1.2015].

[31] Q. Vohra and E. Chen, "BGP Support for Four-octet AS Number Space," RFC 4893 (Proposed Standard), Internet Engineering Task Force, May 2007, obsoleted by RFC 6793. [Online]. Available: http://www.ietf.org/rfc/rfc4893.txt

[32] G. Huston, "BGP in 2014," January 2015, [cited 23.2.2015]. [Online]. Available: http://www.potaroo.net/ispcol/2015-01/bgp2014.html

[33] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 4291 (Draft Standard), Internet Engineering Task Force, Feb. 2006, updated by RFCs 5952, 6052, 7136, 7346, 7371. [Online]. Available: http://www.ietf.org/rfc/rfc4291.txt

[34] G. Huston, "Flailing IPv6," December 2010, [cited 23.2.2015]. [Online]. Available: http://www.potaroo.net/ispcol/2010-12/6to4fail.html

[35] G. Huston, "Testing Teredo," April 2011, [cited 23.2.2015]. [Online]. Available: http://www.potaroo.net/ispcol/2011-04/teredo.html

[36] B. Carpenter, "IPng White Paper on Transition and Other Considerations," RFC 1671 (Informational), Internet Engineering Task Force, Aug. 1994. [Online]. Available: http://www.ietf.org/rfc/rfc1671.txt

[37] J. Livingood, "Considerations for Transitioning Content to IPv6," RFC 6589 (Informational), Internet Engineering Task Force, Apr. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6589.txt

[38] D. Wing and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts," RFC 6555 (Proposed Standard), Internet Engineering Task Force, Apr. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6555.txt

[39] O. Bonaventure, "Happy eyeballs makes me unhappy..." December 2013, [cit. 27.2.2014]. [Online]. Available: http://perso.uclouvain.be/olivier.bonaventure/blog/html/2013/12/03/happy.html

[40] N. Sarrar, G. Maier, B. Ager, R. Sommer, and S. Uhlig, "Investigating IPv6 Traffic," in Passive and Active Measurement. Springer Berlin / Heidelberg, 2012.

[41] J. Bitte, "HTTP logging and information retrieval tool," [cit. 27.2.2014]. [Online]. Available: http://dumpsterventures.com/jason/httpry/

[42] T. Podermański and M. Grégr, "Worldwide online IPv6 penetration," August 2013, [cit. 27.2.2014]. [Online]. Available: http://6lab.cz/live-statistics/data-source

[43] CZ.NIC, ".cz statistics," [cited 22.12.2013]. [Online]. Available: http://stats.nic.cz

[44] D. Wing, "AAAA and IPv6 Connectivity statistics," July 2009, [cit. 27.5.2015]. [Online]. Available: http://perso.uclouvain.be/olivier.bonaventure/blog/html/2013/12/03/happy.html

[45] E. Vyncke, "IPv6 Deployment Monitoring: Internet metrics," July 2013, [cit. 22.7.2015]. [Online]. Available: http://www.ipv6hackers.org/meetings/ipv6-hackers-1

[46] R. Zajíc, "Jak obejít nefunkční IPv6 na justice.cz," [cited 22.12.2013]. [Online]. Available: http://zajic.v.pytli.cz/2012/06/10/jak-obejit-nefunkcni-ipv6-na-justice-cz/

[47] NANOG discussion, "IPv6 Cogent vs Hurricane Electric," [cited 22.3.2016]. [Online]. Available: http://mailman.nanog.org/pipermail/nanog/2015-December/082669.html

[48] P. Saab, "Facebook IPv6 Strategy," March 2015, [cit. 06.10.2015]. [Online]. Available: https://youtu.be/An7s25FSK0U

[49] S. Greenstein, "Framing empirical work on the evolving structure of commercial Internet markets," Understanding the Digital Economy (Cambridge, MA: MIT Press, 2000a), 2000.

[50] ETSI, "Handover interface for the request and delivery of retained data," 2010, [version 1.7.1].

[51] M. Patrick, "DHCP Relay Agent Information Option," RFC 3046 (Proposed Standard), Internet Engineering Task Force, Jan. 2001, updated by RFC 6607. [Online]. Available: http://www.ietf.org/rfc/rfc3046.txt

[52] INNET VŠB - Technical University of Ostrava, "New computer registration," 2013, [cited 07.8.2013]. [Online]. Available: http://idoc.vsb.cz/en/okruhy/cit/tuonet/pripojeni/studenti/koleje/registrace/

[53] B. Robenek, "Informační systém klubu Silicon Hill - správa sítě," 2013. [Online]. Available: http://installfest.cz/if13/files/slidy/robenek-is-sh.pdf

[54] A. Houdek, "Pravidla používání internetu na kolejích KaM (mimo kolej 17.listopadu)," 2013, czech, [cited 07.8.2013]. [Online]. Available: http://www.kam.cuni.cz/KAM-33.html

[55] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC 7011 (INTERNET STANDARD), Internet Engineering Task Force, Sep. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7011.txt

[56] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), Internet Engineering Task Force, Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3954.txt

[57] S. Kawamura and M. Kawashima, "A Recommendation for IPv6 Address Text Representation," RFC 5952 (Proposed Standard), Internet Engineering Task Force, Aug. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5952.txt

[58] Internet Assigned Numbers Authority, "Internet Protocol Version 6 Address Space," 2013, [cited 09.8.2013]. [Online]. Available: http://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml

[59] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," RFC 4862 (Draft Standard), Internet Engineering Task Force, Sep. 2007, updated by RFC 7527. [Online]. Available: http://www.ietf.org/rfc/rfc4862.txt

[60] T. Narten, R. Draves, and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," RFC 4941 (Draft Standard), Internet Engineering Task Force, Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4941.txt

[61] T. Aura, "Cryptographically Generated Addresses (CGA)," RFC 3972 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFCs 4581, 4982. [Online]. Available: http://www.ietf.org/rfc/rfc3972.txt

[62] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," RFC 4861 (Draft Standard), Internet Engineering Task Force, Sep. 2007, updated by RFCs 5942, 6980, 7048, 7527, 7559. [Online]. Available: http://www.ietf.org/rfc/rfc4861.txt

[63] B. Haberman and R. Hinden, "IPv6 Router Advertisement Flags Option," RFC 5175 (Proposed Standard), Internet Engineering Task Force, Mar. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5175.txt

[64] C. Perkins, D. Johnson, and J. Arkko, "Mobility Support in IPv6," RFC 6275 (Proposed Standard), Internet Engineering Task Force, Jul. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6275.txt

[65] J. Jeong, S. Park, L. Beloeil, and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration," RFC 6106 (Proposed Standard), Internet Engineering Task Force, Nov. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc6106.txt

[66] IANA: Thomas Huth, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," October 2013, [cited 24.10.2013]. [Online]. Available: http://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.txt

[67] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," RFC 3315 (Proposed Standard), Internet Engineering Task Force, Jul. 2003, updated by RFCs 4361, 5494, 6221, 6422, 6644, 7083, 7227, 7283, 7550. [Online]. Available: http://www.ietf.org/rfc/rfc3315.txt

[68] T. Narten and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)," RFC 6355 (Proposed Standard), Internet Engineering Task Force, Aug. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6355.txt

[69] A. Conta and S. Deering, "Generic Packet Tunneling in IPv6 Specification," RFC 2473 (Proposed Standard), Internet Engineering Task Force, Dec. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2473.txt

[70] E. Nordmark and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers," RFC 4213 (Proposed Standard), Internet Engineering Task Force, Oct. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4213.txt

[71] A. Durand, R. Droms, J. Woodyatt, and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion," RFC 6333 (Proposed Standard), Internet Engineering Task Force, Aug. 2011, updated by RFC 7335. [Online]. Available: http://www.ietf.org/rfc/rfc6333.txt

[72] X. Li, C. Bao, W. Dec, O. Troan, S. Matsushima, and T. Murakami, "Mapping of Address and Port using Translation (MAP-T)," RFC 7599 (Proposed Standard), Internet Engineering Task Force, Jul. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7599.txt

[73] O. Troan, W. Dec, X. Li, C. Bao, S. Matsushima, T. Murakami, and T. Taylor, "Mapping of Address and Port with Encapsulation (MAP-E)," RFC 7597 (Proposed Standard), Internet Engineering Task Force, Jul. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7597.txt

[74] R. Bush, "The Address plus Port (A+P) Approach to the IPv4 Address Shortage," RFC 6346 (Experimental), Internet Engineering Task Force, Aug. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6346.txt

[75] R. Despres, S. Jiang, R. Penno, Y. Lee, G. Chen, and M. Chen, "IPv4 Residual Deployment via IPv6 - A Stateless Solution (4rd)," RFC 7600 (Experimental), Internet Engineering Task Force, Jul. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7600.txt

[76] B. Carpenter and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds," RFC 3056 (Proposed Standard), Internet Engineering Task Force, Feb. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3056.txt

[77] C. Huitema, "An Anycast Prefix for 6to4 Relay Routers," RFC 3068 (Historic), Internet Engineering Task Force, Jun. 2001, obsoleted by RFC 7526. [Online]. Available: http://www.ietf.org/rfc/rfc3068.txt

[78] G. Huston, "Flailing IPv6," December 2010, [cited 17.12.2015]. [Online]. Available: http://www.potaroo.net/ispcol/2010-12/6to4fail.html

[79] T. Anderson, "IPv6 dual-stack client loss in Norway," December 2010, [cited 17.12.2015]. [Online]. Available: https://fud.no/ipv6/

[80] E. Aben and T. Anderson, "6to4 - How Bad is it Really?" December 2010, [cited 17.12.2015]. [Online]. Available: https://labs.ripe.net/Members/emileaben/6to4-how-bad-is-it-really

[81] B. Carpenter, "Advisory Guidelines for 6to4 Deployment," RFC 6343 (Informational), Internet Engineering Task Force, Aug. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6343.txt

[82] O. Troan and B. Carpenter, "Deprecating the Anycast Prefix for 6to4 Relay Routers," RFC 7526 (Best Current Practice), Internet Engineering Task Force, May 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7526.txt

[83] R. Despres, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)," RFC 5569 (Informational), Internet Engineering Task Force, Jan. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5569.txt

[84] W. Townsley and O. Troan, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) – Protocol Specification," RFC 5969 (Proposed Standard), Internet Engineering Task Force, Aug. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5969.txt

[85] B. Carpenter and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," RFC 2529 (Proposed Standard), Internet Engineering Task Force, Mar. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2529.txt

[86] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," RFC 4380 (Proposed Standard), Internet Engineering Task Force, Feb. 2006, updated by RFCs 5991, 6081. [Online]. Available: http://www.ietf.org/rfc/rfc4380.txt

[87] J. Hoagland, "The Teredo Protocol: Tunneling Past Network Security and Other Security Implications," November 2006, [cit. 3.1.2016]. [Online]. Available: http://www.symantec.com/avcenter/reference/Teredo˙Security.pdf

[88] Microsoft, "Teredo Overview," January 2007, [cit. 5.1.2016]. [Online]. Available: https://technet.microsoft.com/en-us/library/bb457011.aspx

[89] D. Veit and C. Palmer, "Usage of Teredo and IPv6 for P2P on Windows 10 and Xbox One," May 2015, [cited 7.1.2016]. [Online]. Available: http://mailman.nanog.org/pipermail/nanog/2015-May/075244.html

[90] C. Palmer, "Usage of Teredo and IPv6 for P2P on Windows 10 and Xbox One," May 2015, [cited 7.1.2016]. [Online]. Available: http://mailman.nanog.org/pipermail/nanog/2015-May/075244.html

[91] A. Nakagawa, "Operational Experience of MAP-E," Working Draft, IETF Secretariat, Internet-Draft draft-akira-v6ops-mape-experience-00, July 2015. [Online]. Available: http://www.ietf.org/internet-drafts/draft-akira-v6ops-mape-experience-00.txt

[92] F. Gont, A. Cooper, D. Thaler, and W. Liu, "Recommendation on Stable IPv6 Interface Identifiers," October 2015, work-in-progress, [cit. 18.1.2016]. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6man-default-iids-08

[93] E. Jankiewicz, J. Loughney, and T. Narten, "IPv6 Node Requirements," RFC 6434 (Informational), Internet Engineering Task Force, Dec. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6434.txt

[94] T. Anderson, "Issue 32621: Support for DHCPv6 (RFC 3315)," June 2012, [cit. 21.1.2016]. [Online]. Available: https://code.google.com/p/android/issues/detail?id=32621

[95] W. Kumari, O. Gudmundsson, P. Ebersman, and S. Sheng, "Captive-Portal Identification Using DHCP or Router Advertisements (RAs)," RFC 7710 (Proposed Standard), Internet Engineering Task Force, Dec. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7710.txt

[96] G. Halwasia, S. Bhandari, and W. Dec, "Client Link-Layer Address Option in DHCPv6," RFC 6939 (Proposed Standard), Internet Engineering Task Force, May 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6939.txt

[97] M. Elich, M. Grégr, and P. Čeleda, "," in Traffic Monitoring and Analysis, ser. Lecture Notes in Computer Science 6613. Springer Verlag, 2011, pp. 64–71. [Online]. Available: http://www.fit.vutbr.cz/research/view˙pub.php?id=9604

[98] M. Elich, P. Velan, T. Jirsik, and P. Celeda, "An investigation into teredo and 6to4 transition mechanisms: Traffic analysis," in Local Computer Networks Workshops (LCN Workshops), 2013 IEEE 38th Conference on, Oct 2013, pp. 1018–1024.

[99] M. Grégr, "IPv6 transition techniques monitoring tool," October 2012. [Online]. Available: http://www.fit.vutbr.cz/~igregr/prods.php.en?id=268

[100] L. Polčák, M. Holkovič, and P. Matoušek, "A New Approach for Detection of Host Identity in IPv6 Networks," in Proceedings of the 4th International Conference on Data Communication Networking, 10th International Conference on e-Business and 4th International Conference on Optical Communication Systems. SciTePress - Science and Technology Publications, 2013, pp. 57–63. [Online]. Available: http://www.fit.vutbr.cz/research/view˙pub.php?id=10362

[101] N. Moore, "Optimistic Duplicate Address Detection (DAD) for IPv6," RFC 4429 (Proposed Standard), Internet Engineering Task Force, Apr. 2006, updated by RFC 7527. [Online]. Available: http://www.ietf.org/rfc/rfc4429.txt

[102] S. Routhier, "Management Information Base for the Internet Protocol (IP)," RFC 4293 (Proposed Standard), Internet Engineering Task Force, Apr. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4293.txt

[103] R. Hofstede, A. Sperotto, T. Fioreze, and A. Pras, Networked Services and Applications - Engineering, Control and Management: 16th EUNICE/IFIP WG 6.6 Workshop, EUNICE 2010, Trondheim, Norway, June 28-30, 2010. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ch. The Network Data Handling War: MySQL vs. NfDump, pp. 167–176. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13971-0˙16

[104] P. Velan and R. Krejčíř, "Flow information storage assessment using IPFIXcol," in Dependable Networks and Services. Springer, 2012, pp. 155–158.

[105] M. Grégr, P. Matoušek, T. Podermański, and M. Švéda, "Practical ipv6 monitoring - challenges and techniques," in Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011). IEEE Computer Society, 2011, pp. 660–663. [Online]. Available: http://www.fit.vutbr.cz/research/view˙pub.php?id=9639

[106] M. Grégr, T. Podermański, and M. Švéda, User identification in IPV6 network. Zilina University Publisher, 2012, pp. 5–8. [Online]. Available: http://www.fit.vutbr.cz/research/view˙pub.php?id=9960

[107] L. Polčák, M. Grégr, M. Kajan, P. Matoušek, and V. Veselý, "Designing lawful interception in ipv6 networks," in Security and Protection of Information. Brno University of Defence, 2011, pp. 114–126. [Online]. Available: http://www.fit.vutbr.cz/research/view˙pub.php?id=9620

[108] P. Richter, M. Allman, R. Bush, and V. Paxson, "A Primer on IPv4 Scarcity," ACM SIGCOMM Computer Communication Review, vol. 45, no. 2, 2015, pp. 21–31. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2766330.2766335

[109] M. Mueller, B. Kuerbis, and H. Asghari, "Dimensioning the elephant: an empirical analysis of the IPv4 number market," info, vol. 15, no. 6, 2013, pp. 6–18. [Online]. Available: http://dx.doi.org/10.1108/info-07-2013-0039

[110] Alfa Telecom s.r.o, "Registration of IP and Autonomous systems," 2016, [cited 13.6.2016]. [Online]. Available: http://ip-as.com/index.html

[111] G. Maier, F. Schneider, and A. Feldmann, NAT Usage in Residential Broadband Networks. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 32–41. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-19260-9˙4

[112] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)," RFC 1631 (Informational), Internet Engineering Task Force, May 1994, obsoleted by RFC 3022. [Online]. Available: http://www.ietf.org/rfc/rfc1631.txt

[113] P. F. Tsuchiya and T. Eng, "Extending the ip internet through address reuse," SIGCOMM Comput. Commun. Rev., vol. 23, no. 1, Jan. 1993, pp. 16–33. [Online]. Available: http://doi.acm.org/10.1145/173942.173944

[114] P. Francis, "Selected publications," 2008, [cited 22.4.2016]. [Online]. Available: http://www.cs.cornell.edu/people/francis/publications.html

[115] D. Clark, L. Chapin, V. Cerf, R. Braden, and R. Hobby, "Towards the Future Internet Architecture," RFC 1287 (Informational), Internet Engineering Task Force, Dec. 1991. [Online]. Available: http://www.ietf.org/rfc/rfc1287.txt

[116] P. F. Tsuchiya, "The IP Network Address Translator (Nat): Preliminary Design," in Bell Communications Research, 1991.

[117] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022 (Informational), Internet Engineering Task Force, Jan. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3022.txt

[118] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address Allocation for Private Internets," RFC 1918 (Best Current Practice), Internet Engineering Task Force, Feb. 1996, updated by RFC 6761. [Online]. Available: http://www.ietf.org/rfc/rfc1918.txt

[119] J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space," RFC 6598 (Best Current Practice), Internet Engineering Task Force, Apr. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6598.txt

[120] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," RFC 3489 (Proposed Standard), Internet Engineering Task Force, Mar. 2003, obsoleted by RFC 5389. [Online]. Available: http://www.ietf.org/rfc/rfc3489.txt

[121] F. Audet and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP," RFC 4787 (Best Current Practice), Internet Engineering Task Force, Jan. 2007, updated by RFCs 6888, 7857. [Online]. Available: http://www.ietf.org/rfc/rfc4787.txt

[122] S. Guha, K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh, "NAT Behavioral Requirements for TCP," RFC 5382 (Best Current Practice), Internet Engineering Task Force, Oct. 2008, updated by RFC 7857. [Online]. Available: http://www.ietf.org/rfc/rfc5382.txt

[123] R. Penno, S. Perreault, M. Boucadair, S. Sivakumar, and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements," RFC 7857 (Best Current Practice), Internet Engineering Task Force, Apr. 2016. [Online]. Available: http://www.ietf.org/rfc/rfc7857.txt

[124] G. Huston, "Anatomy: A look inside network address translators," The Internet Protocol Journal, vol. 7, no. 3, 2004, pp. 2–32.

[125] J. Roskind, "Quic: Design document and specification rationale," 2013, [cited 22.4.2016]. [Online]. Available: https://www.chromium.org/quic

[126] Z. Chen, C. Zhou, T. Tsou, and T. Taylor, "Syslog Format for NAT Logging," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-behave-syslog-nat-logging-06, January 2014. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-behave-syslog-nat-logging-06.txt

[127] S. Sivakumar and R. Penno, "IPFIX Information Elements for logging NAT Events," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-behave-ipfix-nat-logging-09, May 2016. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-behave-ipfix-nat-logging-09.txt

[128] C. Donley, C. Grundemann, V. Sarawat, K. Sundaresan, and O. Vautrin, "Deterministic Address Mapping to Reduce Logging in Carrier-Grade NAT Deployments," RFC 7422 (Informational), Internet Engineering Task Force, Dec. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7422.txt

[129] Wikipedia, "Birthday problem — Wikipedia, The Free Encyclopedia," 2016, [Online; accessed 1-July-2016]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Birthday˙problem&oldid=726414796

[130] S. Sanfilippo, "Redis," April 2009, [cit. 20.6.2016]. [Online]. Available: http://redis.io

[131] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani, "Scaling Memcache at Facebook," in NSDI 13. Lombard, IL: USENIX, 2013, pp. 385–398. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/nishtala