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

Towards Utilizing Network Diversity for Future Internet Protocols

vorgelegt von
Juhoon Kim (M.Sc.)

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

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Adam Wolisz, TU Berlin
Gutachterin:	Prof. Anja Feldmann, Ph. D., TU Berlin
Gutachter:	Prof. Don Towsley, Ph. D., University of Massachusetts
Gutachter:	Prof. Dr. Luigi Iannone, Telecom ParisTech
Gutachter:	Dr. Ramin Khalili, Telekom Innovation Laboratories / TU Berlin

Tag der wissenschaftlichen Aussprache: 29.04.2014

Berlin 2014
D 83

Eidesstattliche Erklärung

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

Datum Juhoon Kim (M.Sc.)

Abstract

In the past few decades, the Internet has evolved from a research network to an important infrastructure used by everyone. This means it has changed in almost all aspects including size, topology, and usage. Indeed, due to the ever-increasing number of users and capabilities of end-devices, various types of network diversity have been created in the Internet. For example, users can connect to the Internet through *different service providers* in *various access speeds* using *miscellaneous devices* equipped with *multiple network interfaces* via *two different address spaces (IPv4 and IPv6)*. Such diversity can be found almost everywhere in the Internet. However, to what extent this diversity can be used to improve the Internet is largely uncharted.

Recently, several scalability issues emerged in the Internet, *i. e.*, those related to the large number of network-enabled devices. The exhaustion of the Internet address space is one, the plethora of the routing information within the core Internet is another one. Moreover, the constantly increasing volume of digital content causes significant mismatch between the demand for the high end-to-end performance and the actual capabilities of the Internet infrastructure. These problems hinder the future evolution of the Internet; hence, they are the topics of this thesis.

First, we conduct an assessment study of IPv6 adoption in today's Internet by analyzing a rich set of Internet traffic data collected from a large Internet eXchange Point (IXP). Although the result shows a sharp increase of IPv6 traffic in recent years, the level of IPv6 deployment is still marginal. Thus, the two Internet protocols have to coexist for a while.

Next, to cope with scalability issue caused by the enormous number of routing entries in the core of the Internet, we closely examine the most promising solution for this problem, the Locator/ID Separation Protocol (LISP). In particular, our work focuses on evaluating the performance, security overhead, and resilience of LISP.

By analyzing recent Internet traffic, previous studies found that a major portion of Internet traffic is delivered by HTTP ($> 60\,\%$ [67]) and most of the content providers replicate their content on multiple servers [91]. To this end, we confront the challenge of end-to-end performance by suggesting an easily deployable solution, the Multi-source Multipath HTTP (mHTTP), for improving the HTTP performance based on utilizing various types of network diversity.

Zusammenfassung

In den vergangenen Jahrzehnten hat sich das Internet von einem Forschungsnetzwerk hin zu einer wichtigen, von der Allgemeinheit genutzten Infrastruktur entwickelt. Dabei hat es sich in nahezu allen Bereichen einschließlich seiner Größe, Topologie und Nutzungsweise entscheidend verändert. Die permanent steigende Anzahl von Nutzern und Endgeräten eröffnet Räume für unterschiedliche Aspekte von Netzwerkdiversität. So können sich Endnutzer über *verschiedene Netzwerkprovider* mit *unterschiedlichen Geschwindigkeiten*, über *diverse Geräte*, die mit *multiplen Netzwerkschnittstellen* ausgestattet sind und in *zwei verschiedenen Adressräumen* (IPv4 und IPv6) mit dem Internet verbinden. Diversität findet sich überall im Internet, jedoch inwieweit sie sich nutzen lässt, um das Internet zu verbessern, ist noch weitgehend unklar.

In letzter Zeit traten im Internet Skalierungsprobleme auf, die mit der großen Anzahl von Endgeräten, mit der Erschöpfung des Internet-Adressraums und der Fülle von Routinginformationen im Kern des Internets zusammenhängen. Gleichzeitig erzeugt die konstant steigende Menge an digitalen Inhalten eine signifikante Diskrepanz zwischen der Nachfrage nach hoher End-to-End-Leistung und den gegenwärtigen Fähigkeiten der Internet-Infrastruktur. Da diese Probleme die zukünftige Entwicklung des Internets hemmen, sollen sie Thema dieser Dissertation sein.

Zunächst führen wir eine Studie zur Annahme von IPv6 im heutigen Internet durch. Hierzu analysieren wir einen großen Datensatz, welcher an einem großen Internet-Knoten (IXP) aufgezeichnet wurde. Obwohl das Ergebnis einen starken Anstieg von IPv6-Datenverkehr in den letzten Jahren aufzeigt, ist das Gesamtniveau der IPv6-Nutzung weiterhin marginal. Demnach werden die beiden Internetprotokolle eine Zeitlang weiterhin nebeneinander bestehen.

Um das Skalierungsproblem anzugehen, das durch die enorme Anzahl von Routing-Einträgen im Kern des Internets verursacht wird, untersuchen wir danach mit dem Locator/ID Separation Protocol (LISP) die vielversprechendste Lösung. Dabei steht die Bewertung der Leistung, der Sicherheitsaspekte und der Ausfallsicherheit von LISP im Mittelpunkt.

Vorhergehende Studien haben durch die Analyse aktuellen Datenverkehrs gezeigt, dass der Großteil der Daten im Internet HTTP-basiert ist ($> 60\,\%$ [67]) und dass Inhaltsanbieter ihre Inhalte auf verteilte Server replizieren [91]. Um die End-to-End-Leistung zu erhöhen, schlagen wir mit Multi-Source Multipath HTTP (mHTTP) eine leicht anwendbare Lösung vor, die den Datentransfer verbessert, indem unterschiedliche Aspekte von Netzwerkdiversität verwendet werden.

Publications

Pre-published Papers

Parts of this thesis are based on the following peer-reviewed papers that have already been published. All my collaborators are among my co-authors.

International Conferences

Saucez, D., Kim, J., Iannone, L., Bonaventure, O., and Filsfils, C. **A Local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers.** *In Proceedings of the 11th IFIP Networking* (2012)

Kim, J., Iannone, L., and Feldmann, A. **A Deep Dive into the LISP Cache and What ISPs Should Know about It.** *In Proceedings of the 10th IFIP International Conference on Networking* (2011)

Peer-reviewed Journals

Kim, J., Iannone, L., and Feldmann, A. **Caching Locator/ID mappings: An experimental scalability analysis and its implications.** *Computer Networks. Elsevier* (2012)

Workshops and Poster Sessions

Kim, J., Chen, Y., Khalili, R., Feldmann, A., and Towsley, D. **Multi-Source Multi-Path HTTP (mHTTP): Multi-source Multipath HTTP: a proposal to utilize the path diversity in the Internet.** *Poster at SIGMETRICS* (2014)

Kim, J., Iannone, L., and Feldmann, A.**Can TCP and Locator/ID Seperation get along?** *Poster at CHANGE and Ofelia Summer School* (2011)

Kim, J., Schneider, F., Ager, B., and Feldmann, A. **Today's usenet usage: Characterizing NNTP traffic.** *In Proceedings of the 13th IEEE Global Internet Symposium* (2010)

Technical Reports

Kim, J., Nadi, S., and Feldmann, A. **Watching the IPv6 Takeoff from an IXP's Viewpoint.** *2014-01,ISSN: 1436-9915* (2014)

Kim, J., Khalili, R. Feldmann, A., Chen, Y., and Towsley, D. **Multi-Source Multi-Path HTTP (mHTTP): A Proposal.** *arXiv:1310.2748v3* (2013)

Kim, J., Chatzis, N., Siebke, M., and Feldmann, A. **Tigers vs Lions: Towards Characterizing Solitary and Group User Behavior in MMORPG.** *2013-01,ISSN: 1436-9915* (2013)

Under submission

Parts of this thesis are based on the following paper that is currently under submission.

International Conferences

Peer-reviewed Journals

Workshops and Poster Sessions

Technical Reports

Contents

1

Introduction

The Internet is an information superhighway over which more than one exabyte[1] of data is communicated every day. Through this infrastructure, users can get information on almost every subject, and distant users can communicate with each other faster and cheaper than before via text, voice, and/or video. Indeed, via the Internet, anyone can share his/her ideas/opinions/interests publicly and discuss them without any constraints in time and space. Furthermore, the Internet greatly affects the way people consume media entertainment, *e. g.*, movies, music, games, and books; hence, it has created new types of online business models, *e. g.*, content provisioning, content delivery, Infomediary[2], and online advertising.

Since this revolutionary invention has emerged from the laboratory, it did not stop growing in size as well as in range. Today, the Internet links more than 2.5 billion users [130] and more than 9.6 billion electronic devices [129] worldwide. These numbers are expected to grow even more at increasing rates. The huge success of the Internet comes with some penalties and opportunities, namely, network complexity and various types of diversity. For example, there are plenty of choices of network-enabled devices on the market, many devices are equipped with multiple homogeneous or heterogeneous network interfaces (*e. g.*, Ethernet, WiFi, 2G/3G, and LTE), and/or users connect to the Internet via different service providers at different access speeds. This highly ubiquitous Internet diversity provides great opportunities to researchers in the sense of taking advantage of it for the Internet's future evolution.

[1] A billion gigabytes
[2] A compound word from information and intermediary

In recent years, the continuous growth of the Internet and the increasing level of Internet diversity brought several problems to the surface. First, the IPv4 address space of the Internet is almost exhausted. Second, the explosive growth of end-devices, fine-grained address space, and multihoming overburden routers in the core Internet with an excessive number of routing entries. This can lead to performance degradation of routers since each router for each packet has to do a longest prefix match based on the entire routing space. Third, the demand for excellent end-to-end performance is hard to meet since the volume of digital content is increasing at higher rate than the infrastructure bandwidth. For instance, the file size of recent high-quality video types (*e. g.*, 1080p, Blu-ray, or 4k movies) is ranged from 8 GB to 100+ GB.

To address these problems, this thesis covers three aspects. First, in order to see how swiftly the Internet moves towards the next version of Internet Protocol (namely, IPv6), we assess the level of IPv6 adoption from the viewpoint of a large European Internet eXchange Point (IXP). Second, we examine the performance and trade-off of the most potential solution of the core routing table saturation problem, namely, Locator/ID Separation Protocol (LISP). Third, we design, implement, and evaluate a novel socket architecture, Multi-source Multipath HTTP (mHTTP), to improve the performance of end-to-end content delivery by utilizing various types of network diversity in today's Internet.

1.1 Network Diversity in Today's Internet

In this section, we explore different types of network diversity that motivated us to study their implications and opportunities.

1.1.1 Address Space Diversity

The Internet is currently in a transition period with regard to its basic network protocol, *i. e.*, from IPv4 to IPv6. Even though this transition started about two decades ago, the complete change of the Internet protocol takes considerable time. Worse yet, recent measurement studies [23, 102, 146], including this thesis, find that traffic routed via IPv6 addresses is still marginal (less than 1 % of the total traffic) at their vantage points. Nonetheless, more and more network operators configure their routers and servers to be reachable via IPv6 addresses, *e. g.*, a dual-stack network. To this end, it is expected that two different address systems will coexist in the Internet for a while — a prediction that motivates us to study the current situation of IPv6 deployment in this thesis.

1.1.2 Interface Plurality

Since the cost of a Network Interface Card (NIC) is comparably low, deploying several of them is an affordable option for most of today's network devices. However, currently, the actual benefit of having multiple interfaces is derived from the network resilience rather than bandwidth increase. This is mainly due to the fact that a network connection cannot be shared across multiple interfaces.

Existing engineering practice and academic research tries to overcome this limitation by combining multiple network interfaces into one logical interface (also known as channel bonding or link aggregation). Along with the standard, *i. e.*, 802.1AX-2008 [118], there are also a large number of proprietary technologies (EtherChannel [124], Port Aggregation Protocol (PAgP) [135], Multi-link trunking [121], etc.) for bundling multiple network interfaces. However, most of these approaches are designed to operate between two network devices and require the same network equipment on both end-devices.

Thus, these technologies do not necessarily enhance the performance of the end-to-end communication across the Internet since the maximum network throughput is still limited by the slowest link on the communication path.

1.1.3 Interface Heterogeneity

Today, most end-devices have not only multiple interfaces, but also heterogeneous network interfaces. For laptops and desktop computers, it is not unusual to have at least one Ethernet interface and an additional wireless interface. For hand-held mobile devices, *e. g.*, smart phones and tablets, almost all products on the market are equipped with a WiFi interface, 3G, and/or LTE (Long Term Evolution) interfaces.

However, current end-devices do not use their available network interfaces in parallel, but rather in serial order by shifting communication from one interface to another. In this way, resilient data transmission can be achieved, but the transfer rate is still limited by the bandwidth of one network interface (thus, link) that is used.

There are a handful of studies that aim at enabling the concurrent use of multiple heterogeneous (or homogeneous) network interfaces for a single network communication, *e. g.*, Multipath TCP (MPTCP) [31, 96], SCTP [109], pTCP [43], mTCP [148]. Unlike the techniques mentioned in Section 1.1.2, most of these are transport layer solutions, thus, they have to involve the end-systems. Indeed, they require support on both end-systems. How to tackle this problem with only changing the software on one system is an open research topic.

For the sake of simplicity, we refer to the *interface plurality* and the *interface heterogeneity* together as the *interface diversity* in the rest of this thesis.

1.1.4 Upstream Link Diversity

Upstream link diversity comes on the back of the interface diversity when multiple network interfaces of a device are connected to independent upstream service providers, *i. e.*, multihoming. Nowadays, most end-devices have multiple points of connection to the Internet and even can use them at the same time. However, before hand-held mobile devices became popular, multihoming was mostly applicable to networks for the purpose of improving reliability.

Indeed, network-level multihoming has been one of the main reasons for saturating the core Internet routing table. According to CIDR Report [123] and the study of Clayton *et al.* [18], the growth of the global routing table is exponential and the recent trend exhibits about 25 % of growth per year. Meng *et al.* [72] (2005) and Clayton *et al.* [18] (2009) respectively find that 20 % and 37 % of all global routing information is due to multihoming. Given that the cost (in terms of time and CPU/memory consumption) of the longest prefix match in the routing table is proportional to the size of the table, it has become a serious problem from the performance perspective. Fortunately, the research community has proposed several solutions based on the address space decoupling, *e. g.*, ILNP [8], HAIR [29], Six/One [116]. Currently, the Locator/ID Separation Protocol (LISP) [26] stands out among its sibling protocols for its deployment level [132, 133] and the size of its development community [125, 134, 142].

1.1.5 Path Diversity

A combination of multiple types of diversity, particularly, interface diversity and upstream link diversity (multihoming), introduces multiple pathways (*i. e.*, zonally or fully disjoint paths) between two end-systems which we refer to as *path diversity* in this thesis. From an end-device's point of view, having multiple paths towards the Internet does not only lead to network resilience, but also sheds light on the question: How can we improve the performance of the end-to-end content delivery without upgrading the physical media of the current Internet infrastructure? For example, Multipath TCP (MPTCP) [31, 96] is a widely studied mechanism for answering this question based on the availability of multiple paths. More precisely, MPTCP enables a TCP connection to split into multiple sub-flows and deliver segments via different interfaces, thus over different paths. In this way, MPTCP achieves an aggregated network throughput from multiple available paths. However, even with such a promising feature, we may not be able to experience the benefits from using MPTCP until the protocol is widely deployed. This is due to the design of MPTCP that requires an agreement and the protocol specific negotiation between two end-systems.

1.1.6 Data Source Diversity

In the global Internet, redundancy is often the simplest, but still the most effective strategy for reducing the end-to-end delay and/or for distributing the network load to multiple entities. A Content Distribution Network (CDN) is a good example for reducing the delivery time between content and its consumers by locating identical copies of content across the globe.

CDNs typically use the Domain Name System (DNS) in order to suggest the location of a replica to users, however, the quality of the server selection often suffers from using third-party DNS resolvers, *e. g.*, GoogleDNS and OpenDNS. Therefore, many academic/industry researches, that try to improve the performance of the CDN infrastructure, is focusing on the server selection mechanism [90, 107].

However, the availability of content in multiple locations offers more than just a short network delay when it is combined with the content chunking mechanism such as HTTP's *byte range request*. More precisely, fetching content simultaneously from multiple data sources by splitting it into several data chunks can lead to throughput aggregation of distinct paths between a client and multiple data sources. Indeed, we show that the performance of content delivery can be significantly increased by this technique.

1.2 Main Contributions

The main contributions of this thesis are ranging from an analysis of today's Internet traffic to the performance prediction of potential future Internet protocols. In this section, we summarize the major contributions of this thesis.

1.2.1 Characterization of Today's Internet Traffic

In the first part of this thesis, we explore the usage pattern of the Internet by thoroughly analyzing passively collected Internet traffic. A vantage point for observing network traffic is an important factor in such studies since the significance of analyzing Internet traffic often lies in understanding the collective behavior of Internet users. To this end, our traffic analyses are performed based on the traffic traces collected from a large European Internet Service Provider (ISP) and one of the biggest Internet eXchange Points (IXPs).

To be specific, by analyzing ISP traffic, we confirm the survival (or revival) of the Network News Transport Protocol (NNTP) whose life is generally believed to be finished. Based on the same traffic data, we also unveil the user behavior and the pattern of network traffic generated by World of Warcraft as a representative of today's Massively Multi-player Online Role Playing Games (MMORPGs). To achieve

this, we develop protocol analyzers (*NNTP Analyzer* and *WoW Analyzer*) using Bro-NIDS [87] as the codebase.

Furthermore, by evaluating IXP traffic, we assess the level of IPv6 adoption and show its upward tendency in today's Internet. This work mainly focuses on 14-month worth traffic data between two global IPv6 events, *i. e.*, *World IPv6 Day* and *World IPv6 Launch Day*, in order to understand the change in global traffic pattern.

1.2.2 Evaluation of the Locator/ID Separation Protocol

The second part of this thesis evaluates the implication and performance of deploying the Locator/ID Separation Protocol (LISP) by modifying today's Internet architecture. Especially, our evaluation focuses on the mapping cache, *i. e.*, the key component of LISP in which bindings of two different name spaces are stored, in perspectives of scalability, reliability, and security.

Accordingly, we carry out experiments for answering the following questions:

- How scalable and beneficial (or detrimental) is the LISP Cache?
- What is the cost for improving the security level of LISP?
- What happens if one of the LISP routers fails in a multihomed network?
- Will the TCP communication suffer from the initial address mapping of LISP?

Throughout this study, most of the experiments are done by using our own LISP Cache emulator. The emulator is specially designed to reproduce the behavior of the LISP Cache based on the pattern of today's Internet traffic. In order to take today's Internet usage into consideration, real-world packet-level Internet traffic is used as the input for the emulator.

1.2.3 Design & Implementation of the Multi-source Multipath HTTP

Finally, in the third part of this thesis, we design and implement Multi-source Multipath HTTP (mHTTP), a novel socket architecture that aims at decreasing the content delivery time by offloading the transmission onto multiple paths utilizing network diversity and data source diversity.

mHTTP is a solely receiver-oriented solution that does not require any type of modification at the server-side infrastructure. Moreover, modifications at the receiver are limited to the existing socket APIs, thus applications do not need to be changed at all. By virtue of such uncomplicated design principles, mHTTP can be seen as a quickly deployable solution for boosting the performance of end-to-end content delivery.

Our prototype implementation proves that the concept of the Multi-source Multipath approach works fine with the current Internet architecture and performs remarkably better than a single path HTTP. Moreover, via a measurement study, we find that mHTTP performs as good as Multipath TCP (MPTCP), thus, mHTTP can also be seen as a viable alternative of MPTCP for HTTP traffic.

1.3 Thesis structure

This thesis manuscript is organized as follows:

Chapter 2 provides an overview of topics, data sets, and analysis tools used all through this thesis.

Chapter 3 presents measurement studies on three different protocols in order to understand the characteristics of today's Internet traffic and the collective behavior of Internet users. This work includes three different network protocols, namely, Network News Transport Protocol (NNTP), World of Warcraft, and IPv6.

Chapter 4 analyzes the implication of deploying the Locator/ID Separation Protocol (LISP) with focuses on the EID-to-RLOC mapping cache. This work evaluates LISP Cache from perspectives of the scalability, performance, security overhead, and reliability.

In Chapter 5, we extensively discuss the concept of the Multi-source Multipath HTTP and present its prototype implementation. This includes a thorough performance analysis of mHTTP and a comparison with MPTCP using the content download time as metric.

Chapter 6 concludes this thesis by summarizing the main contributions and by outlining potential future work.

2
Background

This chapter aims at acquainting readers with an overview of the topics discussed in this thesis.

2.1 Traffic Characterization

Studying characteristics of today's Internet traffic is, on the one hand, a relevant academic/industrial activity for keeping track of the Internet evolution and for developing the technology roadmap of the Internet infrastructure. On the other hand, it is valuable for estimating the implication and performance of the future Internet infrastructure.

2.1.1 Passive Measurement

Internet traffic measurement is a broadly spanned field of study in computer networking. Indeed, due to the Internet's complexity and diversity produced by an enormous number of users, understanding the behavior of end-users is one of the most emphasized tasks. Passively monitoring the pattern of Internet usage is the most appropriate technique as it does not interfere with the normal Internet usage. This type of measurement studies are often interested in, but not limited to, finding out throughput, time-of-day pattern, the length of communication, the size distribution of content, and the most influential content types and/or applications.

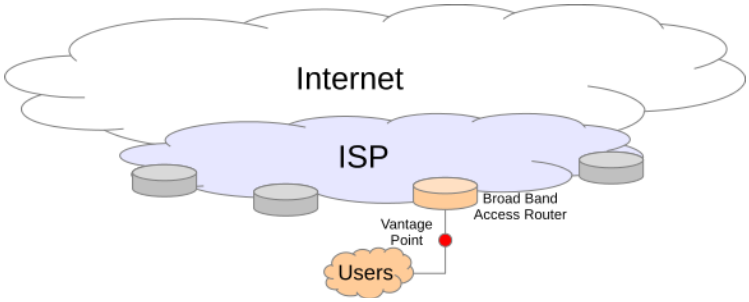


Figure 2.1: Our ISP vantage point

Accordingly, the monitoring location and the number of end-users behind the monitoring point is an important criterion to estimate the representativeness of the result of such studies. This type of experiment is commonly referred to as the *Passive Measurement* in contrast to the *Active Measurement* that is used to find out the performance characteristics by injecting experimental traffic into the Internet.

2.1.2 Vantage Points

Thanks to our excellent collaboration with a number of academic/industrial researchers, we were able to conduct our experiments based on several sets of anonymized Internet traffic collected from two different vantage points.

Internet Service Provider (ISP)

The ISP vantage point is one of the Point of Presences (PoPs) located within a tier-1 European ISP. The monitor, equipped with Endace monitoring cards, operates at the broadband access router connecting customers to the ISP's backbone. We count more than 20,000 DSL lines behind this vantage point. All traffic traces collected from this vantage point are packet-level, highly anonymized, and not sampled. The throughput of DSL lines ranged from 1Mbps to 16Mbps for downstream traffic, and from 0.2Mbps to 1Mbps for upstream traffic. We refer to Figure 2.1 for better insight into our ISP vantage point.

Internet eXchange Point (IXP)

An Internet eXchange Point (IXP) is a physical infrastructure for interconnecting Internet Service Providers (ISPs) and ASes in order to reduce the end-to-end latency and the cost of transit traffic. A simplified topology of an IXP is illustrated in Figure 2.2. Through such infrastructures, its member ASes can establish peering (or transit) relationships with other member ASes. The volume of daily network traffic

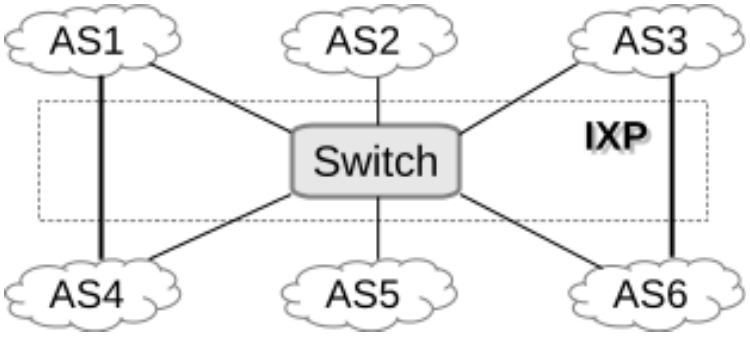


Figure 2.2: A simplified IXP topology

at an IXP depends, amongst other things, on the number of networks connected to the IXP. The mentioned IXP has more than 400 members which makes it one of the largest IXPs in Europe.

Due to the immense amount of traffic volume, it is considered to be virtually impossible to capture and store all packets in the IXP vantage point. To this end, packet sampling, with the rate of 1:16k, is implemented on the switch. Moreover, we only record the first 128 bytes of the sampled packets. That is still sufficient enough to include all relevant information, *e. g.*, IP headers, tunneling headers, and transport layer protocol headers.

2.1.3 Packet Anonymization

For privacy reasons, the anonymization of traffic is one of the most important tasks to carry out before conducting any measurement studies. In fact, in order to eliminate any chance of information leakage, the anonymization shall be performed before a packet is stored in the disk.

In this regard, all packets in our vantage points (ISP and IXP) are pipelined to the anonymization process before being stored in a disk for the purpose of muddling IP addresses and the confidential information within the payload, while the consistency of IP addresses and the size of the network prefix are preserved.

For the ISP traffic collection, we use Bro's integrated function for the anonymization. In the case of the IXP traffic collection, we developed software that anonymizes network prefixes obtained from a publicly available BGP table and the first k bits of an IP address with the same hash algorithm where k is the length of the network prefix of the IP address.

2.1.4 Analysis Tools

The traffic we collect from the two vantage points is a stream of continuous bytes, thus, it is necessary to develop software that extracts meaningful information about a target protocol in a human-readable text form. Software that does this task is commonly referred to as a protocol analyzer.

Indeed, the development of a protocol analyzer, *e. g.*, *NNTP Analyzer*, *WoW Analyzer*, and *IPv6 Analyzer*, must be based on a thorough understanding of the protocol. Coping with the underlying network technologies, *e. g.*, TCP/UDP for application specific-protocols and transition techniques for IPv6, is another time-consuming task. Therefore, we make the best use of existing software in order to build our own protocol analyzer on top of that. In this section, we describe some of the most popular network monitoring tools and traffic analysis tools we used for building our analyzers.

TCPDUMP and TShark

TCPDUMP [140] is a popular packet-level traffic monitoring tool which is often used for translating recorded or live network packets into the human-readable text format. Although TCPDUMP is an effective and scalable packet-level traffic parser, it does not have a built-in intelligence for identifying application-level protocols. In contrast to TCPDUMP, *TShark* ships a broader set of protocol analysis features, but it is slow and does not scale with a large volume of data. Thus, TShark is often used as a benchmarking tool in the beginning phase of the analyzer development.

sFlow tools

sFlow [138] is a network traffic monitoring technology designed to provide a scalable solution for collecting Internet traffic at low cost. Since sFlow is robustly operable in high speed networks, it is commonly used in highly concentrated networks such as an IXP. In order to capture and monitor traffic in our IXP vantage point based on the sFlow technology, we use *sFlow tools*, *i. e.*, an analysis toolkit developed by *InMon* [137].

Bro NIDS

For developing NNTP Analyzer and WoW Analyzer, Bro NIDS [87] is used as the basecode. Bro is a Network Intrusion Detection System (NIDS) designed to observe network traffic and to find abnormal traffic pattern in real-time (and/or from recorded traffic). The abnormality definition of the protocol or traffic can be customized by the operator by writing a policy script. Based on Bro's design choice that

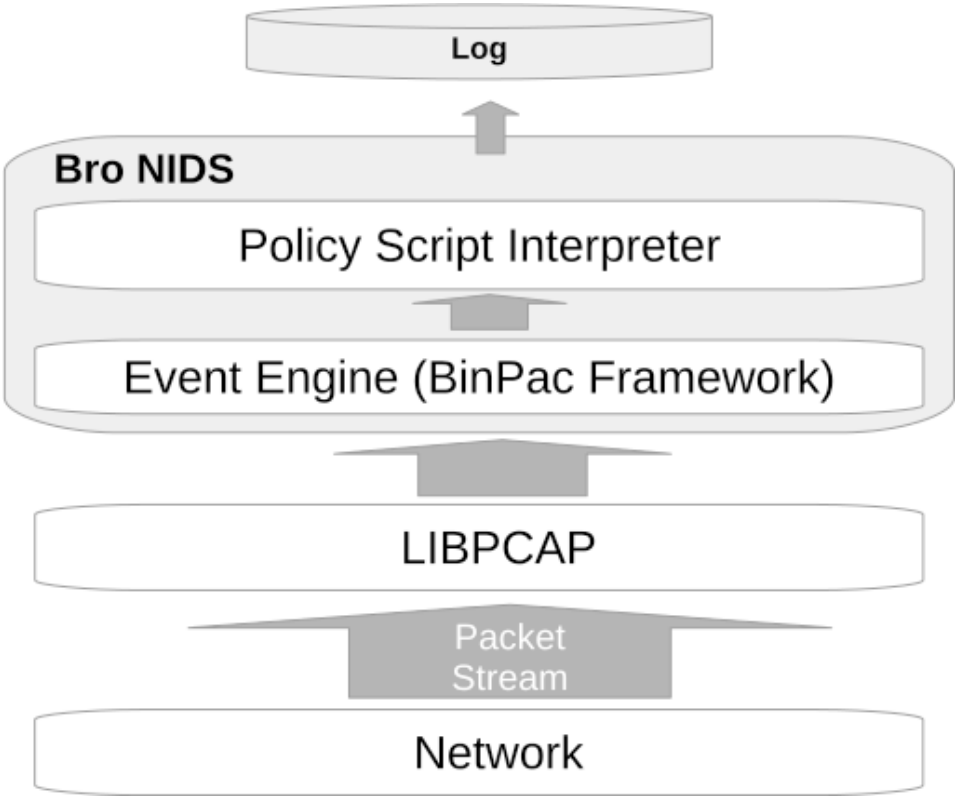


Figure 2.3: The traffic analysis process of Bro NIDS

allows operators to configure the policy independently from its core system, we can make good use of its capabilities for building our own analyzers. Figure 2.3 illustrates the traffic monitoring process of Bro NIDS. As the thickness of arrows indicates, the volume of data decreases in each layer of the monitoring process by filtering (LIBPCAP), identifying the application (event engine), and defining specific actions on events (policy script interpreter). The main reasons for choosing Bro as the basis of our analysis tool can be explained by following the three unique features.

- Bro is designed in such a way that TCP/UDP protocol analyzers are hierarchically separated from application layer protocol analyzers, so that developers do not need to deal with the complexity that transport layer protocols have (*e. g.*, TCP stream re-assembly).

- Bro supports a specialized development framework for a protocol analyzer, namely BinPac [85]. The code written in BinPac script language can be eventually translated into *C++* code by the *protocol parser generator* shipped with the BinPac framework. The development process is greatly eased by this framework.

- Bro provides an indigenous protocol identification mechanism, *i. e.*, Dynamic Protocol Detection (DPD) [24]. DPD allows the analyzer to identify the protocol used in a network connection by matching the behavior of the connection with known characteristics of the protocol. More precisely, the analyzer identifies a potential protocol in the beginning of the connection based on traditional classification methods (*e. g.*, signature and/or well-known network port numbers) then confirms or denies the decision depending on the connection's further behavior (*e. g.*, a handshake between the server and the client, a request/response pair, etc.). This unique feature of Bro greatly improves the accuracy of the traffic classification.

2.2 Content Delivery in Today's Internet

Next, we give an overview of HTTP-based content delivery in today's Internet. We aim at taking advantage of multiple types of diversity, *e. g.*, data source diversity, path diversity, and interface diversity, via these content delivery infrastructures.

2.2.1 High Performance Demand for the Content Delivery

Popular forms of digital content and their encoding technologies have changed over the course of time. In the beginning of the Internet age, text and small-sized sound/image objects were the most desirable forms of digital content and predominantly transferred by application protocols such as Network News Transfer Protocol (NNTP), Simple Mail Transfer Protocol (SMTP), and Hyper-Text Transfer Protocol (HTTP). Although these protocols are designed to deliver text-based information, thanks to binary-to-text encoding techniques, binary data can be embedded in the payload and be transferred along with the text information. While such protocols are still popular today, characteristics of popular content have changed a lot. Now, binary bulk content is the biggest player in Internet traffic and the quality of digital media increases every day. Accordingly, the average volume of content increases and the delivery time of content becomes longer since the bandwidth of the existing infrastructure does not increase as fast as the volume of content. Therefore, meeting the user's demand for the high end-to-end delivery performance is a matter of common interest between research and industry communities.

To cope with this challenge, a number of business entities have deployed redundant content servers across the world in order to serve content from the best possible server at user's location. Today, such an infrastructure is positioned in the middle of the content delivery business and the major fraction of digital content is transferred through this type of overlay network.

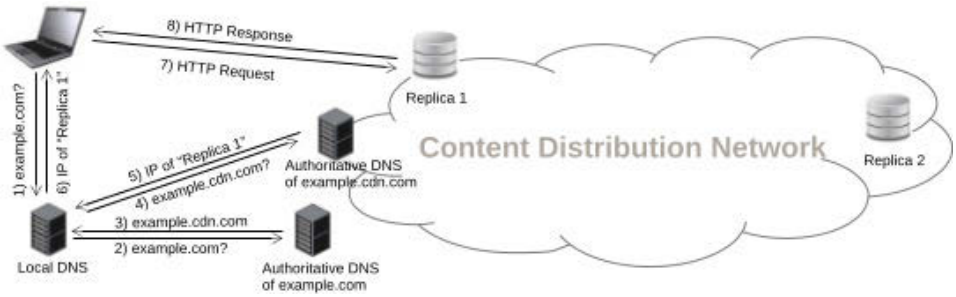


Figure 2.4: A brief overview of the server selecting operation in a CDN

2.2.2 Content Distribution Infrastructures

Next, we outline the high-level concept of popular content delivery infrastructures in today's Internet.

Content Distribution Networks (CDNs)

In general, content is delivered from a server to a receiver across the world through multiple network entities, *e. g.*, access networks, ISPs, IXPs, and backbone networks. In that case, the performance of the content delivery, in terms of time, is restricted by the narrowest or the most congested link between the two end-systems (server and receiver). On the contrary, by replicating content in multiple data centers, a Content Distribution Network (CDN) offers the receiver a nearby copy of content to avoid a bottleneck link between the two end-systems. With redundancy, CDN can also provide high content availability to users.

As illustrated in Figure 2.4, a typical server selection mechanism of CDNs, *e. g.*, *Akamai*, is based on mapping a URL of the original content server on a surrogate URL (hereafter called *alias*) provided by the CDN operator (from 1 to 3 in the figure). Such a mapping is done by the Canonical Name (a.k.a., CNAME) record specified in [74,75]. Through this mapping, a user's DNS request is changed from the name translation of the original content server to that of the alias (4 in the figure). Since the mapping between the alias and an IP address is under the control of the CDN operator, CDN can suggest a location (IP address) of a replica based on the IP address of the user's local DNS (5 and 6 in the figure).

One-Click Hosters (OCHs)

A One-Click Hoster (OCH) is an online file storage service that allows users to upload, download and/or share files through a simple web-based user interface, *e. g.*, *RapidShare*, *Uploading*, *FileFactory*, and *Sendspace*. Due to the easy interface, high

storage volume (*e. g.*, > 10GB/file), high delivery performance [7], and piracy of digital content [62], the popularity of OCHs is gradually increasing. According to reports of *ipoque* [105,106], OCHs are responsible for up to 10 % of the total Internet traffic in several vantage points. Moreover, Poese *et al.* [90] find that about 15 % of the total Internet traffic in their vantage point is generated by the content delivery between OCHs and users. We refer readers to the work of Antoniades *et al.* [7] and Lauinger *et al.* [62] for gathering more information about the ecosystem of OCHs.

Video Streaming Services

A streaming-over-HTTP is a common method for transporting video content. Today, most of the popular video streaming providers such as *YouTube* and *NetFlix* serve their content in this way. According to [64], *YouTube* alone accounted for more than 20 % of the total Internet traffic in Europe in 2012. Furthermore, Maier *et al.* [67] find that more than 25 % of the total HTTP traffic in their vantage point carry video content. Accordingly, to meet the users' growing demand for service quality, video streaming providers adopted CDN-like infrastructure for distributing their content.

2.2.3 Diversity in CDIs

Content replication has widely proven to be a powerful solution for satisfying the user's performance demand for the delivery of content. Although the cost of content distribution is inevitable in such a mechanism, its redundancy feature may be more effectively utilized. For example, the simultaneous download of a single object from multiple data sources can be seen as one means to utilize such an infrastructure in more effective manner.

There are video streaming protocols that try to take advantage of replication of streaming contents provided by CDNs [1, 54, 112]. Moreover, BitTorrent is a well-known application that takes advantage of content replication among its users [19]. However, these protocols are software dependent and cannot be generally used.

For this end, in Chapter 5, we design and develop a novel socket architecture that realizes this idea, the Multi-source Multipath HTTP (mHTTP). mHTTP is solely receiver-oriented solution and does not rely on any specific software.

In the case of an OCH, data source diversity is accommodated in a slightly different form. Like regular CDNs, an OCH typically operates multiple servers that hold identical copies of a file. However, OCHs differ from CDNs on the level of server distribution and the way of the replica selection. OCHs tend to locate their servers in a more centralized manner, *e. g.*, in one data center or in a small number of data centers. How OCHs choose the replica to serve the user's request is unclear, but most of them do not seem to use the complicated server selection mechanism, *e. g.*, a DNS-based mapping technique such as the one used by *Akamai*. Besides the source

diversity due to multiple servers, it is common to find the same file hosted in different OCHs or even in the same OCH.

2.3 Internet Redesign and the Locator/ID Separation Protocol

In order to explore the concept of the future Internet, we first provide an overview of current Internet problems that the future Internet needs to solve and explore design principles of the future Internet that are debated within the research community.

After that, we outline one of the most pressing scalability problems of the core Internet and show how a core/edge decoupling mechanism, as one of the fundamental building blocks of the future Internet architecture, addresses this problem.

2.3.1 Internet Redesign

The current Internet faces several challenges, *e. g.*, security, mobility, and scalability, which were unforeseen in the early stage of its evolution. Furthermore, increasing demand for the high Quality of Service (QoS) and availability incite the research community to rethink the design of the current Internet architecture.

Although these issues and challenges are widely recognized, it is not straightforward to build a new Internet architecture. The main difficulty lies in changing the Internet architecture without causing the economic loss in the society and the interruption of the Internet use.

In the research community, two design principles for the future Internet have been discussed [28]. One is *Clean Slate* design, *i. e.*, the complete reinvention of the Internet, and the other one is *Incremental* design, *i. e.*, the gradual modification of the Internet.

Indeed, the Internet has evolved through the incremental method mainly due to the above-mentioned concerns. For example, as we discuss in this thesis, core/edge decoupling techniques commonly adopt the incremental method as their deployment strategy.

2.3.2 Scalability Issue with regard to routing in the Core Internet

The Border Gateway Protocol (BGP) is the *de facto* standard protocol for the inter-domain routing in today's Internet. In order to correctly deliver packets to end-hosts across the Internet, the most aggregated part of the Internet (a.k.a., the core Internet or default free zone) needs to store complete knowledge of global routing paths. Due to the fine-granined subdivision of the address space and the upstream link diversity

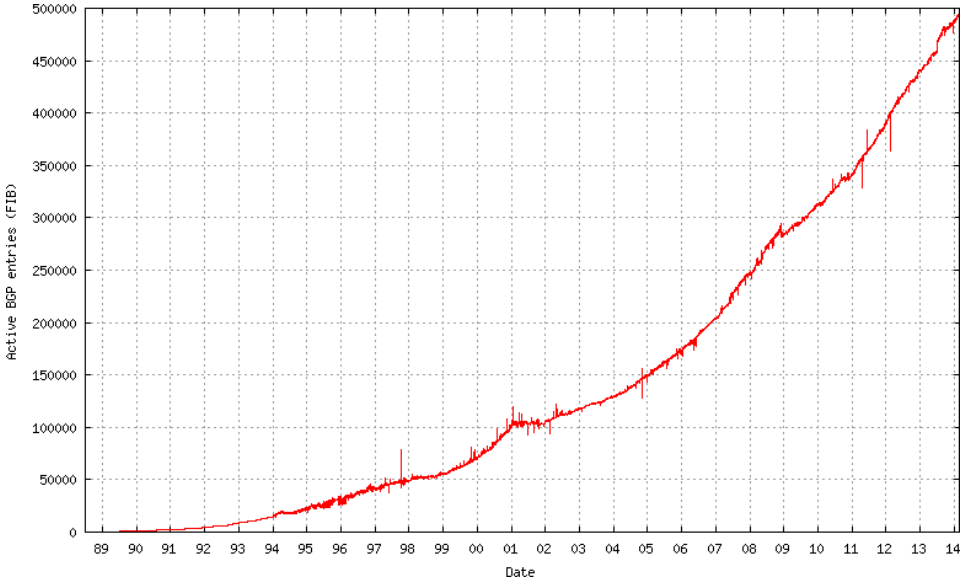


Figure 2.5: The growth in the number of BGP entries (source: potaroo.net)

(multihoming), the number of entries in the routing table within the core Internet increases at an enormously rapid rate (see Figure 2.5). Therefore, routers in the core Internet suffer from handling such an excessive number of BGP entries. It is crucial to note that the point at issue does not only lie in the upgrade cost of the hardware (*e. g.*, memory/CPU), but also in the time taken for performing the Longest Prefix Match (LPM) operation.

To this end, the research community has started to look for alternative routing architectures based on the concept of decoupling the core network from edge networks. Among different proposals, the Locator/ID Separation Protocol (LISP) is widely acknowledged as the most potential candidate for solving the problem. Therefore, in our study, we choose LISP as the representative protocol of the core/edge (or Locator/ID) decoupling technology.

2.3.3 An Overview of LISP

The main idea of LISP is to split the IP address space into two orthogonal spaces, one used to identify the end-hosts, and one used to locate them within the Internet topology. Such a re-use of the IP addresses makes LISP incrementally deployable, hence the interest of the IETF in it. LISP is meant to be used on the border routers of stub networks, such as in the scenario presented in Figure 2.6. Stub networks internally use an IP prefix, called EID-Prefix as for End-point IDentifier, which is part of the ID space. This prefix does not need to be globally routable, since it is used only for routing in the local network. On the contrary, the core of the Internet,

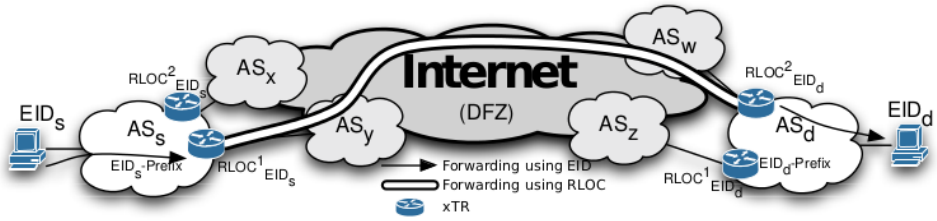


Figure 2.6: Example of LISP deployment and communication.

known as Default Free Zone (DFZ), will use globally routable IP addresses that are part of the locator space. In particular, the IP addresses used by the stub networks' border routers on their upstream interfaces (toward the provider) are part of such a space and represent the Routing LOCators (RLOCs), since they allow one to *locate* EIDs in the Internet. The binding between an EID-Prefix and the set of RLOCs that locate it is a *mapping*. By introducing the above-described separation, EID-Prefixes are no longer announced in the DFZ, allowing a size reduction of BGP's routing table. Further information on the benefits of deploying LISP can be found in the work of Quoitin *et al.* [93] and Iannone *et al.* [47].

When communication is initiated in the context of LISP, end-hosts generate normal IP packets using EIDs as source and destination addresses, where the destination EID is obtained, for instance, through a DNS query. LISP then tunnels those packets from the border router of the source network to the border router of the destination network, using the RLOCs as source and destination addresses in the outer header. The tunneling is done by using an IP-over-UDP approach, with the addition of a LISP-specific header between the outer UDP header and the inner (original) IP header[1]. In the LISP terminology, the border router in the source network performing the encapsulation is called Ingress Tunnel Router (ITR), while the one in the destination network, performing the decapsulation, is called Egress Tunnel Router (ETR). In general, they are just named xTRs.

2.3.4 LISP Database and LISP Cache

To perform tunneling operations the routers use two data stores, namely the LISP Database and the LISP Cache. The LISP Database stores mappings that bind the local EID-Prefixes (*i. e.*, inside the local network) to a set of RLOCs belonging to the xTRs deployed in the network. The purpose of the LISP Database is two-fold. For outgoing packets, if a mapping exists for the source EID it means that the packet has to be LISP encapsulated and the source RLOC is selected from the set of RLOCs associated to the source EID-Prefix. For incoming packets, if a mapping exists for the

[1]The LISP header contains additional information on RLOCs' reachability and traffic engineering. Details on this header can be found in the protocol specification ([26], [48]).

destination EID, then the packets are decapsulated. The LISP Database is statically configured on each xTR. Its size is directly proportional to the number of the EID-Prefixes and xTRs that are part of the local network. Due to its static nature and limited size, the LISP Database does not present any scalability issue and is not a critical component of the LISP architecture.

The LISP Cache temporarily stores the mappings for EID-Prefixes that are not part of the local network. This is necessary to correctly encapsulate outgoing packets, in particular to select the RLOC to be used as destination address in the outer header. Mappings are stored only for the time that they are used to encapsulate packets, otherwise, after a timeout, they are purged from the cache. As such the LISP Cache is the key component of the LISP architecture, and determining its implications on the existing Internet infrastructure is an important task to be done.

3

Today's Internet

One constant in the Internet during the last 10 years has been the steady growth of the Internet traffic by more than 50 % each year [35, 83]. However, the traffic mix has undergone substantial changes. Initially, protocols such as FTP, SMTP, and NNTP were popular. Then, around 1994, HTTP entered into the picture. Most of the protocols from this early Internet age were designed for exchanging text-based information. A little while later, multimedia data such as video, audio, and pictures as well as RAR archives became the dominant content [49, 61, 67, 106] types in the Internet. To transfer such binary content, P2P protocols such as Napster and Gnutella became popular but were later overtaken by eDonkey and BitTorrent. According to [105], the major fraction (around 70 %) of the Internet traffic was generated by these P2P applications in 2007. Yet, several recent ($>$ 2009) traffic studies report that there has been another substantial change in the application mix [49, 61, 67, 106]. These studies show that HTTP is responsible for more than 60 % of the total Internet traffic and that NNTP generates a significant amount (up to 5 %) of traffic. Moreover, Massively Multi-player Online Games (MMOGs) have started to gain popularity. The number of users of such games shows a super-linear growth tendency since the beginning of the 2000s. Today, it is estimated that the number of Internet users subscribed to MMORPGs is more than 22 million worldwide [144].

Unlike such drastic changes on the application layer, the underlying network protocol (*i. e.*, IPv4) has been hardly modified. Considering that this protocol was invented about three decades ago, Internet Protocol (IP) may be seen as one of the slowest Internet technologies to innovate. In fact, modification of IP was not a pressing matter during the evolution of the Internet. However, the situation has significantly

changed since the last block of the IPv4 addresses has been assigned to the RIRs in mid-2011. First, the Internet Assigned Numbers Authority (IANA) announced that they assigned the last block of IPv4 addresses, then Asia Pacific Network Information Centre (APNIC) reported that they reached the final stage of IPv4 exhaustion [119]. After all, major network operators decided to make effort together for deploying IPv6. As a result, several network operators report that IP traffic shows a small movement towards IPv6 recently. Therefore, it is of great interest for researchers to know the current situation of the IPv6 adoption.

In this chapter, we first present measurement studies of two application protocols, *i. e.*, NNTP and World of Warcraft, that have been understudied. The two analyses will help us understand the characteristics of today's Internet usage. After that, we present our assessment study of IPv6 adoption by analyzing a set of traffic traces collected from an Internet eXchange Point (IXP).

3.1 Today's Usenet Usage

We choose the Network News Transfer Protocol (NNTP), *i. e.*, the underlying application protocol of the Usenet, as the starting point of our measurement study. The reason behind this choice is based on reports of its noticeable popularity in IPv4 traffic [67] and in IPv6 traffic [102]. Exploring NNTP first will also help readers to better understand the results of the IPv6 traffic analysis carried out in Section 3.3.

Usenet evolved from its UUCP (RFC 976) architecture to a worldwide distributed Internet discussion system in which users read and post public messages to one or more categories, known as newsgroups, via NNTP. These messages are called articles or posts, and collectively are referred to as news. Usenet is similar to bulletin board systems and as such is the precursor to various Internet forums.

The Usenet is realized across a constantly changing set of servers that store and forward messages among each other. Since Usenet servers do not have unlimited disk space to hold every article that was ever posted, the oldest articles are deleted as new articles arrive. Usenet distinguishes between binary groups and text based groups since the storage and bandwidth requirements for binary groups are substantial. Due to its size, a single binary message may squeeze out several hundreds of text-only postings and its download may impose significant bandwidth cost for the server provider. Therefore, Usenet administrators started to exclude binary groups on their publicly accessible news servers. Moreover, the traditional Usenet features (*e. g.*, text-based groups) are offered by forums and mailing lists and are integrated into blogs and online social networks (OSNs). As a consequence, NNTP became less desirable and eventually its usage declined as such alternatives became available.

However, the finding by Maier *et al.* [67] that Network News Transport Protocol (NNTP) traffic is responsible for up to 5 % of residential network traffic inspires us

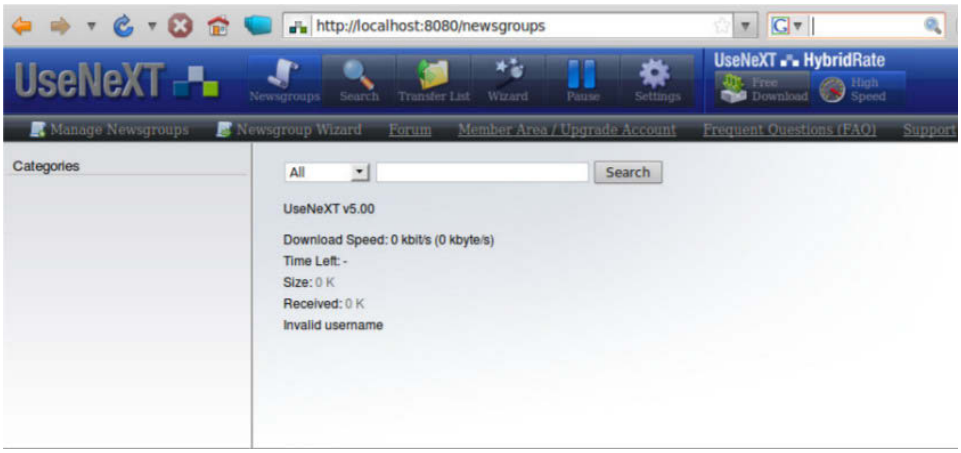


Figure 3.1: Screen-shot of an NNTP client of a fee-based offer

to revisit today's Usenet usage. Given this revival it is important to understand its causes and examine its characteristics. Therefore, we developed an analyzer for the Bro [87] Network Intrusion Detection System (NIDS). Our analyzer takes advantage of Bro's capabilities and utilizes dynamic protocol detection [24] and a specialized protocol semantic parser.

In this section, we present observations from the passive packet-level monitoring of more than 20,000 residential DSL lines from a major European ISP. This unique vantage point coupled with our NNTP protocol analyzer provides a broad view of its usage. We used Bro's online analysis capabilities to collect anonymized NNTP summaries which lasted for more than two weeks. In addition, we applied our analyzer to the traces also studied by Maier *et al.* [67].

Our initial expectation was that NNTP servers do not provide binary data. However, we find that most of the traffic ($> 99\,\%$) is binary content. We find that this is enabled by NNTP servers that require their users to pay a monthly fee and to authenticate themselves before using the server. Examples for such fee-based server providers are GigaNews or UseNext. Examining content-types of this binary traffic we find that archive formats as well as multimedia (AVI, MP3, and JPG) are dominant and are predominantly transferred with yEnc as binary-to-text encoding.

Furthermore, we note that such fee-based NNTP offers often provide customized NNTP clients (see for example Figure 3.1). Some of these clients take advantage of opening multiple simultaneous TCP connections and use globally unique article IDs instead of per news-group indexing. Moreover, the per connection throughput for NNTP is significantly larger for NNTP clients than for traditional P2P clients. Therefore, we conclude that NNTP's revival is due to the possibility of using NNTP as a high performance client/server-based alternative to P2P file-sharing, even though users have to pay a monthly fee.

3.1.1 Network News Transport Protocol (NNTP)

While NNTP is a well-known and well-established protocol its technical details might have faded from the readers memory. Therefore, we briefly review the basics of the protocol underlying Usenet—NNTP—in this subsection. Usenet is one of the oldest parts of the Internet. In fact, it predates the world wide web by more than 10 years. It was designed and is still being used as a system to exchange messages (called articles) between groups of users with similar interests—a functionality that today is also provided by, *e. g.*, web forums.

Usenet is structured as a (pseudo) hierarchy of groups, *e. g.*, `comp.os.minix` or `comp.os.linux.networking`. Anyone with access to a news server can subscribe to any of the news groups and then read or post messages within the group. The news servers are usually hosted by ISPs or at universities[1]. They are connected to form an overlay network, which is used to replicate articles between servers so that everyone has access to all articles not just those posted to the local news server. However, there is one limitation: It is the decision of the administrator of a news server if a particular news group is hosted on his server, *e. g.*, nowadays most news servers typically do not host the majority of the binary groups in order to avoid the risk of excessive bandwidth usage.

While NNTP is used for both client-server as well as server-server communication, in the remainder of this section, we focus on client-server communication since our focus is on NNTP usage by residential users. The Network News Transfer Protocol [27] and its message formats [5, 77] (until Nov 2009: RFC 1036 [42]) are very similar to SMTP. In fact, many of the article header fields are identical to the email message header fields, *e. g.*, `From` and `Subject`. In addition, there are NNTP specific headers, for example the `Message-ID` header, which contains a unique identifier for an article valid on all news servers.

The IANA assigned ports 119/tcp for NNTP and 563/tcp for SSL-encapsulated NNTP (NNTPS) are the default ports for communication between an NNTP client and server. The dialog starts when the client contacts the server. The server answers with a greeting message. Then the client can issue its commands. The server replies to each client command with a three digit status code, a status message, and an optional data block in "multi-line" encoding which contains, *e. g.*, the requested article. Likewise, a client can send a "multi-line" data block to the server, *e. g.*, to post an article. At the beginning of the connection a news server may require authorization before the client can issue any commands. Overall, we identify 33 different NNTP commands that are either specified in one of the RFCs or offered by popular servers such as INN. Examples include selecting a group (`GROUP`), listing the articles within a group (`LISTGROUP` or (`X`)`OVER`), and fetching (`ARTICLE`, `BODY` and `HEAD`) or sending (`POST`) articles.

[1] Users whose ISP does not offer access to a news server can subscribe for a small fee to independent servers, *e. g.*, `http://news.individual.net/`

The article itself is composed of a newline-separated list of headers and an article body separated by a double newline[2]. Within NNTP everything is encoded as a "multi-line" data-block, including article bodies, or command results. This "multi-line" data-block format of NNTP imposes several restrictions on the content. For example it cannot contain NUL characters, it has a limited line length, and the period character (full stop) at the beginning of a line has to be replaced by an alternative interpretation. Therefore, it is impossible to transfer binary data without encoding. Popular encodings are yEnc, UUencode, MIME, base64, BinHex, Quoted-Printable and XXEncode.

Note that MIME has just recently been standardized (RFC 5536 [77] and 5537 [5]) as transfer encoding for the Usenet while yEnc and UUencode are well established. UUencode has originated on System V and has traditionally been used to transfer email messages and Usenet articles before the corresponding TCP based protocols were in use. With Usenet in mind yEncode has been designed to minimize the overhead imposed by alternative encoding schemes. The idea is to only encode characters if it is absolutely required to adhere to the message format standard. The name yEncode is actually a wordplay: "Why encode?". While it is in principle possible that other binary encodings may be in use and may not be identified, their traffic volume in our data sets is minimal.

3.1.2 Methodology

For detecting NNTP connections we use a DPD signature that checks if the server side of the connection contains an NNTP welcome banner and if the client side uses one of the 33 NNTP commands[3]. After a connection is identified as candidate for NNTP an analyzer written in BinPAC is attached to the connection and generates an event for every detected request command, reply message, and client and server side data sections (multi-line).

Note, that because NNTP is stateful, as opposed to, *e. g.*, HTTP, our analyzer also needs to keep state per instance. Reasons for this necessity include client-side data transfers (*e. g.*, via `POST`, `IHAVE`, or `XREPLIC` commands) or replies with or without multi-line data blocks depending on the request. In Figure 3.2, we show that if the client wants to post a multi-line article it first sends a single-line query, waits for a positive response from the server, and only then sends a multi-line data block. In Figure 3.3, we illustrate different reponse-types for status code 211 depending on the command issued by the client. Therefore, the analyzer needs to remember the last command for every connection, producing state that needs to be managed.

[2] A newline in this context is the two character string `\r\n`.

[3] To minimize the number of false positives we currently filter traffic on TCP port 25, mainly used by SMTP.

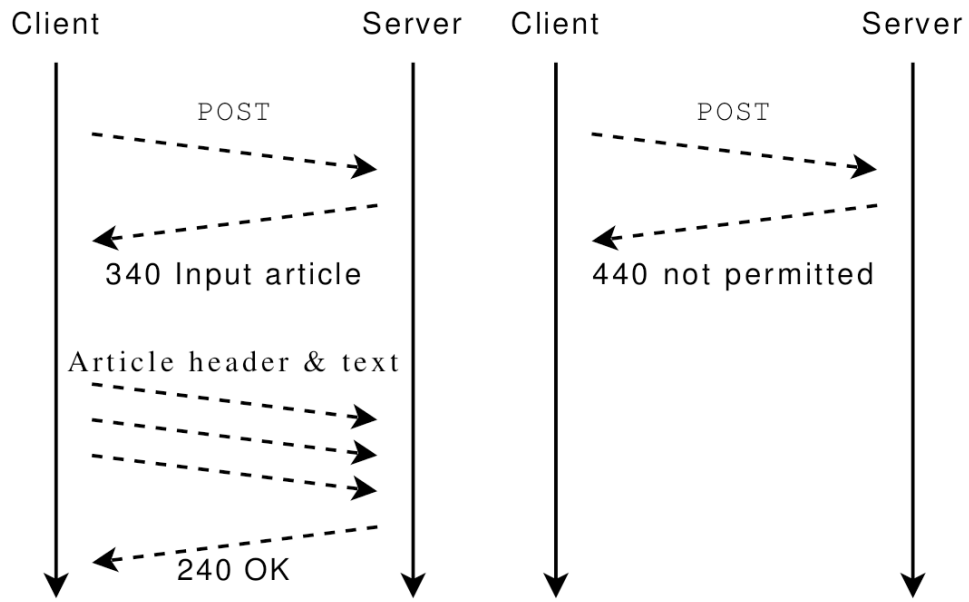


Figure 3.2: Example scenarios of a `POST` transaction: Multi-line data is transmitted only when `POST` request is granted by the server (left).

NNTP uses different encoding types to transfer binary content. To determine the encoding type we use the BinPAC analyzer itself instead of the scripting layer to lower CPU load. Thus, we add an event if a text-encoded binary payload is located to inform the scripting layer about encoding type and method. We distinguish between *(i)* yEncode (yEnc) [39], *(ii)* UUencode [41], *(iii)* MIME [32], and *(iv)* non-binary content.

Articles with MIME attachments are identified by looking for and inspecting the `Content-Type` header. Content types and filenames are then extracted by parsing the MIME formatted body. UUencoded article bodies are detected by looking for an UUencode header in the article body. This header starts with the string '`begin␣`' followed by a three digit number denoting the UNIX permissions in octal notation, and a file name. The binary block ends with the string '`end`' on a single line. yEnc works similar to UUencode: the binary block starts with a string '`=ybegin␣`' followed up by parameters describing the length of each block, the size of the resulting file, and the file name on the same line. Each block ends with the string '`=yend␣`' followed by additional parameters.

Our methodology for determining the content-type of a binary encoded file is straightforward and relies either on the filename extension or on the content-type of the binary-to-text encoding. In future work we plan to extend the analyzer to decode the binary file and use libmagic to determine the file-type.

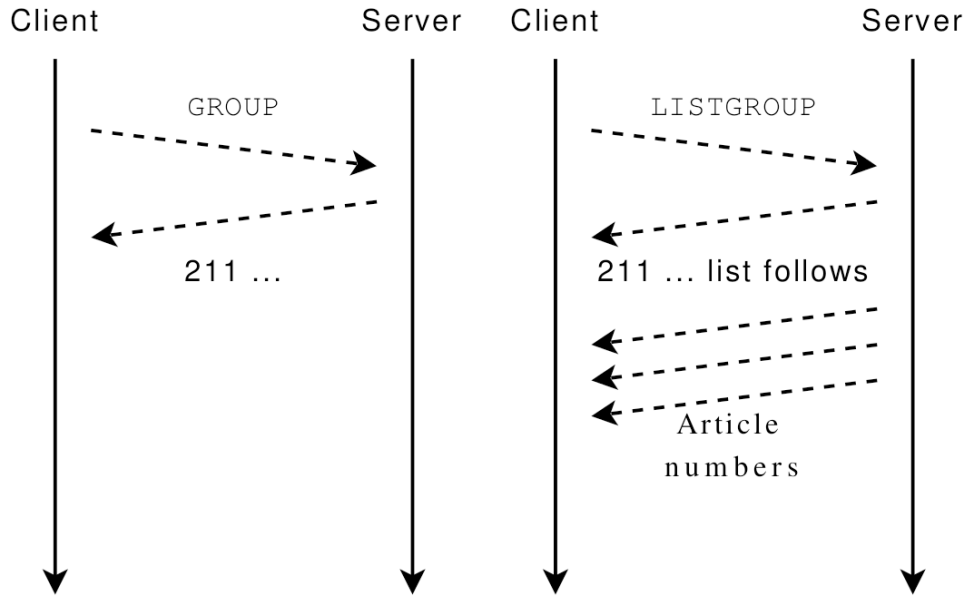


Figure 3.3: Example scenarios of a 211 response code: Multi-line data is transmitted only when a `LISTGROUP` request was sent (right).

Name	Start date	Duration	Size	NNTP
NNTP-15d	Wed 05 Aug '09 3am	$15\frac{1}{4}$ d	n/a	n/a
SEP08	Thu 18 Sep '08 4am	24 h	>4 TB	5 %
APR09	Wed 01 Apr '09 2am	24 h	>4 TB	2 %
AUG09	Fri 21 Aug '09 2am	48 h	>11 TB	2 %

Table 3.1: Overview of anonymized packet traces and summaries.

The analyzer produces an anonymized one line summary for every single or multi-line request or response including a time-stamp, a connection identifier, anonymized IP addresses of client and server, size of the action, and information about the observed action. This output is post-processed to generate one-line summaries of corresponding NNTP request and response pairs, called NNTP transactions in the remainder of the section. In these one-line summaries we also report the news group in which the request was issued.

3.1.3 Datasets

We base our study on multiple sets of anonymized packet-level observations of residential DSL connections collected at our ISP vantage point (see Section 2.1.2).

Table 3.1 summarizes characteristics of the data sets, including their start, duration, size, and fraction of NNTP. We used Bro's online analysis capabilities to collect an anonymized trace summary which covers more than two weeks of NNTP traffic (NNTP-15d). Moreover, we use several anonymized one/two day packet traces collected over a period of 11 months (SEP08, APR09, AUG09). These are the same traces as studied by Maier *et al.* [67]. While we typically did not experience any packet loss, there are several multi-second periods (less than 5 minutes overall per packet trace) with no packets due to OS/file-system interactions.

All traces contain a noticeable fraction of NNTP traffic. Similar results have been observed at different times and at other locations of the same ISP. For the online trace summary we only capture NNTP traffic and therefore do not know the exact fraction. However, comparing the volume to other traces we presume that it corresponds to a similar fraction.

We omit analyzing NNTPS for two reasons: *(i)* due to the encrypted nature of NNTPS it is impossible to inspect the content and *(ii)* the amount of traffic on the respective port is small compared to the observed NNTP traffic. Indeed, we do not observe more than 0.3 % of total traffic on port 563, the standard SSL port of NNTP.

3.1.4 Results

To understand the revival of NNTP we have to study how NNTP is used today. Accordingly, in this subsection, we start by studying the overall characteristics of our traces. Then we examine the volume distribution of NNTP transactions, the popularity of commands, encoding methods, content types, news groups, and NNTP servers.

NNTP Characteristics

In our traces of residential Internet traffic 2 % (5 % for SEP08) is identified as NNTP. We observe roughly 150 users in each of the packet level traces (SEP08, APR09, AUG09) and roughly 300 users in NNTP-15d. We also observe that the typical NNTP client uses multiple TCP connections in parallel (1st quartile: 2–3, median: 4–6, 3rd quartile: 7–8) and that up to 12 different servers are contacted by a single client—maybe to balance the load across different servers. Note that less than 1 % of the users (150 out of 20000) causes more than 2 % of the traffic with NNTP only. In addition these users are also among the top users in terms of per-line traffic contribution. This is confirmed by Maier *et al.* [67]. They also report that P2P use and NNTP use is unlikely to be observed at the same line.

Furthermore, we find that roughly 99 % (only 96 % in AUG09) of the transmitted volume is due to binary transfers. Yet, we notice that the fraction of binary queries

is rising. The fraction is 55 % for SEP08 whereas it is roughly 90 % for the other traces (APR09, AUG09, NNTP-15d). Moreover, the average duration of observed NNTP connections[4] is 24 minutes for SEP08 while the average connection duration of all 2009 traces is 6 or 9 minutes. In order to understand the origin of these differences we investigate SEP08 in more detail. We find that there are three DSL lines in this trace that exhibit strange behaviors:

- One line is responsible for roughly 30 % of the NNTP volume in SEP08 (mainly via `BODY` commands). In all other traces the top contributing line is responsible for at most 10 % of the volume.
- Another line is responsible for more than 90 % of the `GROUP` and `STAT` commands in SEP08.
- A third line interacts with a server such that after normal NNTP activities the server sends tons of single-line responses with an unknown status code (not specified in the RFCs).

These three lines are responsible for roughly 48 % of the NNTP transactions and thus we also report results for SEP08 without these 3 lines, labeled as "SEP08-w3l". For example, we find that 79 % instead of 55 % are binary downloads for SEP08-w3l. Since, SEP08 also contains some regular news reading activity the results for SEP08-w3l are closer to those of the other traces but still differ. For privacy reasons, we do not investigate these 3 lines in detail.

Distribution of Transaction Volumes

Recall that an NNTP transaction consists of an NNTP request and its corresponding NNTP response. Figure 3.4 depicts the cumulative distribution of the transaction volumes of each of the traces. This plot highlights several specifics of NNTP traffic. For instance, a single-lined message must not exceed 512 bytes. This results in the step around 100 bytes. The most frequent transaction sizes are 252 kB and the median transaction size 387 kB (see helper line in plot). They seem to correspond to the sizes of typical binary data blocks, which are parts of multi-part archives or multimedia files. We assume that these sizes are due to either the software used to partition the files or server restrictions on article length. Although we tried to confirm any of the explanations, we did not find evidence.

Popularity of NNTP Commands

Although we identified more than 30 possible NNTP commands and include all of them in our signature for detecting NNTP traffic, only 16 of them are observed in any

[4] For this statistic we only consider complete NNTP connections, *i. e.*, those that contain the command `QUIT`.

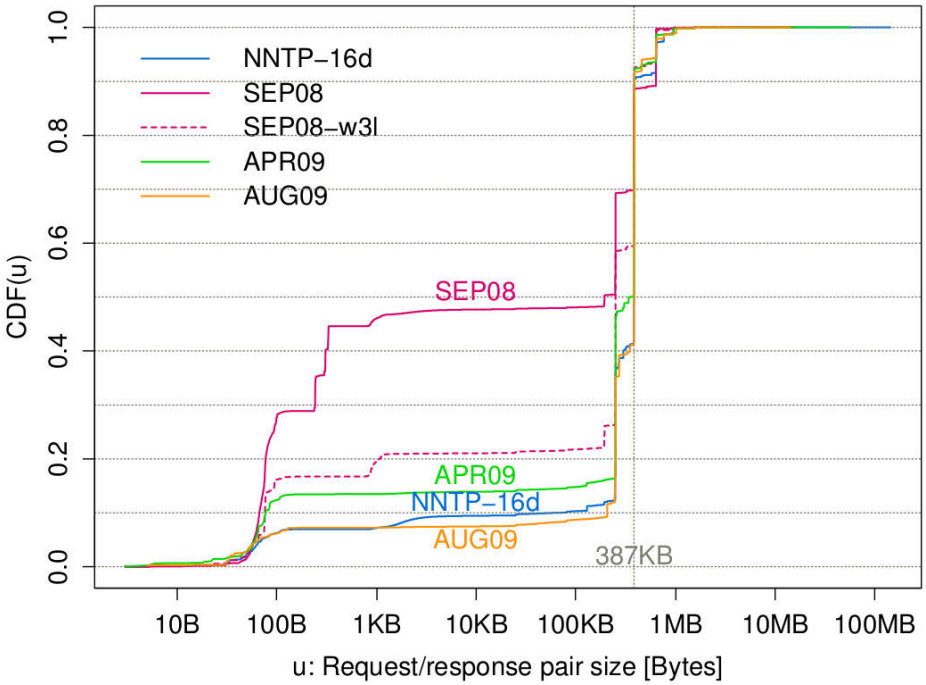


Figure 3.4: Cumulative distribution function of transaction volumes

Trace	reply only	ARTICLE	BODY	GROUP	STAT	HEAD	AUTHINFO	QUIT	MODE	XOVER
SEP08	15,9 %	43.6 %	15.7 %	14.1 %	6.4 %	2.1 %	1.0 %	0.1 %	0.1 %	<0.1 %
SEP08-w3l	-	82.6 %	8.1 %	0.9 %	-	4.1 %	1.8 %	0.2 %	<0.1 %	<0.1 %
APR09	-	83.2 %	10.3 %	1.1 %	-	<0.1 %	2.3 %	0.7 %	0.2 %	<0.1 %
AUG09	-	66.5 %	27.1 %	0.4 %	-	<0.1 %	2.0 %	0.2 %	1.2 %	<0.1 %
NNTP-15d	-	76.5 %	18.1 %	0.6 %	-	0.3 %	1.8 %	0.2 %	0.6 %	<0.1 %

Table 3.2: Frequency of commands

of our traces. The overall frequencies of the top commands are listed in Table 3.2. As expected, the most frequently issued command is `ARTICLE`. Interestingly, the `BODY` request is the next most popular and it is significantly more popular than `HEAD`. This indicates that the users are not downloading the complete meta information about the bodies that they download. Note the effect of removing the 3 lines from SEP08.

In terms of volume only the commands `ARTICLE` and `BODY` are relevant and together contribute more than 99 %. NNTP offers two different identifiers for articles: *(i)* an article number relative to the newsgroup or *(ii)* a globally unique article identifier. The first kind of identifier requires the user to first select the corresponding newsgroup, *e. g.*, via the `GROUP` command. The latter identifier can be used without entering a newsgroup. We find that the globally unique article identifiers dominate and may explain the small number of `GROUP` commands.

Trace	non-binary	yEnc	UUEnc	MIME
SEP08	47.7 %	52.2 %	0.1 %	<0.1 %
SEP08-w3l	21.1 %	78.6 %	0.3 %	<0.1 %
APR09	14.7 %	84.8 %	0.5 %	<0.1 %
AUG09	9.8 %	89.9 %	0.3 %	<0.1 %
NNTP-15d	9.8 %	89.9 %	0.2 %	<0.1 %

Table 3.3: Frequency of Binary-to-text encoding methods

Trace	archive /rar	archive /par2	video /avi	audio /mp3	image /jpg	other types
SEP08	84.30 %	2.57 %	5.19 %	1.46 %	0.40 %	6.08 %
APR09	84.13 %	1.73 %	7.08 %	2.36 %	2.07 %	2.63 %
AUG09	83.81 %	2.01 %	3.16 %	2.90 %	1.26 %	6.86 %
NNTP-15d	81.18 %	2.24 %	6.73 %	3.11 %	0.66 %	6.08 %

Table 3.4: Frequency of Binary File Types

Popularity of Binary-to-text Encoding Methods

Given that most requests and almost all of the bytes are due to binary content we next explore which binary-to-text encodings are used. In all traces yEnc dominates with more than 99 % of the bytes being transferred (Table not shown). The frequency of requests for each encoding method is shown in Table 3.3. Among the binary encodings yEnc is still dominant. We assume that yEnc's prevalence results from its significantly smaller encoding overhead.

Content-types of Binary Encoded Files

Given that almost all NNTP traffic is binary encoded we now explore which files are contained within these binary transfers, see Table 3.4. The most frequently transferred file format is archive/rar. RAR is a file compression software which is able to archive files and separate them into multiple smaller files. The file extension of separated files is three characters either 'rar', a 3-digit number or 'r' followed by a 2-digit number. Files for which the extension matches this condition have been assumed to be RAR files. We also observe parchive parity or index files (PAR2). These can be used to recover broken or unavailable data [86]. In addition, we see a significant fraction of multi-media formats. We note that in P2P file-sharing systems multimedia is more popular than archives according to ipoque [106].

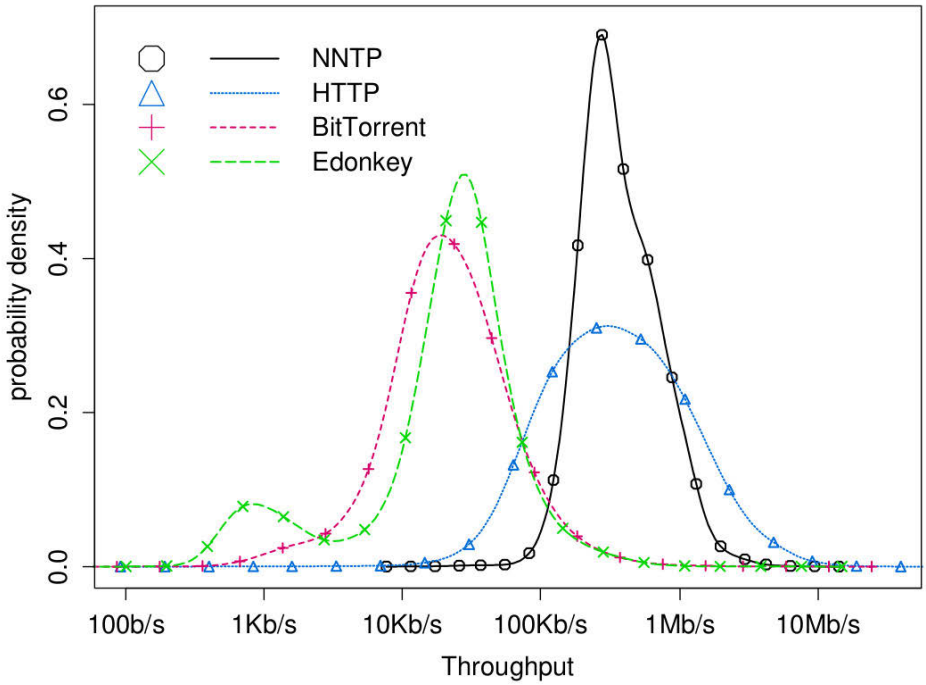


Figure 3.5: Probability density function of achieved throughput of flows > 50 kBytes for different protocols in APR09.

Popularity of News Groups

Given our results so far we expect to see a large popularity of binary news groups. This is indeed the case. Most of the observed Usenet activities are in sub groups of `alt.binaries`. However, since most articles are requested via their globally unique ID we cannot identify which group they belong to. Moreover, for globally unique IDs, it may be wrong to presume that an article belongs to the most recently selected group. Therefore, we do not present numbers in this subsection.

Popularity of News Servers

Since binary news groups are usually not available on public NNTP servers we now examine which servers are contacted and apparently provide the desired binary content. To identify the server operator we use two approaches: *(i)* we parsed the welcome message line of the servers, *(ii)* we cooperated with the ISP to perform reverse lookups of the servers IP addresses. Both methods yield the same results: Most of the Usenet servers offering binary groups are commercial servers that require a monthly fee, *i. e.*, GigaNews, UseNeXT, Alphaload, and Firstload. Further inves-

tigation reveals that there is even a reselling business for commercial NNTP server access.

Our investigation reveals that more than 93 % of the queries are directed to and more than 99 % of the volume of all NNTP traffic is exchanged with commercial Usenet servers. Browsing the web pages of these commercial Usenet server operators and taking advantage of some free trial offers we find that such offers come along with a special NNTP client. Clients for Unix-like OSes, *e. g.*, MacOS or Linux, are often realized as background processes that are controlled via a Web browser by using machine local communication. These clients also provide a search engine for NNTP articles. This is an additional feature that NNTP does not provide by default. It is possible that such search queries are not using the NNTP protocol.

Throughput of NNTP

Next, we pose the question why people are willing to pay a monthly fee for something that is likely to also be freely available, *e. g.*, via P2P file-sharing services. Therefore, we investigate the achieved throughput of NNTP flows and compare these to BitTorrent, eDonkey, and HTTP flows. Figure 3.5 shows a probability density function of the logarithm[5] of the achieved throughput for APR09. We can clearly see that NNTP outperforms the P2P-based systems. Given that both P2P-based systems and NNTP use multiple connections in parallel we assume that the overall time to download the same amount of data is an order of magnitude smaller for NNTP. This is basically due to the fact that NNTP servers usually have better Internet connectivity as compared to P2P users. For HTTP the achieved throughput is in the same order as for NNTP, although HTTP has a higher variability than NNTP. Thus, better performance may be the reason to either use fee-based NNTP servers or fee-based One-Click Hosters (such as Rapidshare or MegaUpload) for file sharing. One advantage for NNTP is the ability to search NNTP newsgroups for specific content. Moreover, NNTP features a protocol/operation inherent content replication and content distribution schemes.

Overall, the results show that NNTP is a bandwidth intensive application protocol due to the misuse of Usenet as a high performance file-sharing platform. In the next section, we choose another protocol that has different characteristics, *i. e.*, a rather delay intensive application protocol, in order to see a clear contrast between two protocols.

[5] Coupled with a logarithmic scale on the x-axis, plotting the density of the logarithm of the data facilitates direct comparisons between different parts of the graphs based on the area under the curve.

3.2 Traffic Analysis of a Massively Multi-player Game

Findings in the previous section prove that NNTP's role in today's Internet is significant due to its share in total traffic (up to 5 %). However, the portion of users participating in Usenet is rather small (less than 0.01 %). In this section, we examine a contrasting protocol of NNTP (in terms of the number of users and the traffic share) within the same measurement environment.

Given the fact that the number of Massively Multi-player Online Role Playing Game (MMORPG) subscribers rapidly increases every year and that it reached 22 million in 2011 [144], for ISPs, it is considered as an important task to understand characteristics of Internet traffic generated by MMORPGs and the collective behavior of their users. According to several reports (see for instance [144, 145]), World of Warcraft (abbreviated as WoW) alone accounts for more than 50 % of the total MMORPG subscriptions. Thereby, this particular online game has attracted much attention of academia and industry.

Therefore, as the second contribution of this chapter, we present an analysis of MMORPG traffic and users' gaming behavior focusing on World of Warcraft as a representative game. The contributions of this study are mainly three-fold. *(i)* we reveal the characteristics of World of Warcraft game traffic by thoroughly analyzing Internet traffic collected from a European tier-1 ISP (*i. e.*, the ISP vantage point described in Section 2.1.2). *(ii)* we present changes of traffic behavior by observing two data sets collected in different time periods. *(iii)* we compare the gaming behavior of users who play the game alone and users who play the game together with other users behind the same middle box. As far as we know, none of the previous studies [56, 73, 89, 111, 114] questioned on the theme (the user behavior of MMORPGs) from this perspective.

3.2.1 Overview of World of Warcraft

World of Warcraft was first introduced in the market by Blizzard Entertainment in 2004 and soon became one of the most subscribed (the number of its worldwide subscriptions reached 12 million in October 2010 [145]) online games. A user[6] within World of Warcraft is represented as a graphical form, which is often called the avatar, whose identity (name, gender etc.) is usually different from the user's real world identity. This type of game is commonly referred to as a Massively Multi-player Online Role Playing Game (MMORPG).

Unlike traditional computer games, in which the user plays as the only intellectual game entity and the other entities act based on the story line programmed by the game developers, MMORPGs provide their subscribers virtual worlds in which users meet and communicate with other users.

[6]The term "user", "subscriber", and "player" are interchangeably used in this study.

Indeed, the interaction between users is the most distinguishing feature of online games. For MMORPGs, more than a thousand users play the game at the same time and they build the real-world-like society in the game world. Moreover, users own private possessions and even trade them with other users in virtual world currency. These sorts of social features allow MMORPG providers to introduce a different business model from that of traditional games. While users pay for the purchase of traditional computer games, MMORPG users pay fee based on time they play.

Although World of Warcraft is designed to accommodate a massive number of players, it needs a certain level of load balancing. To this end, Blizzard Entertainment provides multiple copies of the virtual world for their users, which are called realms. Thus, avatars are only able to interact with the other avatars that are within the same realm.

3.2.2 Classification Method

In order to illustrate our classification method in detail, we first provide some level of technical information about the World of Warcraft protocol. The World of Warcraft protocol uses TCP as its transport protocol and a client typically opens two connections towards different servers. One connection is established between the client and the logon server in order to authenticate the subscriber and to update relevant server/client information such as the list of game servers and the status of the subscriber's avatar. The other connection is established between the client and the game server in which actual gaming information, *e. g.*, coordinates of the avatar and chat messages, is exchanged.

Our analyzer is implemented to detect the protocol in two steps. The initial step of the protocol identification is based on examining the protocol's unique byte pattern (signature) of the first packet of the connection. We use `^\x00...WoW` and `^..\xed\x01` as regular expression signatures of the logon connection and the game connection, respectively. However, due to the short signature length, relying only on this step yields a high false-positive rate. Thus, the next step verifies if the connection responder replies with the expected signatures. In this step, we use `^\x00` (logon connection) and `^..\xec\x01` (game connection) as signatures.

Even after the protocol classification, the nature of the proprietary software that the World of Warcraft programs have and partly encrypted protocol messages make deep inspection of World of Warcraft traffic extremely difficult. Thus, we use the unencrypted part of protocol messages for our analysis. As we will show in Section 3.2.4, more than 60 % of the World of Warcraft packets are coordination information which can be translated into human readable text.

	Overall Traffic			World of Warcraft				
Name	Mon. Year	Duration	Volume	packets	Volume	Version	Users	Avg. Playing
ISP08	Aug. 2008	12:00 – 12:00	> 4TB	0.91 %	0.48 %	2.4.3	3.0 % (599)	1.76 Hours
ISP10	Mar. 2010	02:00 – 02:00	> 4TB	0.83 %	0.72 %	3.x.y	1.4 % (280)	4.17 Hours

Table 3.5: Overview of anonymized packet traces.

3.2.3 Data Sets

We use two anonymized 24-hour packet-level traffic data sets (ISP08 and ISP10) for our study. These two traces are collected from our ISP vantage point (see Section 2.1.2) in 2008 and in 2010. The relevant information of our traffic data sets and the number of identified World of Warcraft subscribers are summarized in Table 3.5.

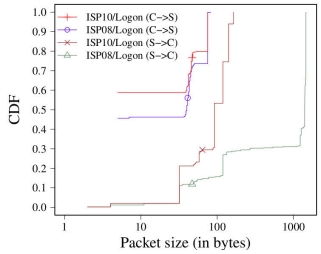
In light of the fact that World of Warcraft traffic is pure control traffic generated by users (*e. g.*, by mouse button clicking and/or by key pressing), as opposed to the media content delivery, we believe that 0.48 % and 0.72 % (see Table 3.5) of traffic contributions are worthwhile to study. Note that traffic generated by software updates is not included.

3.2.4 Traffic Characteristics

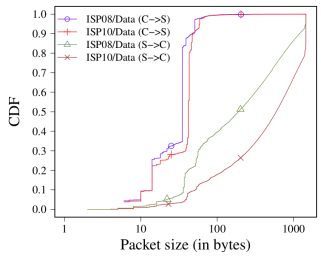
In this subsection, we first illustrate the general characteristics of World of Warcraft traffic. Then, we present the result of our analysis, which is the majority of the World of Warcraft traffic is movement messages in which coordinates of avatars and nearby objects (*e. g.*, non-player characters or game items) are delivered.

General Characteristics

Figures 3.6, 3.7, and 3.8 respectively depict the distributions of packet size (payload only), packet sending rate, and throughput. We report results of logon connections separately from those of data connections since those two types of connections show clearly different behavior in terms of packet count, traffic volume, and duration of connections. Interestingly, we observe that the line shape of server-to-client packet streams of ISP10 is significantly different from that of ISP08 in all logon plots, while the change of client-to-server packet streams is not remarkable. Furthermore, the lines representing data packet streams of ISP10 ((b) of all three figures) have drastically shifted compared to those of ISP08. We conclude that the protocol has been modified during the two years in such ways that a server delivers logon information in a more compressed or more distributed manner and delivers gaming information more frequently.

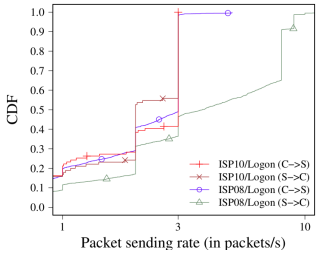


(a) Logon connections

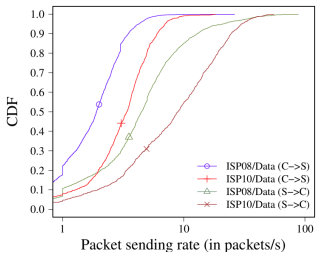


(b) Data connections

Figure 3.6: Packet size

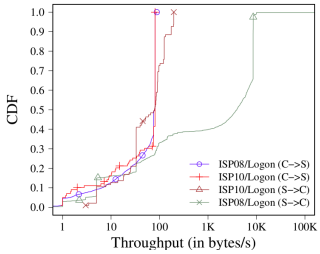


(a) Logon connections

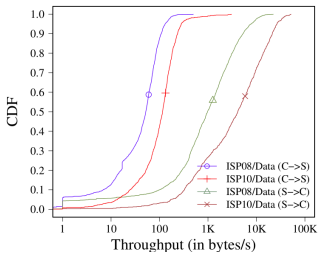


(b) Data connections

Figure 3.7: Packet rate



(a) Logon connections



(b) Data connections

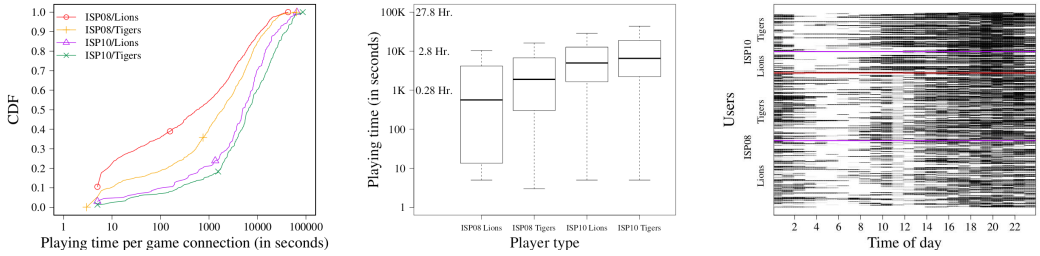
Figure 3.8: Throughput

Comparing the throughput statistics of the two traces shown in Figure 3.8, the later version of World of Warcraft protocol requires more bandwidth than the earlier one. Yet, its bandwidth utilization is relatively low considering the network bandwidth provided by today's ISPs. This means, contrary to popular belief among users, upgrading the link capacity may not be the solution for a better gaming experience.

Movement Messages

Regarding the peak of client-to-server packet streams observed in Figure 3.6 (b), 43 bytes (ISP08) and 51 bytes (ISP10) are typical sizes identified from packets that deliver avatar's coordinates within the virtual world to the server. The reason why the size of the movement message differs in the two traces is that the object's ID is embedded in a different way in different versions of the protocol. This phenomenon leads us to the conclusion that more than 60 % of the total packets delivered from clients to servers is the avatar's coordinate information.

By deeply inspecting server-to-client messages, we find that the majority of messages sent from servers to clients are movement messages for updating the client on coordinates of its nearby players and objects. However, we do not observe the peak that appears in client-to-server packet streams since a server sends coordinates of various numbers of objects within one packet.



(a) Connection duration (b) User playing time (c) Playing time of day

Figure 3.9: Playing time distributions

3.2.5 Tigers vs Lions

Group size (in players)		ISP08			ISP10		
		# of IPs.	# of users	volume	# of IPs	# of users	volume
Tigers	1	487 (75 %)	487 (54 %)	53 %	257 (82 %)	257 (68 %)	65 %
Lions	2	118 (18 %)	236 (26 %)	28 %	46 (15 %)	92 (24 %)	30 %
	3	26 (4 %)	78 (9 %)	10 %	8 (3 %)	24 (6 %)	> 4 %
	4	11 (2 %)	44 (5 %)	< 6 %	1 (<1 %)	4 (1 %)	< 1 %
	> 4 *	7 (<2 %)	55 (6 %)	< 4 %	0 (0 %)	0 (0 %)	0 %
	total	649 (100 %)	900 (100 %)	100 %	312 (100 %)	377 (100 %)	100 %
* Maximum number of users behind a single IP address is 14							

Table 3.6: Categorization of users based on the number of locally grouped co-players

MMORPG subscriber's gaming behavior is an often addressed research topic in online game traffic measurement. Such studies are especially important for ISPs in order to understand a user's increasing demand for a good gaming experience. Assuming MMORPG users form a homogeneous group, previous studies [15, 56, 73, 110, 111] focus either on characterizing the overall gaming behavior of users or on comparing the user behavior in different virtual worlds. However, in order to gain a more complete insight into the gaming behavior of MMORPG subscribers, we take a different approach to those of earlier work.

Playing Behavior

We first classify World of Warcraft players into two groups. The first group consists of users who are the only players behind an IP address. The other group consists of users whose IP addresses are shared with other World of Warcraft users. We refer to the former as Tigers and to the latter as Lions according to their hunting behaviors

(tigers hunt individually, while lions hunt in a pride). One must note that players may be included in both groups multiple times due to changes in their playing locations or the reallocation of IP addresses, thus the number of users reported in Table 3.6 is greater than the one in Table 3.5. It is crucial to mention that we do not focus on characteristics of individual users, but intend to study general differences between locally grouped players and solitary players in terms of playing time, traffic volume, and distance their avatars move within the virtual world. We summarize relevant statistics in Table 3.6.

With few exceptions in ISP08, we find that the number of users per group of Lions is less than 4 which is a common maximum number of physical network ports on home networking devices. The table explains that the fraction of Tigers increases in ISP10.

Figure 3.9 illustrates the subscriber's playing time from different perspectives. While Figure 3.9 (a) depicts the distribution of connection duration, the result shown in Figure 3.9 (b) depicts the distribution of user's playing time. We make two observations from the figures. *(i)* while the number of identified game users decreased during the two years, the playing duration increased. Taking a deeper look, we find that the fraction of users who play the game less than 0.28 hours a day decreased from 40 % to 20 %, while the fraction of users who play the game longer than 2.8 hours in a day increased from 20 % to 40 %. *(ii)* considering that the y-axis of Figure 3.9 (b) is in a logarithmic scale, solitary users (Tigers) are playing notably longer than users playing together behind the same middle box (Lions).

In Figure 3.9 (c), we illustrate the time of day that users play the game. Note that the y-axis of this figure represents World of Warcraft users and x-axis time of day. Each time slot is filled with various depth of gray color depending on the minutes played in the time slot. Thus, a time slot is fully used when it is black and it is not used when it is white. The x-axis is wrapped around the beginning time of the measurements. It is crucial to illustrate the results in this manner in order to make the results comparable. Indeed, the two traces do not begin at the same time of day and also do not begin at mid night (see Table 3.5). The sudden bright cells observed between 11 a.m. and 1 p.m. in ISP10 is due to the fact that the connections established before the beginning of the measurement cannot be recognized by the analyzer. The figure suggests that there is only a negligible difference between Tigers and Lions when considering the time of day they play. Unsurprisingly, popular time slots of a day are identified between 7 p.m. and 11 p.m.

In-game Behavior

Next, we analyze the distance that avatars move in the virtual world during game play. Figure 3.10 illustrates the distance distribution of the avatar's movement. As there is no way to map the distance in the virtual world to the real world's metric, we

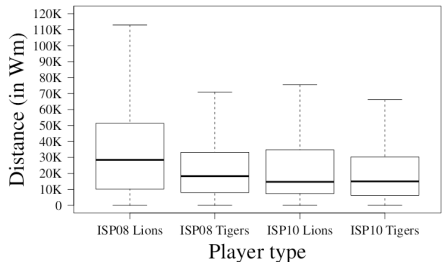


Figure 3.10: Movement of avatars

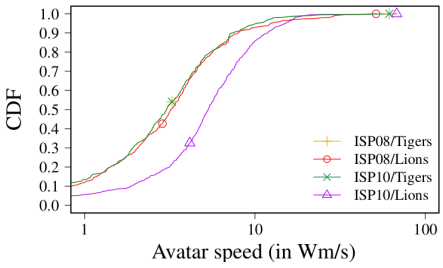


Figure 3.11: Speeds of avatars

invent an imaginary metric `Wm` for measuring the distance in World of Warcraft. In order to provide readers with the intuitional hint of this virtual metric, we illustrate the distribution of the speed at which avatars move in the virtual world in Figure 3.11. In this analysis, we find that about 0.8 % of the identified avatars exhibit unrealistic movement speeds (two to three orders of magnitude larger than the average speed). We assume that these are mainly due to the usage of long distance transportation systems such as the teleport. We, thus, ignore such extremely fast movements from the result. From this evaluation, it is calculated that average speeds of avatars are 4.25 `Wm/s` (ISP08) and 6.39 `Wm/s` (ISP10). Figure 3.10 illustrates that Lions move more than Tigers in game. This is likely due to the fact that Tigers spend more time communicating with other avatars in the virtual world, whereas Lions focus more on hunting in the battle field. This is a reasonable inference because, for Lions, a communication with other users does not interfere in the movement in the virtual world as their in-game friends are within the talking distance.

So far, we have explored characteristics of World of Warcraft traffic and the collective behavior of its users. World of Warcraft neither requires high bandwidth (the highest bandwidth observed in the measurement is about 700 Kbps) nor generates a large traffic volume (up to 0.7 % of total traffic), while the number of users is quite significant (up to 3 % of the number of users behind the vantage point). When comparing this to the results of the NNTP traffic analysis (Section 3.1), we see the clear contrast between these two influential protocols.

3.3 Assessment of IPv6 Adoption from an IXP's Viewpoint

The previous sections presented analyses of some important (in one way or another) application protocols within the scope of IPv4 based on traffic data collected from a stub network. In this section, we move to the network layer to determine the current situation of IPv6 adoption and the relationship between IPv4 and IPv6. For this, we change our vantage point from a stub network (ISP) to the highly aggregated network (IXP).

Although designing a new Internet protocol was not the most pressing matter within the Internet community in the beginning of the 1990s, the unforeseen fast growth in Internet usage had started to cause worries about the exhaustion of the Internet address space.

Internet Protocol Version 6 (IPv6) was developed by the Internet Engineering Task Force (IETF) in 1994 with a view to succeeding the current version of Internet Protocol (IPv4). Besides the expansion of the address space, developers of IPv6 took several demanding features such as security (based on compliance with IPSec), Quality of Service (QoS, via the prioritization scheme and the non-fragmentation principle), and extensibility (using the chain header) into design consideration, while maintaining the simplicity of its predecessor (IPv4). Even with such promising functionalities network operators and software developers were not motivated enough to adopt the new version of the Internet protocol because the current version of the Internet protocol just works fine and it was still too early to feel the scarcity of the address space in their bones. Furthermore, it was commonly acknowledged that inequalities in the benefit of and the demand for the new Internet protocol among network operators made it difficult to adopt IPv6 all together.

To this end, the Internet Society [141] organized a 24-hour global IPv6 test flight (*World IPv6 Day* [120]) on 8 June, 2011 and more than a thousand globally influential service providers have participated in the event. As no severe problems have been reported from participants, the community has decided to move a step forward, which is the *World IPv6 Launch Day* [143] event held on 6 June, 2012. The *World IPv6 Launch Day* event was expected to be an important turning point in the Internet history, because participants agreed to keep their IPv6 connectivity permanently enabled after the event.

In this section, we study the changes to the IPv6 traffic from the viewpoint of a large European IXP (*i. e.*, the IXP vantage point described in Section 2.1.2). We analyze 14 months worth of traffic traces which include the two world IPv6 events.

3.3.1 Methodology

In order to conduct our measurement, we developed an IPv6 traffic analysis tool to which text-formatted information extracted from sFlow traces is fed. Our analysis tool is designed to identify IPv6 packets and to extract relevant information from the identified IPv6 packets.

The tool identifies the transition technology in the current measurement point (IXP) by observing the packet's IP version field, protocol number, and port numbers. For this, we use the protocol number 41 as a clear indicator of 6in4 [81] technology and the protocol number 17 combined with the UDP port number 3544 as a clear evidence of teredo [45] technology. In the same manner, ayiya [70] technology can be identified by using the protocol number 17 and the UDP port number 5072. However, we do

Name	Period	Global IPv6 Event
IXP11-11d	Jun. 05, 2011 – Jun. 15, 2011	*World IPv6 Day*
IXP1x-13d	Dec. 25, 2011 – Jan. 07, 2012	—
IXP12-11d	Jun. 01, 2012 – Jun. 11, 2012	*World IPv6 Launch Day*
IXP12-olympics	Aug. 12, 2012 – Aug. 30, 2012	—

Table 3.7: Data sets.

not consider the ayiya transition technology in our study since we observe only a negligible fraction (<0.1 %) of IPv6 traffic over ayiya. Indeed, teredo and 6in4 are the only transition technologies carrying a considerable amount of IPv6 traffic.

Our measurement study is performed based on four sets of traffic traces (see Table 3.7) collected within our IXP vantage point during a time span of 14 months including *World IPv6 Day* and *World IPv6 Launch Day* (54 days of traffic in total). Note that one of our data sets (IXP11-11d) overlaps the one used in Sarrar *et al.* [102].

3.3.2 Evaluation

In this subsection, we evaluate the change in the level of IPv6 deployment with emphasis on the impact of the two past global IPv6 events (*World IPv6 Day* and *World IPv6 Launch Day*). In general, we observe that IPv6 accounts for 0.5 % of the total Internet traffic in the peak period (IXP12-olympics) of our measurement.

Quantitative Analysis of IPv6 Traffic

In order to see how the level of IPv6 adoption has been evolving, we download monthly snapshots of the IPv6 routing table from the RouteViews Project [136] and illustrate them in Figure 3.12 together with the observed IPv6 prefixes within our data sets. The figure shows a sharp rise in the number of announced IPv6 network prefixes since the *World IPv6 Day*. More precisely, the number of IPv6 network prefixes announced between the *World IPv6 Day* period and the *World IPv6 Launch Day* period (one year) is almost equivalent to the total number of prefixes evolved during the last two decades. Even though we observe only half of the globally announced IPv6 prefixes at our vantage point, this increasing rate is still significant.

When viewing the change of the IPv6 adoption in the perspective of the traffic volume, the increase of IPv6 traffic is even more drastic than the one shown in the prefix data. As we see in Figure 3.13, IPv6 traffic increased by more than 50 % on the

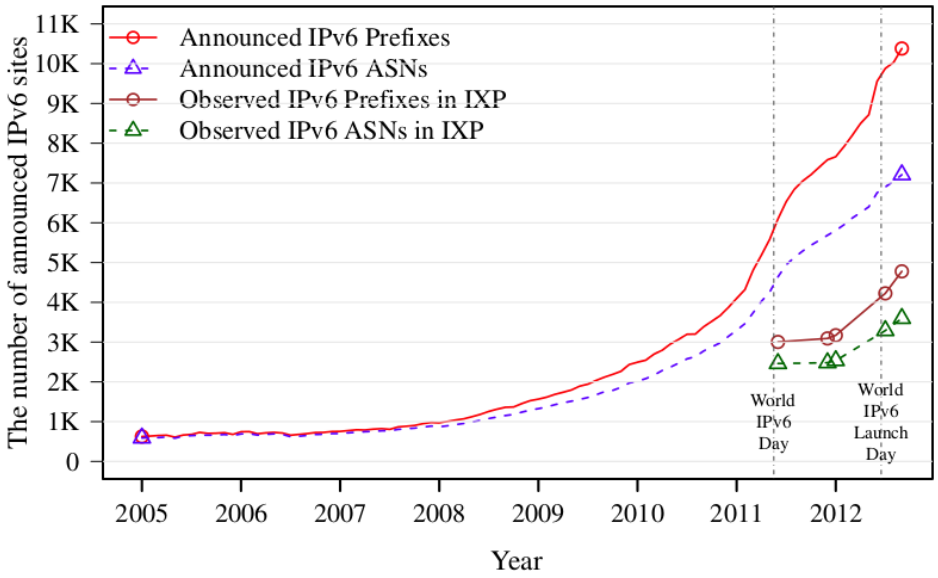
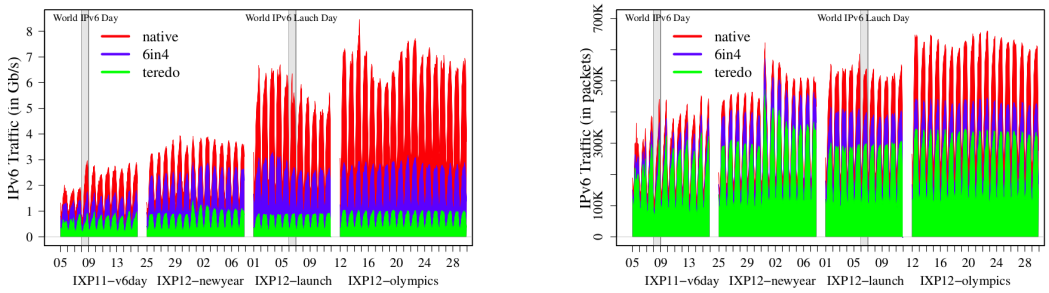


Figure 3.12: The growth of the IPv6 routing table since 2005 (shorter lines are the observation from our data sets during 14 months of measurement period). Lines from RouteViews data are plotted in monthly intervals and lines from IXP data are plotted five times within the period (each point represents the number of prefixes and ASes observed within the corresponding traffic data set).

World IPv6 Day. Moreover, the volume of IPv6 traffic again doubled (approximately from 3Gb/s to 6Gb/s) within the *World IPv6 Launch Day* period. We still observe this growth in our latest data set (IXP12-olympics).

The most significant change that we can discover from the figure is that the increase of IPv6 traffic mainly results from native IPv6 traffic. On the *World IPv6 Day*, we see more than twice the volume of native IPv6 traffic and it does not decrease after the event (even though it was supposed to be a 24-hour test flight). In addition, another multi-fold increase of native IPv6 traffic was experienced during the *World IPv6 Launch Day* period.

With regard to the sudden peak of teredo packets observed on 1 January, 2012 (see Figure 3.13 (b)), we do not have a clear answer for this phenomenon. However, we are of the strong belief that the cause of this peak is rather due to academic/industrial experiments than due to the participation in the *World IPv6 Launch Day* event. This assumption is based on three different observations. First, the number of teredo packets falls back to the normal level in the *World IPv6 Launch Day* period. Second, more than 40 % of the total teredo packets within the period are generated from a



(a) IPv6 traffic volume

(b) IPv6 packet counts

Figure 3.13: IPv6 traffic changes.

few IP addresses within the same IP prefix. Third, most of the teredo packets are bubble packets[7] in which there is no actual payload present.

Next, we evaluate how IPv6 traffic is distributed across prefixes. Figure 3.14 shows a Cumulative Distribution Function (CDF) of the IPv6 traffic contributed by the top-100 IPv6 prefixes. Solid lines in the figure indicate the traffic fraction of the prefix out of the total observed IPv6 traffic, while dashed lines describe only the fraction of outgoing IPv6 traffic. The figure shows that about 1 % (50 prefixes in IXP11-11d and 60 prefixes in IXP12-11d and IXP12-olympics) of the observed prefixes contribute more than 90 % of the total IPv6 traffic. Moreover, almost 30 % (IXP11-11d and IXP12-11d) and 20 % (IXP12-olympics) of the total traffic are contributed by a single prefix. Given the fact that almost all traffic of the prefix is outgoing traffic in IXP11-11d and IXP12-11d, we conclude (and also verify) that the top IPv6 contributor in these periods is a content provider. However, by observing the ratio between incoming traffic and outgoing traffic of the top prefix in IXP12-olympics, it appears likely that the first position in the traffic ranking is now taken by a large transit network. Indeed, by further dataset inspections we confirm the following assumption: The identified transit network is one of the largest IPv6 backbone network in the world.

Yet, readers must note that this evaluation is based on network prefixes. When prefixes are aggregated into the AS level, the content provider (the top IPv6 traffic contributor in IXP11-11d and IXP12-11d) is still the largest IPv6 traffic contributor in terms of the traffic volume.

Application Breakdown

In this subsection, we provide insights into the application mix in IPv6 traffic. Figure 3.15 shows the breakdown of the traffic into applications. Note that the header size of transition protocols (*e. g.*, teredo and 6in4) is included in the traffic volume.

[7] A teredo bubble is a signaling packet typically used for creating and maintaining a NAT mapping

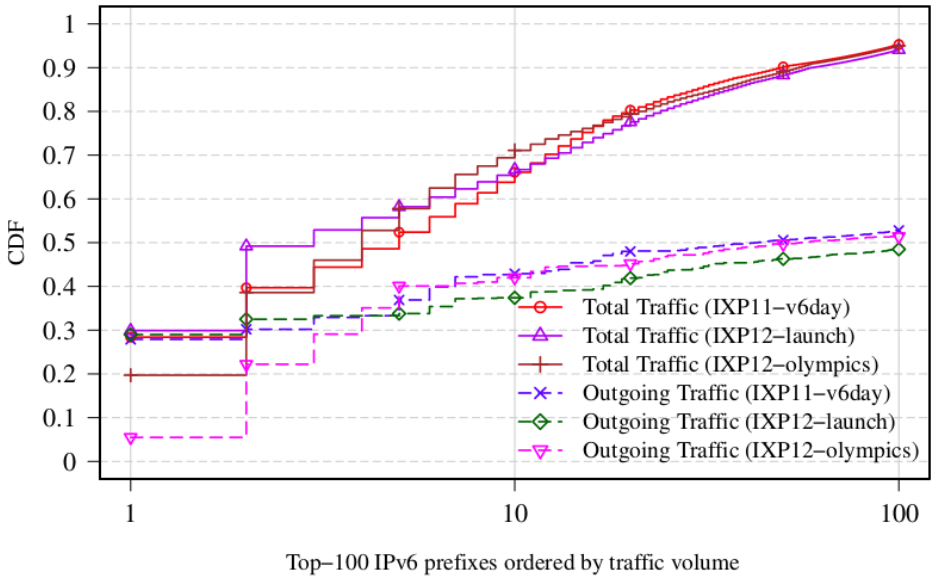


Figure 3.14: CDF of IPv6 traffic contributed by the top-100 prefixes. The prefixes on the x-axis are sorted by the IPv6 traffic contribution from left to right in descending order. A logarithmic scale is used on the x-axis to clearly identify points of the top prefixes.

Hence, we see a significant fraction of teredo traffic in the figure even though teredo bubbles[8] do not contain any actual payload.

The first application protocol that we need to pay attention to is the Network News Transport Protocol (NNTP). Even though the significance of NNTP traffic within the IPv6 network decreases drastically as the influence of web traffic increases, it is important to understand its characteristics since NNTP has been the dominant content carrier in the IPv6 world before HTTP became the major protocol and it is still responsible for a considerable fraction of the IPv6 traffic.

As we discussed in Section 3.1, NNTP was considered to be obsolete for a while before recent measurement studies [49, 61, 67, 106], including this thesis, discovered that NNTP is reviving (or survives). They report that NNTP accounts for up to 5 % of the total residential traffic in today's Internet.

More surprisingly, Sarrar *et al.* [102] report that almost 40 % of the total IPv6 traffic is contributed by NNTP before the *World IPv6 Day*. Our result shows a significantly different fraction (about 20 %) of NNTP traffic in the same period, but this is due to the fact that we consider the header bytes of the transition technologies as part of the application's traffic volume. When considering only the payload, our evaluation matches the one reported in [102].

[8] A teredo bubble is a signaling packet typically used for creating and maintaining a NAT mapping

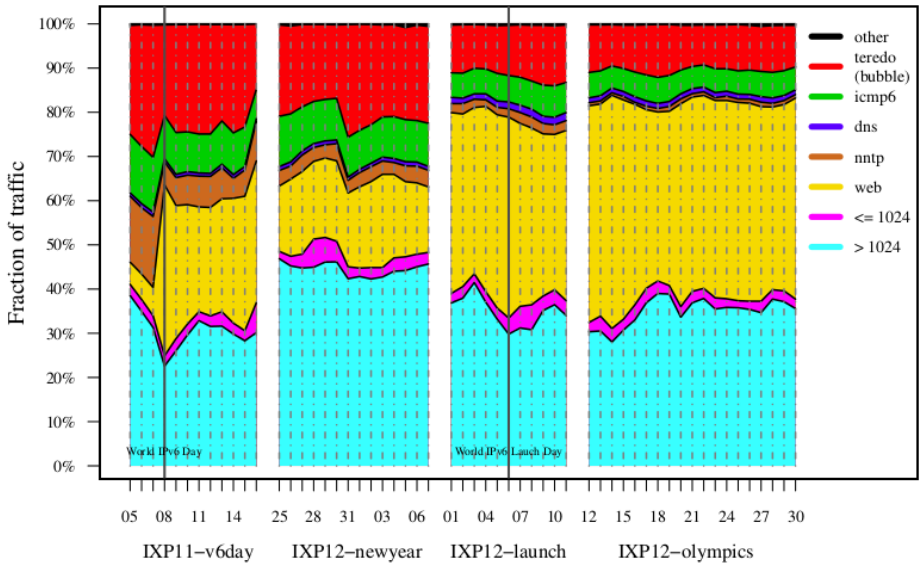


Figure 3.15: Application breakdown. Traffic volume includes the size of encapsulation headers.

The next application protocol that we study is HTTP. As the figure shows, the fraction of web traffic increased remarkably on the *World IPv6 Day* and reached nearly 50 % of the total IPv6 traffic in the *World IPv6 Launch Day* period. The change in traffic shape has a particular meaning since the breakdown of IPv6 traffic into applications becomes similar to the one of today's IPv4 traffic. Furthermore, the major share of IPv6 traffic is real content rather than signaling traffic. This leads us to the conclusion that service providers are beginning to break away from the idea that IPv6 is a faraway story and are becoming more active in providing Internet access through IPv6 to their home and enterprise customers.

Native IPv6 Traffic Among ASes

We now investigate how native IPv6 traffic is exchanged among ASes. For this study, we create a traffic matrix with sorted ASes as columns and rows (more than 3,500 ASes). Figure 3.16 depicts the cumulative fraction of native IPv6 traffic of AS-flows[9]. In the figure, we only consider AS-flows which contribute at least 0.001 % of the total native IPv6 traffic, *i. e.*, 1,386 AS-flows in IXP11-11d, 1,761 AS-flows in IXP12-11d, and 1,928 AS-flows in IXP12-olympics. The figure describes that the traffic fraction of AS-flows increases remarkably in the *World IPv6 Launch Day* period. Interestingly, the amount of IPv6 traffic is largely concentrated in one AS-flow (*i. e.*, the AS-flow labelled as 1 on the x-axis). However, the traffic contribution of this top AS-flow

[9] We define an AS-flow as an asymmetric pair of ASes

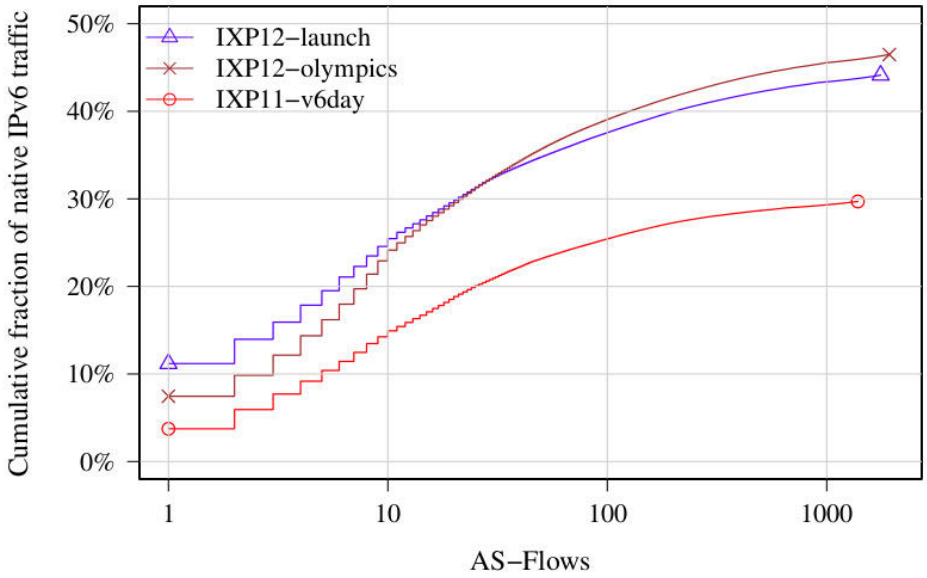
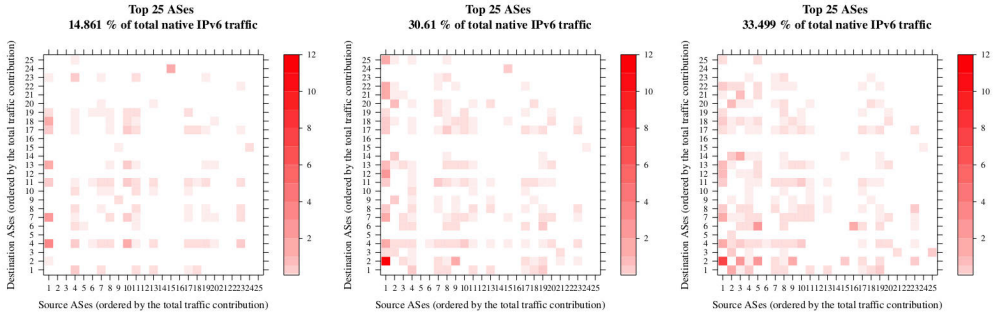


Figure 3.16: Cumulative fraction of native IPv6 traffic from an AS to another AS (AS-flow). Only AS-flows contributing more than 0.001 % of total native IPv6 traffic are considered. Links are sorted from left to right by the amount of traffic in descending order.

decreases from 11.19 % (IXP12-11d) to 7.46 % (IXP12-olympics), while the traffic share of those in the bottom 98 % of IXP12-olympics is larger than that of IXP12-11d. From this result, we infer that the IPv6 traffic relationship among ASes slowly changes from a 1-n shape to a n-n shape. This phenomenon is likely related to the rapid growth of the transit network explained in reference to Figure 3.14.

Before providing more information of this specific AS-flow, we narrow the scope of our investigation. We illustrate the matrix of the native IPv6 traffic fraction among the top-25 ASes in Figure 3.17. The cell in the figures are filled with different levels of color depth according to their share in the total native IPv6 traffic. The aggregate of IPv6 traffic transmitted among these top-25 ASes are 14.86 %, 30.61 %, and 33.50 % in IXP11-11d, in IXP12-11d, and in IXP12-olympics, respectively. By closer inspection of traffic, we verified that the cell filled with the deepest color in Figure 3.17 (b) and Figure 3.17 (c), *i. e.*, one representing the fraction of traffic originating from the AS numbered as 1 (x-axis) and destined for the AS numbered as 2 (y-axis), is the largest AS-flow illustrated in Figure 3.16. Furthermore, we can determine that the AS numbered as 1 is the major source of IPv6 contents. Adding up all native IPv6 traffic generated from this specific AS, we find that 15.21 % (IXP11-11d), 29.93 % (IXP12-11d), and 17.88 % (IXP12-olympics) originate from that single AS.

From a closer inspection of that AS, we identify that it belongs to one of the largest content providers in the world. This might not be surprising because the consumer



(a) IXP11-11d (b) IXP12-11d (c) IXP12-olympics

Figure 3.17: Traffic among top-25 ASes. ASes are sorted from left to right on the x-axis (from bottom to top on the y-axis) by the volume of contributing native IPv6 traffic in descending order.

behavior of Internet contents in the IPv6 world cannot be very different from the one within the IPv4 world. However, we believe that by studying these trends in IPv6 traffic and their core content providers we can help network operators to come up with more effective strategies for implementing IPv6 in their networks.

Readers may still wonder about the identity of the AS which consumes the most IPv6 traffic from the top content provider (the AS labelled as 2 in Figure 3.17). Regarding the largest AS-flow shown in Figure 3.16 and is also illustrated as the cell filled with the deepest color in Figure 3.17, we reveal that the top IPv6 traffic consumer in our measurement is one of the largest ISPs located in Romania. This is a somewhat unexpected discovery for us since Romanian Internet service providers were not particularly notable in the Internet history before. Taking a closer look at the traffic of this specific ISP in the peak period of our IPv6 traffic (IXP12-olympics), approx. 3.02 % of the total native IPv6 traffic is going out from this AS, while about 14.23 % of the total traffic is going into the AS.

3.3.3 Public Reports on IPv6 Traffic

Various content providers and service providers have been reporting the status of IPv6 traffic shown in their networks. In this subsection, we select three relevant reports and summarize them in relation to our measurement results.

Google

Google has been collecting statistics about the IPv6 connectivity of their users and reporting these results since 2008 [128]. They claim that 0.34 % of their total users

accessed their website over IPv6 on the *World IPv6 Day* and that the fraction increased to 0.65 % on the *World IPv6 Launch Day*. Given the fact that Google is the top ranked website in terms of number of visitors and the traffic volume, these fractions indicate a considerable number of users. Our measurement verifies that Google is one of the biggest sources of IPv6 traffic.

Another observation they have made is that since March 2010 the fraction of IPv6 users using native IPv6 technology has overtaken that using encapsulation technologies, *i. e.*, teredo and 6in4. According to their report, this decrease in users of transition technologies has become more drastic over time. As a result, the share of users behind transition technologies has decreased from 11.76 % (*World IPv6 Day*) to 1.54 % (*World IPv6 Launch Day*) of the total IPv6 users. Although our observation is based on the traffic volume, we confirm this trend of native IPv6 domination. More precisely, we observed that the fraction of IPv6 traffic transferred using transition technologies decreased from 55.49 % (*World IPv6 Day*) to 46.74 % (*World IPv6 Launch Day*). From our measurement viewpoint, the amount of native IPv6 traffic has begun to outstrip the amount of IPv6 traffic using transition technologies since the *World IPv6 Day*.

Hurricane Electric

Hurricane Electric is a global Internet backbone and one of the largest networks in terms of number of customers as well as the largest IPv6 network in terms of the number of connected networks. According to a report of Hurricane Electric [127], in the middle of 2012, 85.4 % of the global Top Level Domains (TLDs) have IPv6 name servers. In an investigation of the top 1,000 Usenet servers, they discovered that 16.22 % of them have IPv6 addresses. Given the fact that the Network News Transport Protocol (NNTP, the protocol used for Usenet) accounts for up to 5 % of the residential network traffic [59, 67], the reported number of IPv6-enabled Usenet servers may produce a substantial fraction of IPv6 traffic. Indeed, Sarrar *et al.* [102] report that almost 40 % of the total IPv6 is NNTP traffic before *World IPv6 Day*. We also observe a considerable amount of NNTP traffic from our measurement.

APNIC

The Asia Pacific Network Information Centre (APNIC) is the first regional Internet registry whose IPv4 address pool has been exhausted. APNIC measures the deployment level of IPv6 based on the IPv6 preference for IPv4 addresses using BGP data. APNIC's report describes that Europe has the highest IPv6 preference (0.51 in July, 2012) for IPv4 addresses among all continents. With the same metric, they report that Romania is the highest ranked country in the world. Similarly, the snapshot of Google's IPv6 statistics in the same period shows that 8.38 % of their users from Romania access their website using IPv6, which is the largest among all countries.

We confirm that more than 17 % of the total IPv6 traffic in the peak period of our measurement was contributed by a Romanian ISP.

3.3.4 Discussion

Throughout our measurements and from reports of various network operators, we observe that there is no immediate prospect of IPv6 transition. However, several findings in our analysis suggest that service providers and content providers are collaborating to act together towards getting the transition done. Besides, considering that actual content is transferred using popular protocols such as HTTP and NNTP, which are also highly influential content transfer protocols found in IPv4 traffic, users seem to start adopting IPv6 as the network protocol.

3.4 Related Work

There are countless research activities for understanding today's Internet usage as well as for characterizing Internet traffic. Although some work such as Ager *et al.* [2] and Chatzis *et al.* [14] study the general traffic behavior from the highly aggregated monitoring point (*e. g.*, IXP), it is virtually impossible to see the complete picture of the Internet traffic from a single viewpoint since the Internet is huge, diverse, and highly complex. Therefore, many researchers divide the branch of the study into multiple sub categories based on different criteria and find their characteristics one by one. For example, in accordance with the aim of this chapter, we split the study (Internet traffic measurement) into sub studies on multiple protocols and analyzed three influential protocols out of them (*i. e.*, NNTP, World of Warcraft, and IPv6). In this section, we discuss recent traffic measurement studies related to each of our analysis.

To the best of our knowledge, the NNTP analysis presented as a contribution of this chapter is the first study of NNTP traffic since the advent of blogs and OSNs that provide NNTP like features in a shiny browser-based dress. However, in different research areas, a handful of work has been done for the same overlay network (*i. e.*, Usenet). For example, Gryaznov [36] presents the statistics of malware postings on Usenet. The author reports that the number of malware postings dropped drastically in 2005 due to the start of malware filtering and the introduction of the NNTP authentication. Smith *et al.* [108] present the feasibility study of the content replication based on NNTP (and SMTP) infrastructure. The authors show in the paper that their replication method allows digital content to be archived in NNTP (and in SMTP) infrastructures for long term.

Within the MMORPGs, previous studies have mainly focused either on characterizing gaming traffic behavior [15, 110] or on determining the users' playing patterns [56, 73, 89, 111, 114]. Although some of findings in our World of Warcraft traffic

analysis overlap studies of the former category, we emphasize that our work falls more into the latter category as the novelty of our findings mainly lies on the gaming behavior analysis of solitary users and group users.

Szabó *et al.* [111] study how gaming traffic is influenced by various in-game activities of game players by deeply inspecting MMORPG traffic. Suznjevic *et al.* [110] evaluate types of actions generated by players within the virtual world. Kihl *et al.* [56] report that 20 % of the households in their measurement environment (a Swedish broadband access network with 12K users) has active World of Warcraft players and that their average playing duration is 2.3 hours per day. Our work is complementary to their work since we perform our measurement based on the traffic traces collected from the topologically similar network in 2008 and 2010, while they conduct the measurement on traffic collected in 2009. Chen *et al.* [15] analyze ShenZhou online game traffic collected at the server side and reports that MMORPG traffic shows irregularities due to the players' drastically diverse gaming behaviors. Varvello *et al.* [114] study avatars' social behavior in Second Life. They find that approximately 0.3 % of the total subscribers are playing the game concurrently at any point of time and that they do not move 90 % of the connected time. They also find that avatars in Second Life tend to organize small groups (2 to 10 avatars). Pittman *et al.* [89] study the population dynamics of virtual worlds over time and the players' movement patterns in the virtual world. Miller *et al.* [73] analyze the avatars' movements within the virtual world and they find that 5 % of visited territory accounts for 30 % of all time spent.

Most of the studies made before the announcement of the name space exhaustion focused mainly on performance analyses of various transition techniques [97], IPv4 vs IPv6 comparison [79], and the deployment level of IPv6 based on the address reachability [20, 53, 69]. This tendency, however, became less pronounced as time passes. Instead, researchers start to take a close interest in the characteristics of IPv6 traffic based on real-world traces [34, 102].

Malone *et al.* [69] quantified the number of IPv6 addresses accessable in the Internet based on traffic data collected within specific websites and DNS servers. They used prefixes of identified IPv6 addresses in order to classify IPv4-to-IPv6 transition techniques. Karpilovsky *et al.* [53] also measured the number of publicly announced IPv6 prefixes by analyzing snapshots of BGP data obtained from Route-Views Project. They reported that RIPE, *i. e.*, Regional Internet Registry (RIR) for European countries, is the increasingly dominant registrar for IPv6 address allocation. Colitti *et al.* [20] analyzed passively collected data from Google's IPv6-enabled web page and evaluated the degree of the IPv6 adoption. Their evaluation shows that the adoption of IPv6 is still low but steadily growing and that the vast majority of IPv6 traffic is contributed by only small number of networks.

Gao *et al.* [34] proposed the method to classify the P2P traffic from flow-baed IPv6 traffic and evaluated the stage of the IPv6 deployment in China. They found that P2P and streaming applications are accounting for the major fraction of IPv6 traffic

in China. Nikkhah *et al.* [79] assessed the performance of IPv6 and compare it to the performance of IPv4. They claimed that the inefficient pathfinding is the major cause of the poor performance that IPv6 shows. Sarrar *et al.* [102] observed changes of IPv6 traffic on *World IPv6 Day* by analyzing traffic traces collected from the same monitoring point that our study bases on. Zander *et al.* [146] proposed a method to mitigate sampling error and report results of IPv6 adoption from clients by testing their readiness with a web-based technique. Finally, Dhamdhere *et al.* [23] studied the deployment level of IPv6 and compare the topological modification of IPv6 with that of IPv4. One of the interesting findings in their work is that paths taken from clients to dual-stacked servers are largely different in IPv4 and IPv6.

3.5 Summary

Throughout our measurement studies, we observe the clear contrast and/or similarity between disparate protocols (*i. e.*, NNTP vs. World of Warcraft, and IPv4 vs. IPv6).

First, in IPv4 traffic, we find that an old text-friendly protocol (NNTP) is actively (mis)used as an alternative file-sharing platform by relatively small number of users, and it is responsible for a big portion of Internet traffic today. On the contrary, we witness that a relatively young protocol such as a World of Warcraft has a large user base, but does not generate a large volume of traffic (thus not bandwidth intensive).

From the analysis of NNTP traffic, we find that more than 99 % of the NNTP traffic is binary data transmission and the distribution of transaction volumes shows that around 80 % of all transaction sizes are at two distinct peaks (252 kB and 387 kB). Furthermore, the achieved throughput of NNTP connections is at least an order of magnitude higher than P2P-systems like BitTorrent and eDonkey, which is likely the reason why NNTP is popularly used for file-sharing.

From the analysis of World of Warcraft, it is evaluated that more than 60 % of the total packets that clients send to the server are for updating coordinates of in-game objects, thus generated by user's mouse clicking or key pressing actions. By observing differences between solitary users (Tigers) and users who play the game together with other users behind the same middle box (Lions). We find that Tigers tend to play the game longer than Lions, while Lions travel longer distance in the virtual world than Tigers.

Second, by analyzing IPv6 traffic, we observe the domination of HTTP and a remarkable contribution of NNTP, which is the same in IPv4 traffic. These findings are important for understanding the current situation of IPv6 adoption since this result implies that the actual content is transmitted within IPv6 traffic.

Even though we observe that IPv6 traffic accounts for only small fraction of the total Internet traffic (0.5 % in the peak period within our measurement), our study

on IPv6 traffic show that the deployment of IPv6 technology is finally on the right track. This claim is based on the sharp increase rate of IPv6 prefixes and IPv6 traffic shown since *World IPv6 Day*. Furthermore, the increase rate does not slow down after the the *World IPv6 Launch Day* event. To this end, it is expected that the continuous effort on the IPv6 deployment such as *World IPv6 Day* and *World IPv6 Launch Day* will increase this trend.

4

Locator/ID Separation Protocol (LISP)

One of the main contributions of the previous chapter was the exploration of a diversity in Internet application and protocol usage. In this chapter, we shift the focus to another type of network diversity, namely, multihoming (we refer to it as upstream link diversity in this thesis) and cope with the scalability problem of the core Internet mainly caused by this diversity.

Today, a major fraction of stub networks are multihomed. This phenomenon, on the one hand, suggests a high degree of path diversity in the Internet that leads to network resilience. On the other hand, however, it leads to scalability issues, mainly concerning the continuously increasing size of the BGP routing tables [122] in the core Internet. In order to deal with the problem, the research community explores alternative architectures. In particular, the Locator/ID Split paradigm, based on the idea of separating the identity from the location of the end-systems, is gaining momentum and seems to be the most promising candidate for a future Internet routing protocol.

A critical component of any Locator/ID Split approach, from a performance and resource consumption perspective, as well as from security and reliability points of view, is the *Cache*. Recall the overview of the Locator/ID Split mechanism and its key components given in Section 2.3.

Taking Locator/ID Separation Protocol (LISP), the most successful proposal currently under discussion at the IETF, as the reference protocol, this chapter evaluates the LISP Cache from perspectives of the performance, security, and resilience. First, we study the implications (cost, overhead, and security trade-off) of deploying LISP Cache at the border gateway from ISP's viewpoint. Second, we analyze the impact of

an initial cache-miss on a single TCP communication based on the OpenLISP [142] implementation. Finally, we evaluate the depth of cache-miss storm caused by a failure/recovery of an Ingress Tunnel Router (ITR) in a LISP-enabled stub network and propose a solution of it based on the mapping cache synchronization.

4.1 Evaluation of the LISP Cache

It is intuitively obvious that decoupling the core Internet from the rest of the Internet reduces the number of routing entries in routers of the core Internet. However, to what extent the burden on the core router will decrease is largely uncharted. Moreover, since most of the Locator/ID Split mechanisms adopt the cache for storing mappings between the two address spaces, it is crucial to investigate whether or not the size of the mapping cache will cause another type of burden in routers.

To this end, we first analyze the implication of deploying LISP with an emphasis on cache[1] size. This work also quantifies cache-misses caused by obtaining initial mapping information; its impact on the performance of the network communication will be further discussed in Sections 4.2 and 4.3.

We use two 24-hour packet-level traces collected from the ISP vantage point (Section 2.1.2), taken in different years, hence showing as well the possible yearly evolution in the performance.

Further, inspired by the security analysis presented in Saucez *et al.* [103], we examine what and how much would change if instead of running vanilla LISP, *i. e.*, the original LISP that requires a mapping only during encapsulation, *symmetric LISP*, *i. e.*, a modified LISP that verifies the mapping also during decapsulation, is used in order to increase security.

We also explore what kind of traffic is generating the highest number of cache-misses. This is not only important in order to understand the dynamics of the LISP Cache, but helps as well to understand which applications are mostly affected. Our results are important for ISPs in order to identify and quantify the resources needed to deploy LISP, with respect to the level of resiliency that they want to guarantee.

4.1.1 Life of a Mapping in the LISP Cache

Although it is critical for estimating the implication of deploying LISP in the system, the LISP specification does not provide any concrete timeout value of a mapping and leaves the choice (or responsibility) to the implementers and system administrators. LISP just associates to the mappings a Time-To-Live (TTL), whose default value is

[1] In the rest of the section we will use the term *LISP Cache* and *cache* interchangeably.

24 hours, which once elapsed mandates the router to re-request the mapping before continuing to use it.

It should be clear that since the entries in the cache can expire, they are also entered in an on-demand fashion. This makes the cache critical for the performance and scalability, since its content, size, and efficiency are totally traffic driven. In particular, the first outgoing packet, destined to an EID for which there is no mapping in the cache, triggers a cache-miss. Such an event, in turn, triggers a query message that the ITR sends to the Mapping Distribution System.[2] The latter is a lookup infrastructure designed to retrieve the mapping for the destination EID of the packet that triggered the query. This is the same principle of DNS, which allows to retrieve the IP address(es) of a server from its FQDN (Fully Qualified Domain Name). So far, the research community has proposed several Mapping Distribution Systems, but only LISP-ALT [25] and LISP-DDT [50] are officially supported by the LISP Working Group in the IETF. A comparison of different mapping systems can be found in the work of Jakab *et al.* [51] and Mathy *et al.* [71].

4.1.2 Symmetric LISP

The attentive reader will have noticed that there is a fundamental asymmetry in LISP when performing encapsulation and decapsulation (see Section 2.3). Indeed, while doing encapsulation, the ITR uses the cache to select what to put in the outer header, whereas on the contrary, the ETR does not use the cache while decapsulating the packet. Rather, it just performs the check described in Section 2.3.4 for the LISP Database and eventually decapsulates the packet.

[2]Note that, in case of a cache-miss, the packet that triggered it cannot be encapsulated, since there is no mapping available. The LISP specifications do not explicitly describe what to do with the packet, however, it is out of the scope of the present work to evaluate this particular aspect.

Algorithm 1 Vanilla LISP

```
if packet.dir = to_internet then
    if Dest. EID mapping exists in xtr's cache
    then
        send(encapsulate(packet))
    else
        drop(packet)
        Map Request for dest. EID
        Cache obtained mapping
    end if
end if
```

Algorithm 2 Symmetric LISP

```
if packet.dir = to_internet then
    if Dest. EID mapping exists in xtr's cache then
        send(encapsulate(packet))
    else
        drop(packet)
        Map Request for dest EID
        Cache obtained mapping
    end if
else if packet.dir = to_local_network then
    if Source EID mapping exists in xtr's cache
    then
        send(decapsulate(packet))
    else
        drop(packet)
        Map Request for source EID
        Cache obtained mapping
    end if
end if
```

This asymmetry leaves the LISP protocol vulnerable to specific attacks on the ETR, as pointed out in the work of Saucez *et al.* [103]. The authors describe attacks exploiting data packets as well as attacks that leverage on fields of the LISP header. By analyzing different forms of attack, the authors conclude that to increase the level of security of the whole LISP architecture, the best solution is to use a symmetric model with a drop policy. In other words, the authors suggest to allow encapsulation *as well as* decapsulation only if mappings are present in both the LISP Cache *and* the LISP Database. Otherwise, when the mapping is not in the cache, a miss is generated and the packet is dropped. Examples of possible attacks that are facilitated by the asymmetric model are source EID and source RLOC spoofing, which can be used to build more sophisticated attacks leading to cache poisoning (*i. e.*, installing wrong information in the cache) or cache overflow (*i. e.*, exhausting the cache memory space). The packet handling procedures of these two LISP models are summarized in pseudo-code form in Alg. 1 and Alg. 2. The rationale behind the introduction of the symmetric model is the assumption that the mapping distribution system is secured and trusted, hence, sanity checks can be performed at both ends, when encapsulating and when decapsulating. For the decapsulation operation this means that if the packet contains information that is not consistent with the content of the cache, then it can be dropped. The implications of introducing such a symmetric model to increase security are two-fold. On the one hand, doing additional checks when carrying out decapsulation may reduce the performance, however this is a common price for security mechanisms. On the other hand, this means that the size of the cache, its dynamics, and the control traffic overhead change.

In the first part of the present chapter, we will refer to the symmetric model as *symmetric LISP* and we will assess its impact on cache size and cache-miss rate as compared to vanilla LISP in order to evaluate what are the trade-offs to increase security in the LISP protocol.

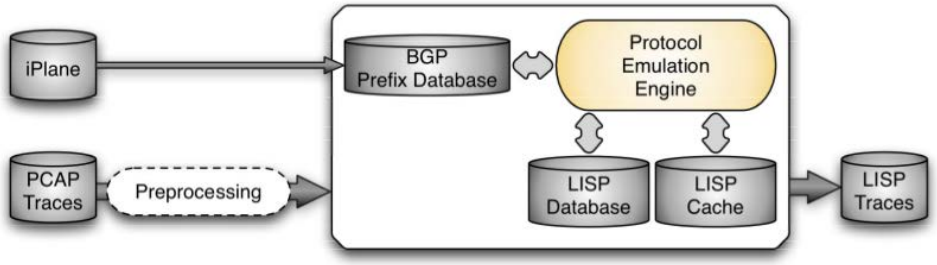


Figure 4.1: Structure of the Locator/ID Split emulator.

4.1.3 Design of the LISP Cache Emulator

In order to evaluate the Locator/ID Split paradigm, using the specifics of the LISP protocol, we implemented software that emulates the complete tasks of LISPs' xTRs. The emulator is essentially designed to be fed with pcap-formatted traffic data and to reproduce the LISP architecture; hence implementing the two main modules (see Section 4.1), namely the LISP Database and the LISP Cache.

The LISP Database is a manually configured list of internal network prefixes (EIDs), while the LISP Cache stores an EID-to-RLOC mapping when there is a request for it and removes it according to the preconfigured expiration rule. In addition to these two modules there is a central logic, *i. e.*, protocol emulation engine, that creates the statistics and periodically writes them to a text file.

In addition, we use a local BGP prefixes database fed with the list of BGP prefixes published by the iPlane Project [66]. The database is used to group EID-to-RLOCs mappings with the granularity of existing BGP prefixes, because, as of today, there is insufficient information to predict the granularity of mappings in a LISP-enabled Internet. This BGP granularity follows the methodology proposed in [46].

4.1.4 Cache Scalability Analysis

A key benefit of LISP is the reduction in routing table size in the DFZ [93], however there exists a trade-off between the reduction in routing table size and the LISP Cache size. The analysis of caching *cost* is important for estimating the actual benefits of LISP.

For experiments performed in this section, we feed our LISP Cache emulator with anonymized packet-level traffic data collected within a large European ISP for 24 hours in April 2009 (APR09) and in March 2010 (MAR10) from our ISP vantage point. The use of traces from two different years allows us to evaluate the yearly evolution that the LISP behavior has undergone. Our packet-level traces, as well as

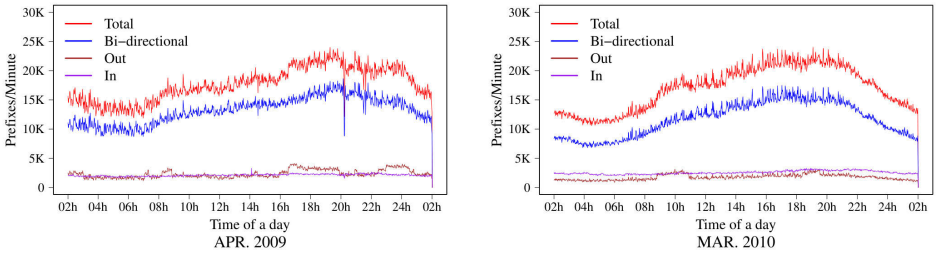


Figure 4.2: Report of the number of correspondent prefixes per minute.

the iPlane BGP prefixes (*cf.* Section 4.1.3) are anonymized using the same cryptographic key in such a way that it preserves the BGP aggregation granularity of IP addresses specified in packet's IP header. This is a crucial in order for us to conduct our emulation since there is no real EID aggregation model, and the BGP-like model remains the most reasonable. To the best of our knowledge, there is no publicly available traffic traces in which the BGP aggregation granularity is preserved and is usable.

The first step in our analysis is to identify and characterize the working set. In our specific case this is represented by the number of observed BGP prefixes in our measurement environment. Section 4.2 shows the evolution of the total number of contacted prefixes per minute, as well as the breakdown of incoming, outgoing, and bi-directional prefixes. An incoming prefix is the prefix of a remote network from which packets between the remote network and the local network are only sent from the remote network (an outgoing prefix is vise-versa). A bi-directional prefix is the prefix of a remote network in which packets between the remote network and the local network are sent in both directions.

We will observe in Section 4.2 that the traffic pattern did not change much over a one year period. The most noticeable change concerns prefixes of outgoing only traffic, which have decreased.[3] Such a decrease can be explained by the continuous reduction in P2P traffic that has been observed in the last years, hence reducing random scans. In both traces the large majority of the observed prefixes (*i. e.*, about 70 % in average in all traces) are bi-directional. In the remaining part of this section, unless differently stated, we will show results concerning the most recent trace (*i. e.*, MAR10) since there is no relevant difference between the two traces. We will explicitly highlight differences in case of presenting results of the both traces.

One thing to keep in mind is that, in the case of vanilla LISP, incoming packets do not play any role in the LISP cache, as explained in Section 4.1.1. On the contrary, when using symmetric LISP, incoming prefixes have an impact on the cache since the ETR does need a mapping as well. Hence, 12.6 % (average value in all traces)

[3]Note that this is not related to the traffic volume, but just expresses the diversity of contacted prefixes.

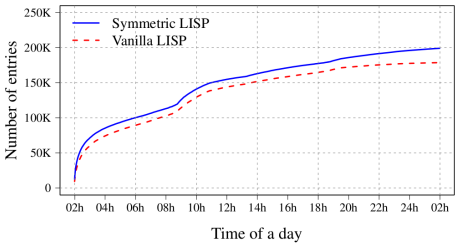


Figure 4.3: Growth of the LISP Cache when original TTL (24 hours) is used. MAR.2010.

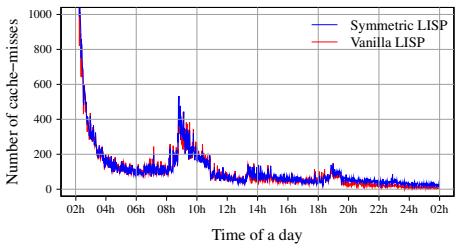


Figure 4.4: Cache-miss rate when original TTL (24 hours) is used. MAR.2010.

of incoming prefixes are a key differentiation between the two versions of LISP. This means that symmetric LISP maintains more entries in the cache.

As previously described in Section 4.1.1, the current LISP specification [26] defines 24 hours as the default period during which mappings can be used before mandating a new request in order to continue using the mapping. This translates to the fact that entries in the cache can in principle be kept for as long as 24 hours. However, such an approach can lead to very large cache sizes, as shown in Figure 4.3. The figure depicts the evolution of the number of entries in a LISP cache when all mappings are considered valid for 24 hours. Both symmetric and vanilla LISP see their caches growing restlessly, reaching the same order of magnitude as the current BGP routing table, hence very large, thwarting the original motivation of the introduction of the Locator/ID Split paradigm. The only advantage of such large cache is that it minimizes the number of cache-misses. Figure 4.4 shows the evolution of the number of cache-misses per minute. As can be seen, after the initial bootstrap period the cache-miss rate falls lower than a hundred per minute. The peak of cache-misses at around 9 am, corresponding to the steepness in the growth of the size, can be explained by the beginning of the working day.

From what has been shown so far, it is clear that the introduction of cache timeouts smaller than the original LISP TTL will reduce the size of the cache, improving its scalability, at the cost of an increase in the number of cache-misses. We explore this trade-off using three different cache timeout values, respectively 60 seconds, 180 seconds (*i. e.*, three minutes), and 1,800 seconds (*i. e.*, 30 minutes), similar to the work in [46]. The reason why we choose 60 seconds, instead of 300 minutes as in [46], is because that work already proved that the 300 minutes timeout value is inefficient considering the cache hit-rate vs. the cache size.

Size vs. Hit-rate Trade-off

Figure 4.5 illustrates the number of entries and the size of the cache measured in one-minute interval for vanilla LISP using the three different timeouts. The figure

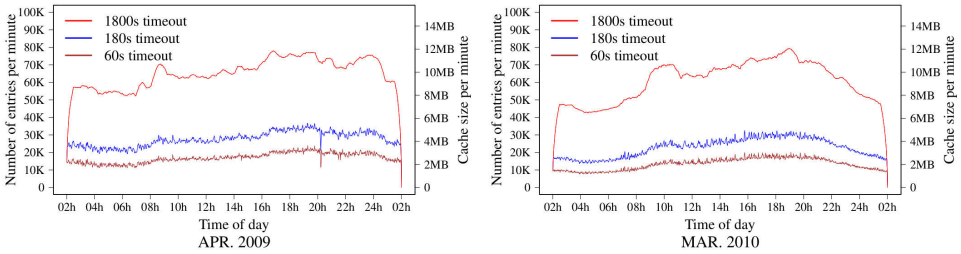


Figure 4.5: Number of entries and cache size (assuming two RLOCs) for vanilla LISP.

has two measurement scales for the y-axis in the plot. The left scale indicates the number of cache entries, while the right scale indicates the size of the cache expressed in MBytes. The drop in the cache size at the end of the plot is caused by the fact that we run the emulation until all cache entries are expired due to the timeout. Comparing the result of MAR10 data with Figure 4.3, we see that the maximum cache size decreased to approximately 10 %, 15 %, and 45 % of the maximum cache size of Figure 4.3 when the TTL is set to 60 seconds, 180 seconds, and 1,800 seconds, respectively.

We observe that the number of entries and the size of the cache are small for the 60 and 180 seconds timeout cases (respectively slightly more than 10,000 and 20,000) and their curves show a spiky behavior. The 1,800 seconds timeout is much larger but exhibits smoother changes, however, it takes about 30 minutes to reach the stable working set (almost 60,000 entries). Interestingly, while for the 1,800 seconds timeout the evolution of the cache is practically unchanged in both traces, for shorter timeouts the cache size seems slightly reduced in the 2010 trace. In other words, one year later, with a very short timeout, the cache would be smaller. This means that at small time scale the traffic pattern has evolved in a way that favors LISP with small cache timeouts.

Different from [46], the size of the cache is calculated using the size of the data structure of a real LISP implementation, namely OpenLISP [142]. In OpenLISP each entry of the cache is a radix node containing all the information of the mapping and pointing to a chained list of RLOCs nodes. Thus, the size C_{size} of the cache is given by: $C_{size} = N_E \times (Radix_s + N_R \times RLOC_s)$. Where N_E and N_R represent respectively the number of entries in the cache and the number of RLOCs per entry. $Radix_s$ is the size of a radix node (56 bytes in OpenLISP), while $RLOC_s$ is the size of the RLOC node (48 bytes in OpenLISP). The results in Figure 4.5 are obtained assuming two RLOCs per mapping.

The previous results show that the selection of an appropriate timeout value is of prime importance for the efficiency of the cache. Thus, to have a deeper understanding of the efficiency of the cache we investigate the ratio between cache-misses and cache-hits. Figure 4.6 depicts the number of hits and misses for the different timeout

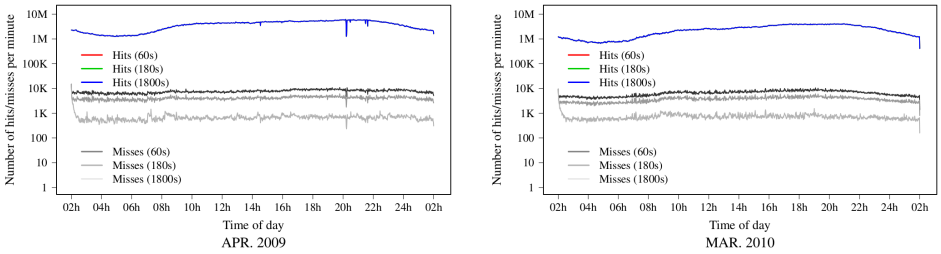


Figure 4.6: Evolution of the number of hits and misses for the different timeout values.

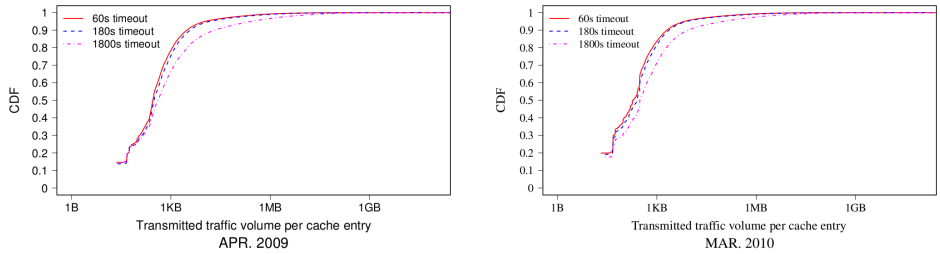


Figure 4.7: Cumulative Distribution Function of the traffic volume per cache entry.

values. Even with the y-axis in logarithmic scale it is not possible to distinguish the lines of the cache-hits, since they are overlapping each other. Hence, we need to look at the difference in the number of cache-misses. Despite the identifiable difference, it is difficult to avoid the claim that the benefit from the longer timeout value is insignificant compared to its cost in terms of size. Our calculations show that the average hit ratio (resp. average miss ratio) is always higher than 99 % (resp. lower than 1 %) for every timeout value, while the size can be more than 5 times bigger when we compare the 60 seconds case with the 1,800 seconds case.

To confirm that a large cache is not useful, we plot the CDF of the traffic volume forwarded by each entry in Figure 4.7, showing that the vast majority of cache entries carry less than 1 MBytes of traffic. Furthermore, Figure 4.8 shows how at least 50 % of the entries have a lifetime only slightly longer than the timeout value. Both CDFs are the consequence of the well-known Internet phenomena that a small number of prefixes are responsible for the majority of Internet traffic. Coupled with the evidence mentioned above, we draw the inference that the minimum timeout value (60 seconds) is the most cost-beneficial.

Traffic Volume Overhead

Another important aspect of LISP is the overhead introduced in terms of traffic volume. The overall overhead O can be evaluated by adding the overhead due to

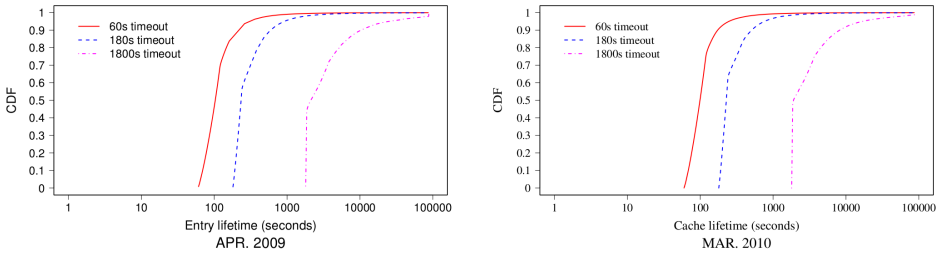


Figure 4.8: Cumulative Distribution Function of entries' lifetime.

the LISP encapsulation, *i. e.*, the number of packets N_P multiplied the size of the prepended header E_H and the volume of traffic generated to request and receive the missing mappings. Note that N_P is the number of packets effectively encapsulated by LISP, hence not counting the packets that trigger cache-misses, which are dropped, but counting their retransmission after the mapping is available in the cache. We assume one single retransmission in our emulations, which we include in N_P. Assuming that one single request/reply exchange is performed for each miss, the latter is given by the number of misses N_M multiplied by the size of the Map-Request message M_{REQ}, for outgoing traffic, and the size of the Map-Reply, for incoming traffic, which is given by a fixed size M_{REP} plus the size of each RLOCs record R times the number of RLOCs N_R. Note that the Map-Request and Map-Reply are the standard messages defined by LISP to request and receive a mapping from the mapping distribution system, independently from the specific implementation. Putting everything together, for outgoing traffic we have: $O_{out} = N_{P_{out}} \times E_H + (N_M \times M_{REQ})$; while for the incoming traffic we obtain: $O_{in} = N_{Pin} \times E_H + (N_M \times (M_{REP} + N_R \times R))$. From the LISP specifications [26] the size of E_H is 36 bytes, the size of M_{REQ} is (without any trailing data) 24 bytes, the size of M_{REP} (without any trailing data) is 28 bytes, to which we need to add 12 bytes for each RLOC record R. With these numbers, it is possible to plot the overhead in Figure 4.9, expressed in MBytes/sec. To clearly separate the traffic flowing in opposite directions, the line plotted in positive values (above zero) indicates outgoing traffic, whereas the line plotted in negative values (below zero) indicates incoming traffic. The shading over and under the traffic line indicates the traffic overhead.

This plot is based on a 60 seconds timeout, which represents the maximum possible overhead since it has the highest number of cache-misses, hence the larger control traffic to request/obtain the mappings. Even in this worst-case scenario, Figure 4.9 shows that the overhead is limited (less than 6 % according to our calculations) in all traces and does not represent a problem for links that are not close to saturation.

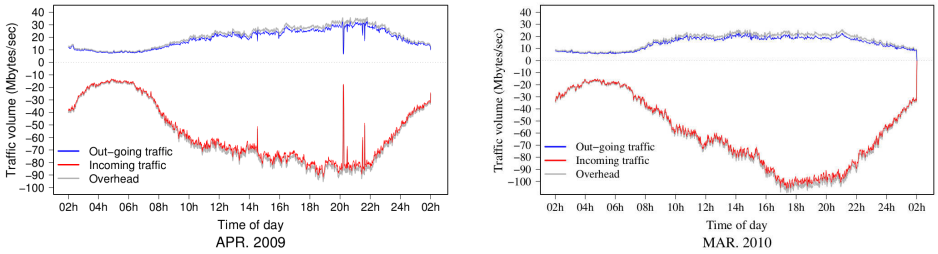


Figure 4.9: Traffic volume and overhead due to LISP (60 seconds timeout value).

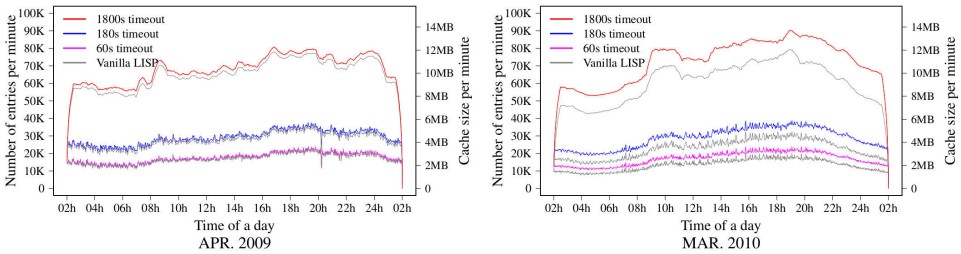


Figure 4.10: Extra overhead caused by the use of the symmetric LISP model.

Vanilla vs. Symmetric LISP

The results presented so far concern the basic variant of LISP, *i. e.*, vanilla LISP. Nevertheless, as discussed in Section 4.1.1, one of the contributions of this thesis is to estimate the extra cache size and the extra traffic overhead that the more secure symmetric model introduces. Since the extension of the mapping mechanism to incoming traffic is the main idea of the symmetric model, an increase in the number of entries and the size of the cache is expected and confirmed by the plot in Figure 4.10, where the overheads for both the 2009 and the 2010 traces is shown. For a clear comparison, the cache size presented in Figure 4.5 is presented again in gray color (lighter gray in black and white print). It is worth notice that the higher the timeout value, the lower the increase; we argue that it is due to the fact that longer timeouts increase entries' reuse probability. Interestingly, the size increase due to the incoming traffic is larger in the most recent trace (*i. e.*, MAR10), meaning that unsolicited inbound traffic has a more diverse pattern compared to the previous year.

This is confirmed by the results in Table 4.1, where the actual numeric increase is computed. It can be observed that there is an increase ranging from 6.8 % up to 19.2 % when the symmetric model is used. However, when comparing the increases by year (2009 vs. 2010) we notice that the size increase is more important in 2010 (almost the double of the increase in 2009). This is explained by the larger number of incoming packets that can be observed in the 2010's trace.

	APR09			MAR10		
Timeout	V.LISP	S.LISP	Diff.	V.LISP	S.LISP	Diff.
60 sec	3.47	3.76	+8.36 %	2.92	3.48	+19.18 %
180 sec	5.32	5.74	+7.89 %	4.71	5.61	+19.11 %
1800 sec	11.29	12.06	+6.82 %	11.50	13.09	+13.83 %

Table 4.1: Cache size (in Mbytes) increase due to security enhancement (two RLOCs)

Another important aspect to look at is how the distribution of cache-misses changes due to the fact that incoming packets can now also trigger them. Figure 4.11 compares the number of cache-misses under vanilla LISP and symmetric LISP. The most interesting result shown in the plot is the fact that the number of misses caused by incoming packets does not correspond to the increase in the number of cache-misses. Indeed, we observe that up to +27.6 % (60 seconds timeout in MAR10) of all cache-misses are additionally produced under symmetric LISP, but the percentage of cache-misses due to incoming packets is much higher, while the number of cache-misses due to outgoing packets shrinks. The latter phenomenon is caused by the fact that all the cache-misses caused by packets replying to externally initiated communications in vanilla LISP, translate in symmetric LISP in cache-misses for the incoming packets initiating the communication. When translated into traffic overhead, the increase of miss generates a small increase in bandwidth consumption. Nevertheless, after applying the same computation as for vanilla LISP, the increase in bandwidth is lower than 0.5 %, hence not critical, if not negligible.

Cache Size vs. Stub Network Size

To deepen our knowledge about the LISP Cache from the perspective of scalability, we look into the impact of the number of end-hosts, *i. e.*, the size of the stub network behind LISP's xTRs. To this end, we repeatedly conduct emulations where we increase the number of unique IP addresses (thus EIDs) from 1,000 to 20,000 with a 1,000 increase at each step. All in all, we perform 20 emulations for vanilla LISP and the same number of emulation for symmetric LISP. Figure 4.12 shows the growth in the number of entries in the LISP Cache. Each point represents the maximum number of entries (y-axis) per number of EIDs (x-axis). Several interesting observations can be made out of this set of emulation. Firstly, the size of LISP Cache does not even double, while the number of EIDs increases 20 times. Secondly, the difference between symmetric LISP and vanilla LISP grows negligibly as the number of EIDs increases. This indicates that the cache size of symmetric LISP grows only slightly faster than vanilla LISP. This slow increase is due to the fact that the traffic pattern characteristics do not change that much when increasing the number of EIDs. As Figure 4.13 shows, the number of different IP addresses that generate a cache-

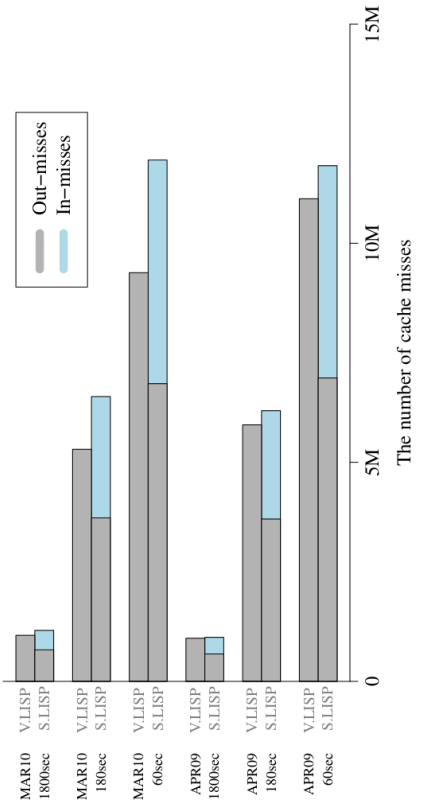


Figure 4.11: Cache-miss distribution of vanilla LISP and symmetric LISP.

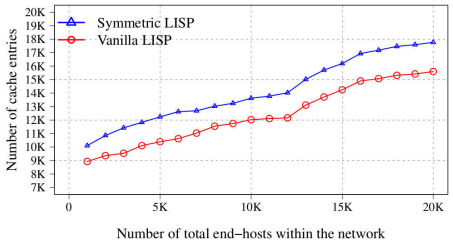


Figure 4.12: Number of the maximum cache entries over different numbers of end-hosts.

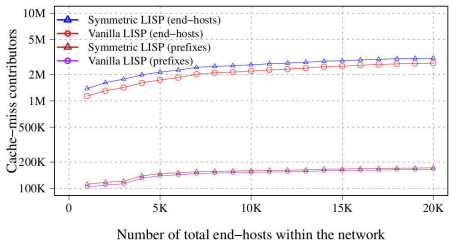


Figure 4.13: Number of destination prefixes causing cache-misses.

miss, which we indicate as cache-miss contributors, grows slowly with the number of EIDs in the local network. Even more, when these IP addresses are aggregated by network prefix the curve becomes almost flat.

Cache Size vs. Mappings Granularity

One might think that seeing current BGP prefixes as the namespace granularity of the future Internet is too optimistic. Thus, we extend our analysis to uniformed /24 prefixes (*i. e.*, an assumption that the Internet address space is split into flat /24 prefixes), as well as uniformed /27 prefixes, as the worst-case scenario. The emulation with a finer fragmentation than /27 is not possible due to the capacity limit of our analysis server. We perform emulation only on the busiest hour of the day (MAR10), *i. e.*, with larger cache size. Figure 4.14 and Figure 4.15 show respectively the cache size and the number of cache-misses with the two different EID-to-RLOC uniform granularities for both vanilla and symmetric LISP. For reference purposes the figures

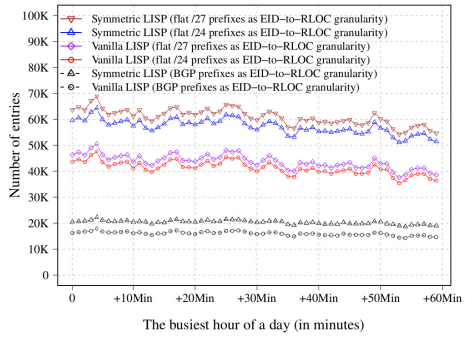


Figure 4.14: Cache size in the busiest hour with uniform mappings granularity.

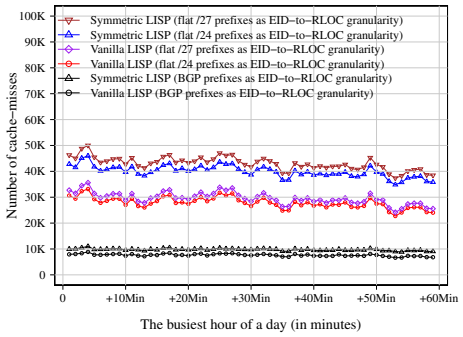


Figure 4.15: Cache-misses in the busiest hour with uniform mappings granularity.

contain the same curves for the BGP granularity. We can make two inferences from this evaluation. Obviously, the uniform granularity leads to an increase in both size and miss ratio. However, cache scalability is still better than compared to pure BGP, since the BGP table in case of /27 only prefixes would contain approximately 116,881,408 prefixes. Furthermore, the difference in both size and miss rate between vanilla and symmetric LISP is increasing when the mappings granularity becomes smaller. Although the latter may be biased by the specific traffic behavior of the given ISP or the given trace, the result is remarkable as the overhead increase of symmetric LISP in finer fragmented granularity is significant.

Cache-miss Rate vs. Applications

From the analysis presented so far, it is clear that an important factor in LISP is the cache-miss rate. Hence the importance of having a deeper look at it. Packet loss (due to the cache-miss) may have different effects on diverse applications, thus it is important to know which class of applications generates most of the overall cache-misses (*i. e.*, experiences the largest number of packet drops). Indeed, when we look at the number of packets that are forwarded per cache entry we observe that a large portion of them are used per one single packet. Figure 4.16 clearly shows that 20 % up to 50 % of cache entries are used by only one packet. In principle, this is the same packet that generated the cache-miss and has been either buffered or retransmitted.

This result strengthens our suspicion that a large portion of cache-misses is generated by the port scanning activity or lookup activities of P2P applications. To better understand this, we classify the packets that generate a cache-miss according first to the transport layer protocol and then to the destination port number.

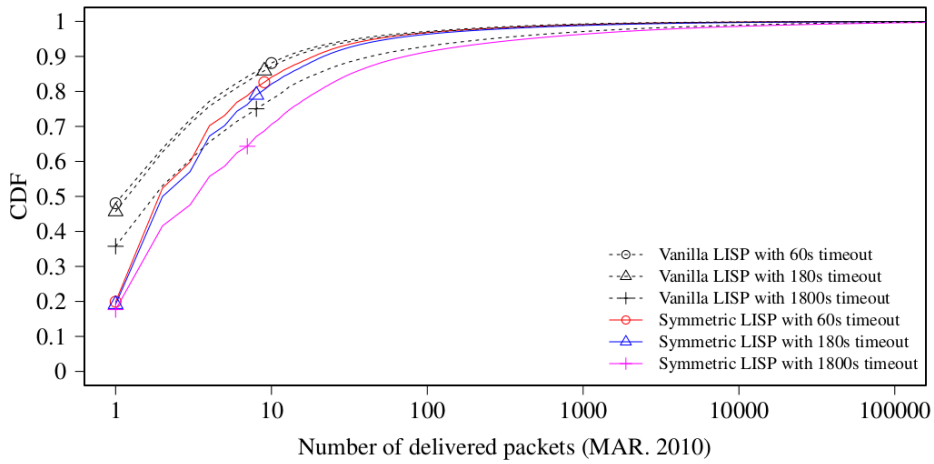


Figure 4.16: CDF of the number of forwarded packets per cache entry.

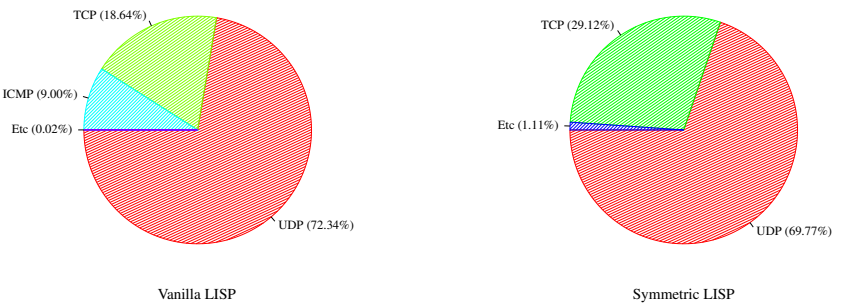


Figure 4.17: Protocol breakdown of packets generating cache-misses (Mar. 2010).

Figure 4.17 shows the transport layer protocol breakdown of cache-miss packets. The fact that UDP is a dominant protocol (approx. 70 %) of cache-miss packets demonstrate that indeed most of the cache-misses are triggered by P2P traffic, since well-known P2P applications such as eDonkey and BitTorrent are performing file/peer scans using this protocol.

Nevertheless, Figure 4.18, which shows the port number breakdown, gives another perspective. To help reading the figure, we provide in Table 4.2 the list of services/applications using a specific port. It turns out that TCP/80 (Web traffic) and TCP/445 (SMB traffic) are the statistically dominant cache-miss triggers. This implies that port numbers of UDP packets that are causing cache-misses are widely distributed. For this reason we grouped them in the "Etc." slice in Figure 4.18. Again reinforcing the argument that large majority of cache-misses are caused by P2P traffic, since it

Table 4.2: Lisp of port numbers per application/service

Port	Applications	Port	Applications
TCP/25	E-Mail (SMTP)	UDP/16001	–
TCP/80	Web (HTTP)	UDP/16630	–
TCP/443	SSL	UDP/22802	–
TCP/445	SMB-over-TCP/IP	UDP/27015	Steam, Counter Strike (Game)
TCP/6881	BitTorrent	UDP/33410	–
UDP/53	DNS	UDP/35691	–
UDP/123	Network Time Protocol	UDP/51413	BitTorrent
UDP/3283	Apple Remote Desktop	UDP/59230	–
UDP/4672	eMule (eDonkey)	UDP/61610	–
UDP/6881	BitTorrent		

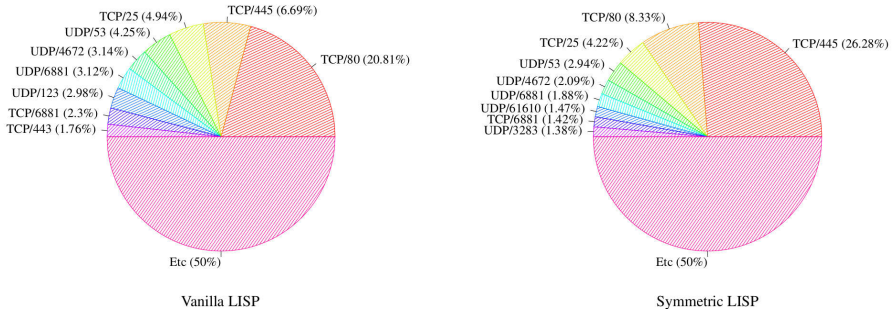


Figure 4.18: Breakdown by port number of cache-miss packets (Mar. 2010).

is well-known that file-sharing applications often scan a very large range of network port numbers.

As previously stated, not all cache-misses are the same, since an important percentage of entries are then used only for one packet, raising the question of what kind traffic/application is generating such kind of cache-miss. We answer this question by examining only cache-miss packets that have as consequence the insertion in the cache of an entry that is used only once. The obtained result is shown in Figure 4.19. Not surprisingly this time UDP/6881 and UDP/4672 are generating a large number of cache-misses. These are well-known default port numbers of BitTorrent and eDonkey applications, respectively. Another interesting observation is the fact that apparently there are quite a number of DNS requests (UDP/53) that remain unanswered.

Until now, we thoroughly analyzed the implication of deploying LISP Cache in today's Internet. The results proposed that it is possible to maintain a high hit ratio with relatively small cache sizes, showing that there is no use in large timeouts. It is shown that how the LISP Cache has good scalability properties when the size of the edge network varies.

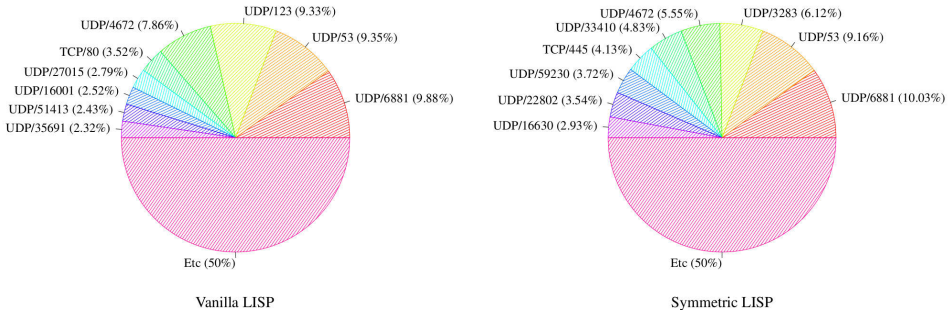


Figure 4.19: Breakdown by port number of cache-miss packets for entries forwarding one single packet (Mar. 2010).

Regarding cache-misses, we found that more than 70 % of the cache-misses are generated by UDP. While we do not expect that a cache-miss effects on UDP-based applications strongly, it may significantly degrade the performance of TCP-based applications due to the retransmission mechanism of TCP. In order to better understand the impact of a cache-miss, the following section will closely investigate how a cache-miss affects the TCP communication.

4.2 Impact of a LISP Cache-miss on a TCP Connection

As we observed from the implication analysis of the LISP Cache, it is unavoidable, in case of cache-miss, that the initial packet is delayed or retransmitted from the source end-host, causing a performance degradation of Internet services. To explore that further, in this section, we focus on the impact of the initial cache-miss from the perspective of an end-to-end network communication.

We focus our analysis only on TCP communication since it is obvious that a cache-miss has a major effect on TCP's retransmission mechanism. For this purpose, we deploy a LISP testbed in order to carry out the experiment and to see how LISP affects the performance of TCP communication.

4.2.1 Experiment Setup

In order to perform our experiment in a real LISP environment, we created a testbed as illustrated in Figure 4.20. FreeBSD 8.0 is used as the operating system of all machines in the testbed. Vanilla FreeBSD is installed in both source and destination end-hosts, while OpenLISP [142] enhanced FreeBSD is installed on ITR and ETR. Furthermore, for ITR and ETR, we write software that emulates the mapping system with tunable query/reply latency. This feature allows us to understand the impact

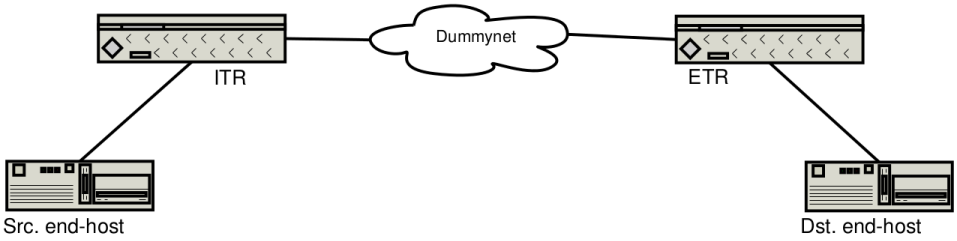


Figure 4.20: LISP testbed

of the mapping latency. We generate TCP traffic using iPerf and produce the propagation delay using Dummynet. In our experiments, we use the default value of FreeBSD 8.0 for TCP parameters (see Table 4.3).

Parameter	Value
Initial congestion window	4 MSS
Delayed ACK time	100 ms
Retransmission TimeOut	3000 ms

Table 4.3: TCP parameters used in the testbed

The experiments are conducted in three different scenarios: the normal Internet technology and the two LISP policies (*i. e.*, vanilla LISP and symmetric LISP) described in the previous section. In each scenario, we carry out the experiments changing the mapping latency from 100 ms to 3000 ms, with 100 ms step. For each latency value, we repeat the experiment 100 times, while collecting traffic from both source and destination end-hosts.

4.2.2 TCP over LISP

In order to show what happens when the initial packet of a TCP flow causes a cache-miss, we illustrate the sequence number of the first 25 packets of source-to-destination flows for both vanilla LISP and symmetric LISP as well as for the normal Internet case (Figure 4.21). Each point of the figure represents an average over 100 experiments. We observe one SYN packet retransmission for vanilla LISP and two packet retransmissions for symmetric LISP. More precisely, the first packet for vanilla LISP is dropped in the border router of the source site, as the ITR does not have the EID-to-RLOC mapping in its local cache. The same occurs for symmetric LISP, however, in this case the second packet is again dropped (this time at the destination) as the ETR also needs a mapping. Since the current LISP implementations

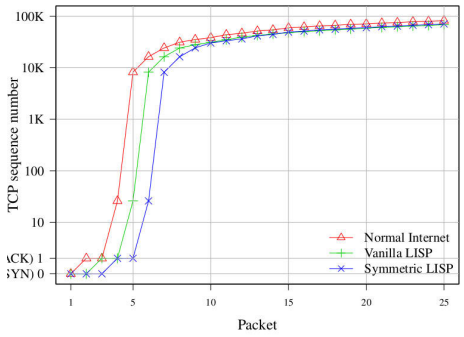


Figure 4.21: Sequence numbers

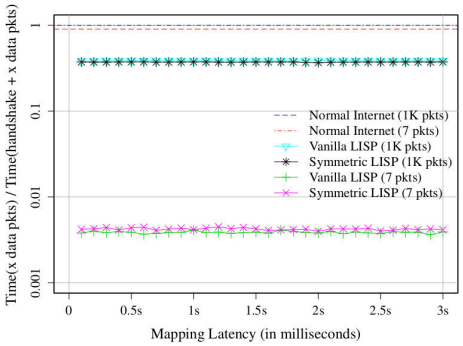


Figure 4.22: Performance comparison

drop packets that cause cache misses, the latency delays due to cache-misses can be estimated by multiplying the number of cache-misses by the TCP Retransmission TimeOut (RTO).

To evaluate the performance of TCP flows we calculate the ratio between the data transfer time and the total connection time (including the handshake). To see the performance difference between short flows and long flows, we performed measurements for flows transferring only 7 packets and for flows transferring 1000 packets (as defined in [38]). Results are presented in Figure 4.22, where we can see that the ratio is close to 1 for both short and long flows in the normal case without losses, while in the case of LISP, it shows a significantly lower ratio. Further, the difference between long flows and short flows is much larger in the context of LISP than in the normal Internet. Given the fact that short flows contribute more than 90 % of all global TCP flows [9], on this observation, it is important to understand the extent of the cache-miss impact.

Keeping these observations in mind, in the next analysis, we move back to the real traffic oriented emulation to quantify cache-misses during a failure/recovery event of an Ingress Tunnel Router (ITR) using traffic collected from two distinct stub networks (one with 20,000 end-users and the other one with 9,000 end-users).

4.3 A Local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers

In the first analysis (Section 4.1) of this chapter, we quantified the cache size and cache-misses based on the assumption of a flawless network environment. Moreover,

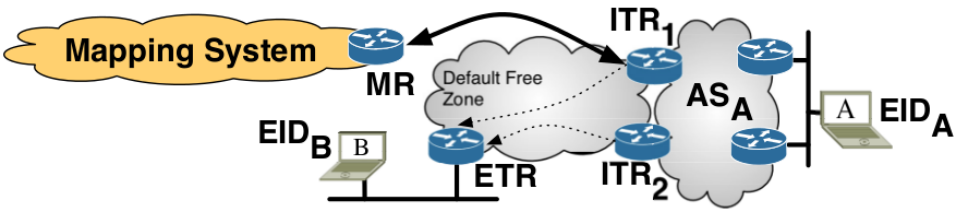


Figure 4.23: A multihomed Internet topology assumed for this section.

in the second analysis (Section 4.2), we zoomed in on the process of an initial cache-miss, and observed how a cache-miss degrades the performance of a single TCP connection.

As a last analysis on LISP Cache, we carry out an emulation (based on the emulator described in Section 4.1 and a flow-level LISP Cache emulator used in the work of IANNONE *et al.* [46]) in order to see how large the number of cache-misses can grow when an event such as the router failure or recovery occurs in multihomed stub networks.

Like the current Internet, LISP is affected by temporary link or node failures that may disrupt end-to-end reachability. More specifically, when an encapsulating router – the so called Ingress Tunnel Router (ITR) – fails, alternate/backup ITRs are not readily able to take over traffic encapsulation because they do not have the necessary EID-to-RLOC mappings, resulting in prolonged packet drops and traffic disruption (see Section 4.2). However, the reaction of LISP to link and node failures has not been studied by the research community. The only document briefly discussing reachability issues is a draft by Meyer *et al.* [63], which focuses on the general implications of having an Internet architecture based on the Locator/ID separation paradigm. To the best of our knowledge, the work presented hereafter is the first analysis of failures in LISP, and, from a more general perspective, the failure of encapsulating routers in the context of a Locator/ID separated architecture.

4.3.1 Router Failure in a LISP-enabled Network

In a normal IP network, such as the enterprise network shown in the right part of Figure 4.23, the Interior Gateway Protocol (IGP) handles link and node failures. Consider for example that ITR_1 and ITR_2 advertise a default route via IGP, a very common deployment. In this case, if (the link attached to) ITR_1 fails, then the other local routers will detect the failure and update their routing tables to forward the packets to ITR_2 so that they can reach the Internet. During the last years, various techniques have been developed to enable routers to quickly react to such failures [115]. With LISP, the situation is different. If ITR_1 fails, IGP will quickly redirect the packets to ITR_2. However, ITR_2 will be able to forward these packets

only if it already knows the corresponding mappings. Otherwise, ITR_2 will have to drop packets while querying the mapping system, an operation that can take tens of milliseconds or more per mapping [51]. Note that a Map-Resolver (MR [33]) in the figure is the entry point of the Mapping System which receives Map-Request message from ITRs and returns Map-Reply message with the requested mapping. However, the solution proposed in this section does not rely on any mapping system, hence their description is out of scope. A comparison of different mapping systems can be found in the work of Jakab *et al.* [51].

4.3.2 The ITR Failure Problem

In order to estimate the importance of the ITRs' failure problem, the first step is to evaluate the level of redundancy present in enterprise and campus networks. Indeed, in today's Internet, these networks often contain several redundant border routers to preserve connectivity in case of failures.

To evaluate the redundancy level of border routers, we analyze Internet topology information from the Archipelago project [17] and BGP tables of the RouteViews Project [136]. We first extract the stub prefixes from the BGP table of Oregon-IX collected on August 12^{th} 2010 [136]. The BGP curve in Figure 4.24 shows the number of neighbor ASes for each prefix. This is an approximation of the multihoming degree. We obtain the traceroute curve in Figure 4.24 by filtering an Archipelago trace captured July 24^{th} 2010 [17]. Archipelago traces are a collection of traceroutes performed from several vantage points to any possible /24 prefix. We consider that the last router that does not belong to the stub BGP prefix is a border router for that prefix, providing an approximation of the number of active routers of the multihomed stub prefixes.

The curves in Figure 4.24 show that when a stub prefix is multihomed, most of the time it has only two border routers. Nevertheless, the long tail of the distribution indicates that for some prefixes, the number of ITRs can be potentially high. For this reason, to assess to what extent ITRs' failures are a problem, we can analyze the impact on traffic for the simple scenario of 2 ITRs, by using trace-driven emulations and the topology presented in Figure 4.23. Our emulations use the software and the methodology used in previous works ([58], [46], and Section 4.1), assuming a mapping granularity equivalent to the current BGP table [46]. With this granularity, results from the previous section (Section 4.1), have shown that cache size is small, allowing us to neglect cache overflow problems.

We use two different 24h-long traces collected in 2010. A first trace, collected in September 1^{st} with a NetFlow collector without sampling, is from a medium size Campus network (∼9,000 active users), connected via a 1 Gbps link to its ISP and 122.35 Mbps average traffic. The second trace has been collected from our ISP vantage point (∼20,000 active DSL customers). We split the network of the captured

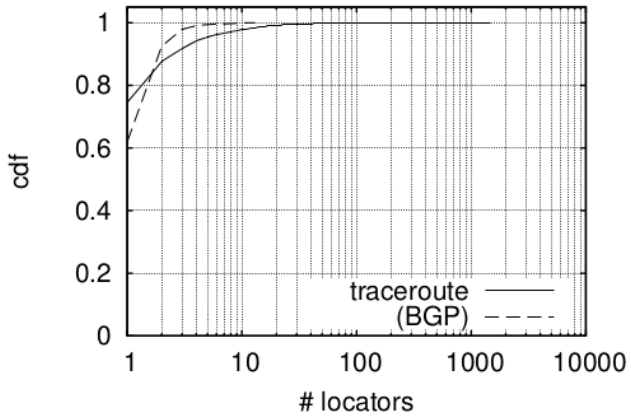
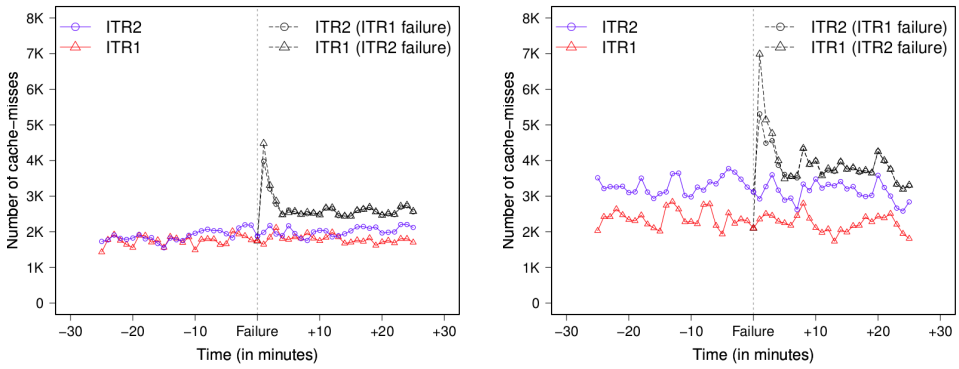


Figure 4.24: Distribution of the number of border routers per BGP prefix.



(a) Campus

(b) ISP

Figure 4.25: Cache-misses per-minute due to ITRs failures.

traces into two subnetworks (served by ITR_1 and ITR_2) in order to implement the multihoming scenario as in Figure 4.23. Since the ISP's topology was unknown, we attach half of the address space to ITR_1 and the other half to ITR_2. On the contrary, the Campus network has two border routers, which we assigned the role of ITR_1 and ITR_2, hence attaching the traffic to the ITRs according to the real topology. From the traces, we selected the busiest hour (*i. e.*, the worst case) and emulated the failure of one ITR in the middle of the selected hour.

Figure 4.28 shows the impact of the failure on the number of cache-misses on the alternate ITR, for the ISP and the Campus networks, which have a similar behavior. The figure shows the normal behavior without any failure (solid lines), as well as the two cases where one ITR fails (dashed lines). During the first minute after the failure there are up to 5,000 additional cache-misses per minute (more than three

times the normal rate) due to the traffic redirection on the alternate ITR. This failure can affect established TCP flows and cause packet drops for seconds or more. Figure 4.25 shows as well that the transient state lasts around 5 minutes.

We point out that we are underestimating the impact since we do not consider mapping delay, *i. e.*, we assume that the mapping is retrieved immediately after the cache-miss. In reality, failure-induced cache-misses have a much more severe impact because they affect established high traffic volume flows. While a normal cache-miss causes the loss of a few packets at flow setup time [84], a failure-induced cache-miss can cause more losses because disrupting high throughput flows on high bandwidth links. Moreover, the peak of cache-miss causes a load peak on the mapping system to retrieve the missing mappings.

4.3.3 Local ITR Failure Protection

To solve the problem presented in the previous section, we propose to synchronize the caches of a group of ITRs at the same site. The set of ITRs belonging to the same group is called the *Synchronization Set*. Replicating the LISP cache on the ITRs ensures that in case of failure and traffic re-routing, the packets of an existing flow will never be dropped because of a cache-miss, whatever the alternate ITR of the set is used.

There are two ways to synchronize ITR caches: either the mappings are *pushed* to the ITRs of the Synchronization Set, or the latter are *notified* that a new mapping has to be retrieved. We discuss both approaches further in Section 4.3.7. Here we just assume that the mapping requested by one ITR of a Synchronization Set is immediately replicated on all ITRs of the set. To ensure that mapping caches remain synchronized, the ITRs should keep it for the same amount of time. This implies that synchronized ITRs can no longer use a simple inactivity timeout [58] to purge unused entries from their cache. Indeed, doing so would lead to a loss of synchronization, since the same mapping can expire on one ITR, because it has not been used, while it is kept on the other ITR that forwarded packets towards this EID prefix.

To avoid such loss of synchronization, we propose to use a different policy, namely to keep each mapping in the cache during the TTL (Time-To-Live) that LISP associates to this mapping. When the mapping TTL expires, the ITR must check the entry usage during the last minute. If the entry has not been used, it is purged. Otherwise, the mapping entry is renewed by sending a Map-Request. This last action triggers the synchronization mechanism, again replicating the mapping to all ITRs of the Synchronization Set. Such an approach guarantees that, if no ITR has used the mapping during the last minute before TTL expiration, all replicas on the different ITRs will be purged. Otherwise, if at least one ITR used the mapping, the mapping

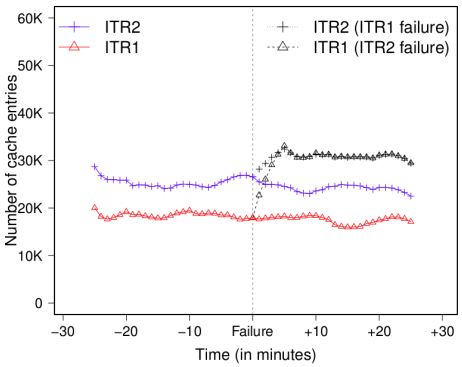


Figure 4.26: Cache size without synchronization (ISP).

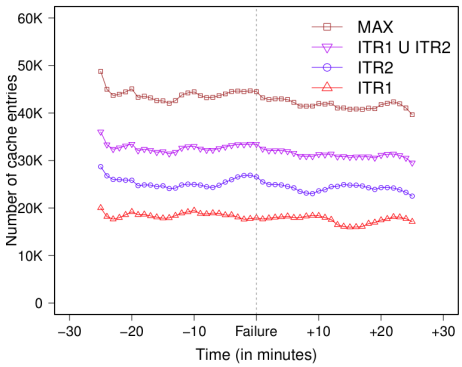


Figure 4.27: Cache size with synchronization (ISP).

will be replicated on all ITRs, renewing it for another TTL time. In both cases there is a consistent state on all ITRs of the Synchronization Set.

4.3.4 ITR Failure Protection Evaluation

To evaluate the cache synchronization technique we follow the same methodology and use the same traces as in Section 4.3.2. On the one hand, we emulate the asynchronous cache strategy, *i. e.*, where each ITR manages its own cache independently. On the other hand, we emulate the synchronous cache strategy, *i. e.*, where ITR_1 and ITR_2 belong to the same Synchronization Set. In all of our emulations, we set the TTL to 5 minutes, which we consider as a reasonable worst case. Indeed, a lower value would generate too much overhead [58]. In practice, it is likely that the TTL will be larger than 5 minutes (the default in current LISP implementations being 24h), reducing the number of cache-misses, but increasing the number of entries that are stored in the cache [46]. However, it is worth noticing that the cost of slightly increasing the cache is very small compared to the cost of adding a redundant link. The figures in the remainder of this section show the cache behavior of ITRs in the normal case (*i. e.*, without failures) using solid lines, while dashed lines correspond to scenarios with failures.

In Figure 4.25, the observed peak of cache-misses indicates that the traffic behind the two ITRs is not identical and not balanced; hence some mappings are in one ITR but not in the other. This is confirmed by Figure 4.26, showing the evolution of the cache size for the ISP network (Campus network exhibits the same behavior), reinforcing the motivation for cache synchronization. Indeed, it can be observed that the ITRs have in general different cache sizes and when one of the ITRs fails the diverted traffic makes the size of the remaining ITR grow.

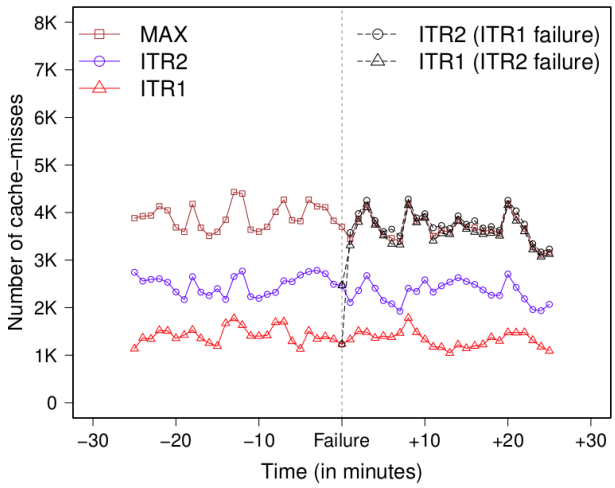


Figure 4.28: Cache-misses with synchronization (ISP).

Figures 4.27 and 4.28 show the evolution of both cache size and cache-misses when our synchronization approach is used. The curves labeled ITR1 and ITR2 show the evolution on the ITRs when they are not synchronized and none fails. The curve labeled MAX shows the maximum obtained when the ITRs are synchronized but the content of their cache is assumed to be completely disjoint (*i. e.*, the worst case). Finally, the curve labeled ITR1 (ITR2 failure) (resp. ITR2 (ITR1 failure)) corresponds to the actual values obtained with our synchronization approach if ITR_2 (resp. ITR_1) fails. Figure 4.28 is interesting for two reasons. First, no peak is observed when the ITR fails, rather, the miss rate corresponds to the miss rate that would have been observed if the network only had one ITR. As one would expect, the miss rate measured in steady state after the failure is identical to the miss rate observed in Figure 4.25 once the steady state has been reached. Second, comparing Figure 4.25 and Figure 4.28, it turns out that the miss rate when no failure occurs is lower when the ITRs are synchronized than when they are not. This difference can be explained by the fact that some form of locality occurs between the traffic of the two ITRs.

In summary, our emulations on the 2 ITRs scenario clearly show that ITRs' cache synchronization brings two main advantages: (*i*) it avoids a miss storm (hence induced packet drops) upon ITR failure; (*ii*) reduces the number of misses (hence packet drops) in the normal case. These benefits come at a small cost of increased cache size. Indeed, Figure 4.27 confirms that the ITRs have naturally some entries in common, which makes the burden of synchronization acceptable.

4.3.5 ITR Recovery: Problem, Protection, and Evaluation

The cache-miss storm in case of failure, as investigated so far, can also be observed when an ITR boots (or comes back online after failure). Indeed, in this case, its cache is empty, hence the traffic attracted for encapsulation will cause misses and will be dropped. Our synchronization mechanism can also be used in this scenario. In case of synchronization, the time needed for an ITR to synchronize with the other ITRs of the Synchronization Set is at most equal to the TTL. In this situation, when the ITR starts, it can begin the synchronization process and wait for a time equivalent to the TTL before announcing itself as an ITR able to encapsulate packets. With this approach the miss rate in the case of the router recovery does not differ from that of the normal ITR operation.

This naive approach is very simple but has a major drawback. The TTL can be set to a high value, refraining the ITR from being used for a long period of time. To overcome this issue, we suggest allowing the ITR receive a copy of the cache from another ITR in the Synchronization Set. The transfer of the cache's information must be done reliably, *e. g.*, using TCP. In this way the ITR can announce itself immediately after the cache transfer, shortening the start up time to a minimum, while preserving the benefits of the synchronization approach.

To better understand the impact on the traffic in the recovery case, we perform emulations in the two ITRs scenario, with ITR_2 recovering after ITR_1 runs alone for the last 30 minutes (using the ISP trace). Once back online, ITR_2 starts synchronizing with ITR_1, *i. e.*, it receives mapping information, but all traffic is still routed toward ITR_1. After 5 minutes (*i. e.*, the TTL value we are using throughout all emulations), when the cache is synchronized with ITR_1, the original setup is restored, sending traffic again to ITR_2.

Figure 4.29 shows the miss rate observed at the ITRs for both the synchronized and the non-synchronized cases. The curve ITR1 (ITR2 failure) gives the miss rate observed on ITR_1 before the recovery of ITR_2, while ITR1 (sync) (resp. ITR2 (sync)) shows the miss rate as observed on ITR_1 (resp. ITR_2) after ITR_2 has recovered when synchronization is used. For comparison, the curve ITR1 (async) (resp. ITR2 (async)) shows the miss rate when no synchronization is used, with a peak of 10,000 cache-misses confirming that waiting TTL instants to lazily synchronize the cache avoids cache-miss storm. For completeness, Figure 4.30 shows the evolution of the cache size before and after recovery. In all cases, the cache moves smoothly towards its new steady state TTL time after the recovery. This smooth convergence motivates the use a fast cache synchronization method (*i. e.*, explicit cache copy request) to speed-up recovery of an ITR.

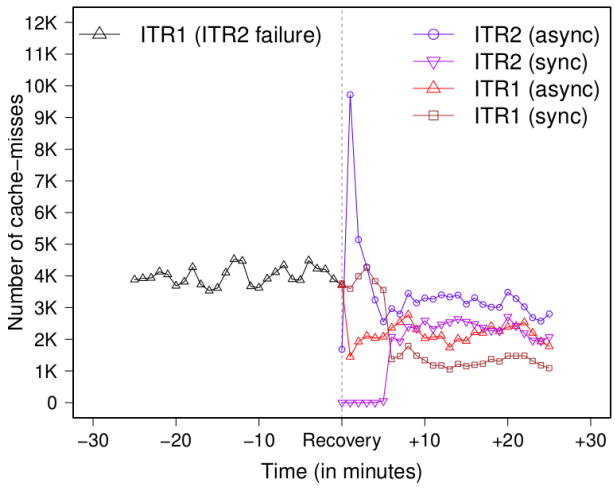


Figure 4.29: Cache-miss on ITR_2 after boot.

4.3.6 Synchronization Set in Large-Scale Networks

As shown in Section 4.3.2, in case of failure, when no synchronization is used, both the cache size and the cache-misses increase, as summarized in Table 4.4. While relatively similar increases can be observed comparing the results of the Campus and the ISP traces, the difference in the increases between ITR_1 and ITR_2 may be significant. Indeed, the average miss rate can be as low as 28 % when ITR_1 fails, but as high as 96 %, for the same trace, when ITR_2 fails. This result suggests that the Synchronization Set is critical and should be carefully computed. The issue is exacerbated in large networks containing potentially many ITRs, which cannot all be synchronized.

For the above-mentioned reasons, it is important to have an idea of how large the Synchronization Set can be. To this end, we exhaustively calculated the Synchronization Set of 8 different real large-scale network topologies. Each of these topologies belong to one of the three following categories: (i) *TIER1* grouping Tier 1 ISPs; (ii) *ISP* grouping national ISPs; (iii) *Research* grouping university campuses or research networks. A summary of these topologies is shown in Table 4.5. We model the networks as a directed acyclic graph where the nodes are the routers and the edges are the links. Then we classify the routers into three different types: the *ITRs*, the *EID routers*, and the *backbone routers*. For the topologies where we had enough information, we considered the routers connected to another ISP as ITRs; the routers connected to the customers or to LANs with end-hosts are considered the EID routers; all the other routers are classified as backbone routers. For the topologies where insufficient information was available, we classified the routers based on their

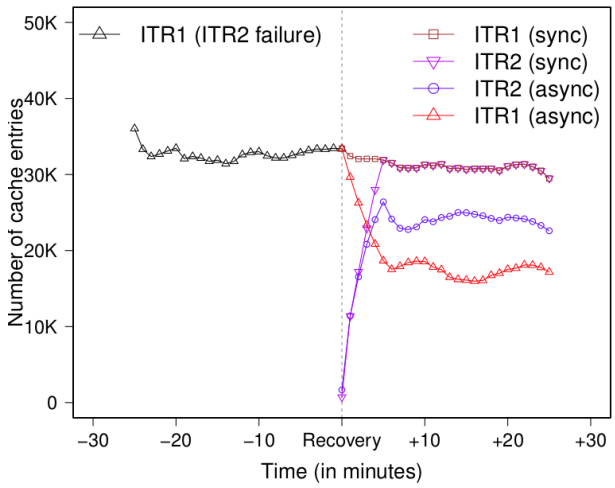


Figure 4.30: Cache size on ITR_2 after boot.

connectivity degree. All routers with a connectivity degree larger or equal to the 90^{th} percentile of the connectivity degree are considered ITRs. Routers with a connectivity degree less or equal to the 80^{th} percentile of the connectivity degree are EID routers. All the routers are classified as backbone routers. Applying this heuristic provides results similar to the topologies where information was available. To construct the Synchronization Set, we exhaustively enumerate the topology changes due to any possible single failure (*i. e.*, router or link).

Figure 4.31 gives the quartiles (including min., max., and median) computed for the number of ITRs potentially used by each EID router. For this evaluation, we consider that only one EID router serves an EID prefix. This is a reasonable assumption, since for the transit networks we evaluated (*i. e.*, TIER1a, TIER1b, NREN, Internet2, and Géant) the EID prefixes belong to the customers, which have only one router toward their ISP (but are multihomed with several ISPs). The figure shows that the number of ITRs potentially used by the EID routers is relatively independent of the type of topology and lies between 1 and 2. Meaning that only a small portion of the ITRs serve the same EIDs, due to the fact that large networks are segmented at relatively independent points of presence. This result clearly shows that synchronizing all the ITRs of a large network would be inefficient, while in our case the burden will be reasonable as an ITR will synchronize only with few other ITRs.

Figure 4.32 shows the number of routers that are potentially behind a single ITR, using again quartiles. The figure shows that in general an ITR has about half a dozen of routers behind it, going up to few tens for large networks. Therefore, the failure of an ITR can potentially impact an important portion of traffic, which has

	Failure	Increase	Increase*	Network
Entries (avg.)	ITR1	55.54%	46.59%	Campus
	ITR1	20.57%	20.30%	ISP
	ITR2	68.54%	56.98%	Campus
	ITR2	63.93%	56.54%	ISP
Misses (avg.)	ITR1	40.32%	55.25%	Campus
	ITR1	17.02%	28.22%	ISP
	ITR2	52.46%	72.95%	Campus
	ITR2	67.14%	96.34%	ISP

* Increase without counting the 5 minutes transient period right after failure

Table 4.4: Increase of cache size and misses due to ITR failure.

to be shifted to the other ITRs. If the caches are not synchronized, the ITRs to which the traffic is diverted might experience a very large miss rate, with many packet drops. Synchronizing the ITRs' caches avoids such drops, as we demonstrate in Section 4.3.4.

4.3.7 Synchronization Techniques

Even if, as discussed above, the Synchronization Set is in practice also small for large-scale networks, it is important to implement a mechanism to achieve such synchronization without introducing excessive overhead or management complexity. There are two possible ways to synchronize ITRs: either, the mappings are *pushed* to the ITRs, or the ITRs are *notified* of the presence of a new mapping and they retrieve the mapping on their own.

In both cases, the synchronization can be implemented either by leveraging on a routing protocol extension, by using a specific existing protocol (*e. g.*, [65], [100], [82], [88]), or by creating a brand new protocol. Extending a routing protocol to synchronize ITRs implies that they must be in the same routing protocol instance (*e. g.*, the same IGP). However, this assumption is too restrictive as there might exist cases where the ITRs are in different networks (*e. g.*, the ITRs are operated by the ISPs and the LISP site is multi-homed). Hence, we do not consider the extension of the routing protocol as an acceptable solution. Fortunately, LISP already proposes features that can be used to implement a synchronization mechanism. Indeed, LISP

Name	Routers	Links
TIER1a	500^+	$2{,}000^+$
TIER1b	200^+	800^+
ISPa	50^+	200^+
ISPb	20^+	60^+
UCL	11	41
NREN	30^+	70^+
Internet2	11	30
Géant	22	72

Table 4.5: Topologies characteristics.

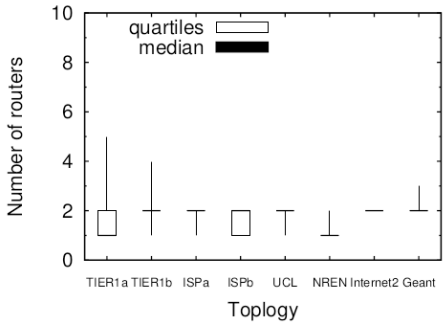


Figure 4.31: Number of used ITRs per EID routers.

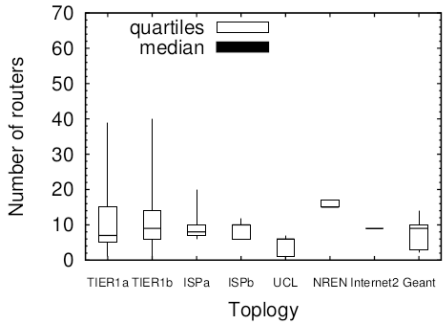


Figure 4.32: Number of routers behind an ITR.

specifies the *Included Map-Reply* [26] feature to push mappings in ITR caches. An included Map-Reply is a special Map-Request, which piggybacks a mapping. LISP also specifies the *Solicit Map-Request* (SMR) bit in Map-Requests to force ITRs to refresh their cache [26]. When an ITR receives a Map-Request with the SMR bit set, it sends a Map-Request to retrieve a mapping for the EID indicated in the SMRed Map-Request.

Since both ITRs and MRs are involved in the mapping resolution, these are good candidates to trigger cache synchronization. However, ITRs are data-plane devices that need to forward packets at line rate. Therefore, imposing them to actively manage the synchronization protocol might cause excessive overhead with consequences on the data-plane performance. On the contrary, Map Resolvers are purely control-plane devices that are not intended to forward packet at line rate. Hence, MRs look like the best candidates to manage the synchronization protocol. To implement the cache synchronization based on notification messages, the MR only has to send an

SMRed Map-Request to all the ITRs, when it receives a Map-Request. However, if the mappings are pushed to the caches, then the MR has to proxy the Map-Requests. The MR performs two operation when it receives a Map-Reply. First, it forwards the Map-Reply to the ITR that requested the mapping. Second, it sends the mapping to the other ITRs by using an included Map-Reply.

All through this chapter, we thoroughly analyzed LISP from multiple perspectives. Despite its overhead due to the security concern and caching mappings, outcomes of our study proved that LISP is a suitable solution to cope with the scalability problem of the core Internet that is largely influenced by multihoming (upstream link diversity).

4.4 Related Work

The concept of Locator/ID Split was already discussed, but never widely explored, in the late 90s ([101], [40]); rather it has been used only to solve specific issues, like for instance cryptographic security with HIP (Host Identity Protocol [76]) or multihoming for IPv6 with Shim6 [80]. The situation has changed after the rechartering of the RRG, with quite a number of proposals being discussed, either based on tunneling (*e. g.*, LISP [26], HAIR [29]) or on address rewriting (*e. g.*, ILNP [8], Six/One [116]). Despite the plethora of proposals, very few works have tackled the evaluation of such a critical component as the cache.

Kim *et al.* [57] study route caching based on Netflow data, showing how caching is feasible also for large ISPs. However, they use a generic approach, not focusing on LISP, and a limited cache size. Iannone *et al.* [46] perform an initial study, without considering security issues, based on a single Netflow trace of a small/medium sized university campus. We adopt a similar methodology to provide comparable results, which we discuss in Section 4.5. Nevertheless, because we base our results on three 24-hour packet level traces captured in different periods, we are able to provide a deeper analysis, including security aspects.

In the work of Zhang *et al.* [147], the authors propose a simple LISP Cache model with bounded size, using a Least Recently Used (LRU) policy to replace stale entries. Their evaluation is based on a 24-hour trace of outbound traffic of one link of the CERNET backbone, the China Education and Research Network. The analysis made in the first part of this chapter, a different approach is taken, assuming that the cache is not limited in size. This is a reasonable assumption since, as we will show later, even for large ISPs there is no need to use large caches and by tuning the expiration timer, caches do not grow so large as to hit memory limits of current routers.

Jakab *et al.* [51] propose a LISP simulator able to emulate the behavior of different mapping systems. However, they focus on the evaluation of the latency of their own

mapping system solution, namely LISP-TREE, and compare it to other approaches but neglect the analysis of the LISP Cache.

4.5 Summary

The awareness of the fact that the current Internet architecture suffers from some scalability issues has triggered an important body of research focused on the Locator/ID Split paradigm. LISP, in particular, has been adopted by the IETF and looks like the most promising solution for wide adoption. In this chapter, we carry out several case studies to understand the implication of deploying LISP in today's Internet.

First, by analyzing a LISP Cache based on the two 24-hour traffic traces, we find that the number of cache entries can be kept low (about 10K entries with 60 seconds as the timeout value) while ensuring high cache hit-ratio (higher than 99 %).

Thus, given that the number of BGP entries reached 400K during the period of our evaluation, and that the BGP entries in a LISP-enabled Internet decreases to about 60 % of the current numbers [18], and that its increase rate is significantly suppressed, LISP seems to be a good solution to tackle the scalability problem of the core Internet.

The study also shows that the cost of improving the security level of LISP is reasonable. The result shows that overhead for applying the security-related policy to the original version of LISP is an order of 20 % (of the traffic overhead introduced by the original LISP).

Second, our evaluation shows that around 30 % of cache-misses are generated by TCP applications. Therefore, we conduct experimental analysis to assess the impact of a LISP cache-miss event on retransmission mechanism of TCP communication. We observe that the LISP cache-miss causes a significant degradation of TCP performance due to the common packet-drop behavior of the current LISP implementations.

Third, we present a thorough study of failure protection and recovery in the context of the Locator/Identifier Separation Protocol (LISP). Protecting Ingress Tunnel Routers (ITRs) is important as the failure of one of them can impact a large portion of the network due to the cache-miss examined in the second analysis. Indeed, our analysis shows that an ITR failure (and also recovery) within a multihomed LISP network can cause large disruptive impact on ongoing traffic.

To this end, we propose a local failure protection mechanism for ITRs by synchronizing mapping information within a small-sized group of ITRs. Our emulation of this technique shows that the load increase due to the synchronization is acceptable and suppresses the loss of packets upon ITRs failure/recovery.

5

Multi-source Multipath HTTP (mHTTP)

In the previous chapters, we discuss only one type of network diversity at a time (*i. e.*, address space diversity in Chapter 3 and upstream link diversity in Chapter 4). Now, in this chapter, we address multiple types of diversity (*e. g.*, interface plurality/heterogeneity, path diversity, and data source diversity) to cope with the challenge of end-to-end performance. Considering that path diversity is mainly created by upstream link diversity, and that address space diversity has the potential to increase the level of path diversity (through dual-stack networks) and data source diversity (through mirrored servers in distinct address spaces), we utilize (or can utilize) almost all types of network diversity with the technology we propose in this chapter.

Our proposal is Multi-source Multipath HTTP (mHTTP) which is an easily deployable solution for increasing the performance of the content delivery in the context of Content Distribution Infrastructures (CDIs) and is designed to take the advantage of various types of network diversity in the Internet. As mHTTP relies on HTTP range requests, it is specific to HTTP which accounts for more than 60 % of the Internet traffic [67].

The reason that deployment of mHTTP is easy is its one-sided design principle. To be more precise, mHTTP needs only client-side but not server-side or network modifications as it is a solely receiver-oriented mechanism. Moreover, the modifications are restricted to the socket interface. Thus, no changes are required to the applications, the kernel of end-systems, or the network at the server side.

In this chapter, we first present the design and implementation of mHTTP, then evaluate its performance by conducting measurements over a testbed and in the wild.

5.1 Towards Utilizing Network Diversity

In today's Internet, one of the main detriments in user experience is completion times of data transfers that for large objects is limited by network capacity. However, recent developments have opened new opportunities for reducing end-to-end latencies. First, most end-user devices have multiple network interfaces (*i. e.*, interface diversity). Second, popular contents are often available at multiple locations in the network (*i. e.*, data source diversity). When combined, these provide substantial path diversity within the Internet that can be used by users to improve their quality of experience.

Previous work has taken partial advantage of this path diversity in the Internet. Multipath TCP (MPTCP) uses the path diversity available between a single server and a single client [31, 96]. Application specific download managers are other examples of related work that benefits from the path diversity between a single server and a single client [126,131]. Content Distribution Networks (CDNs) provide replication of content and smart matching of users to appropriate CDN server, *e. g.*, via PaDIS [90] or ALTO [107] services, which takes advantage of this replication. Moreover, there are application specific video streaming protocols that try to take advantage of the replication of streaming contents provided by CDNs [1, 54, 112]. BitTorrent is another sophisticated application which takes advantage of content replication among its users [19]. The drawback of each of the above approaches is that they do not utilize all of different types of path diversity in the Internet or if they do, they are application specific.

We propose Multi-source Multipath HTTP, mHTTP, which enables users to establish simultaneous connections with multiple servers to fetch a single content. mHTTP is designed to combine the advantage obtained from distributed network infrastructures provided by CDNs with the advantage of multiple interfaces at end-users. Unlike existing proposals: a) mHTTP is a purely receiver-oriented mechanism that requires no modification either at the server or at the network, b) the modifications are restricted to the socket interface; hence, no changes are needed to the applications or to the kernel, and c) it takes advantage of all existing types of path diversity in the Internet.

mHTTP is proposed for HTTP traffic, which accounts for more than 60 % of the total traffic in today's Internet [67]. As stated in Popa *et al.* [92], HTTP has become the *de facto* protocol for deploying new services and applications. This is due to the explosive growth of video traffic and HTTP infrastructure in the Internet in recent years. mHTTP is primarily designed to improve download times of large

file transfers (such as streaming contents). Measurements results have shown that connections with large file transfers are responsible for the bulk of the total volume of traffic in the Internet [68]. Furthermore, while mHTTP decreases download times for large objects by up to 50 %, it does no harm to small object downloads as shown in Section 5.6.

The key insight behind mHTTP is that HTTP allows chunking a file via byte range requests and that these chunks can be downloaded from different servers as long as these servers offer identical copies of the object[1]. mHTTP learns about the different servers that host the same content by either using multiple IP addresses returned by a regular DNS query, sending multiple queries to multiple DNS servers, or utilizing the eDNS feature [22]. It also works with a single server when multiple paths are available between the receiver and the server. Hence, mHTTP can also be used as an alternative to MPTCP for HTTP traffic.

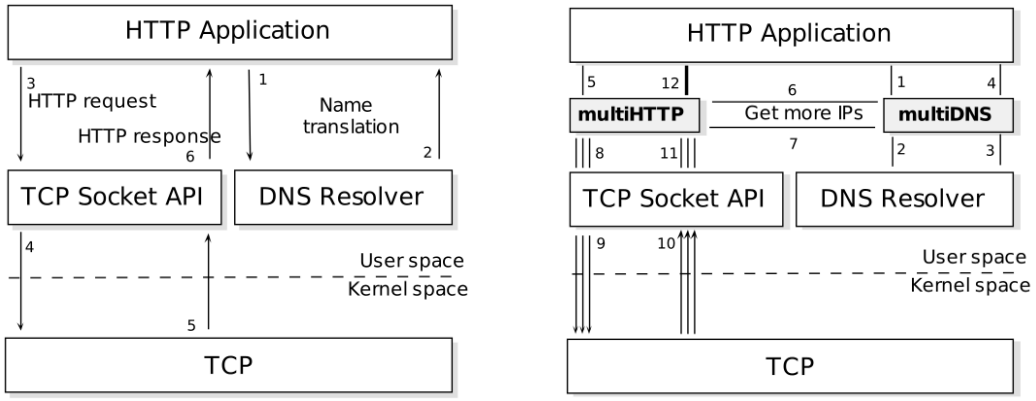
The mHTTP design consists of: (i) multiHTTP: a set of modified socket APIs which splits content into multiple chunks, requests each chunk via individual HTTP range requests from the available servers, reassembles the chunks and delivers the content to the application. (ii) multiDNS: a modified DNS resolver that obtains IP addresses for a server name by harvesting the DNS replies and/or by performing multiple lookups of the same server name by contacting different domain name servers.

Our key contribution is the concept of mHTTP along with a prototype implementation and evaluation. We evaluate the performance of mHTTP through measurements over a testbed and in the wild. We compare the performance of mHTTP with regular HTTP operating over single-path TCP and MPTCP. We observe that

- mHTTP indeed takes advantage of various types of path diversity in the Internet.
- For large object downloads, it decreases download times up to 50 % compared to the single-path HTTP transmission. Moreover, it does no harm to small object downloads.
- mHTTP performs similar to MPTCP while only requiring receiver-side modifications. As MPTCP requires changes to the kernel, both at the sender and receiver, we consider mHTTP to be a viable alternative when running HTTP.

The comparison with MPTCP is performed in a single-server scenario as MPTCP is restricted to the use of a single server. mHTTP, on the other hand, can be used both in single-server and in multi-server scenarios.

[1]The extra header added by the application might differ from server to server. The mHTTP parser, refer to Section 5.4, deals with the application headers.



(a) Regular HTTP/TCP architecture at a client.

(b) mHTTP architecture at a client.

Figure 5.1: Structural differences between HTTP/TCP and mHTTP

5.2 Multi-source Multipath HTTP

A Content Distribution Network (CDN) provides wide-spread replication of popular content at multiple locations in the Internet. Multi-source Multipath HTTP (mHTTP) is designed to combine the advantage obtained from such a diversity with the advantage of diverse network connectivity at end-users. In this subsection, we describe the high-level concept of mHTTP.

5.2.1 Regular HTTP over TCP

Before we discuss the design of mHTTP, we review various components of HTTP communication over TCP. As illustrated in Figure 5.1(a), when an HTTP application tries to download an object from a web server, it first requests to the local *DNS resolver* to translate the human-friendly URL to a set of IP addresses. And then it sends an HTTP request to establish a connection to one of these addresses. *TCP Socket API* is the interface to the underlying transport protocol (TCP). The *TCP stack* in the kernel ensures reliable data transmission and congestion control. Note that (i) one domain name may be associated with multiple IP addresses. Moreover, (ii) different DNS servers may return different IP addresses. This is often observed in server infrastructures which serve popular contents. (i) occurs when the load is spread across multiple servers [90]. (ii) occurs in Content Delivery Networks (CDNs). However, even though the content is in principle available at multiple locations, traditional HTTP/TCP cannot take advantage of this. To overcome this limitation, we propose a novel protocol, mHTTP, built on the regular HTTP-over-TCP architecture.

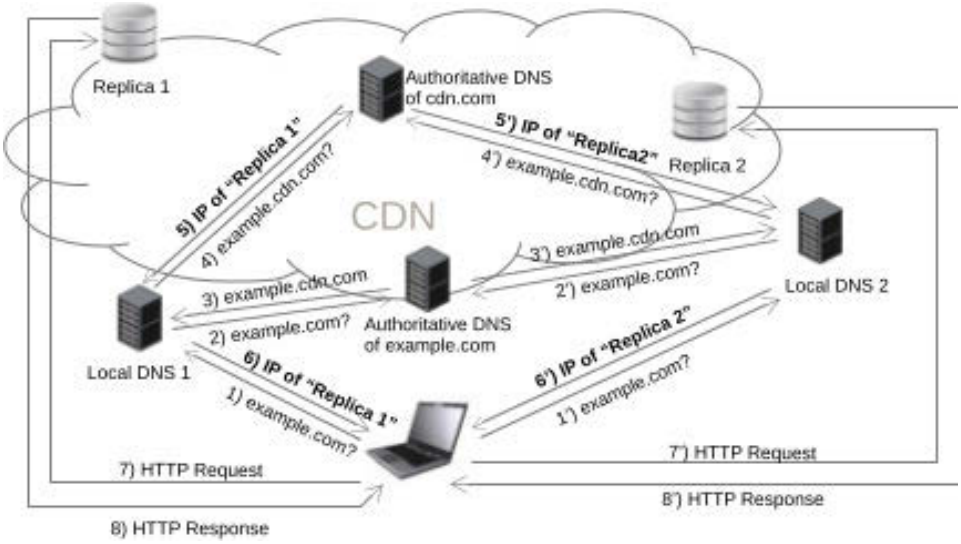


Figure 5.2: An example of the mHTTP operation in a CDN with two connections over two available interfaces. Replica 1 and Replica 2 store the identical copies of content of example.com.

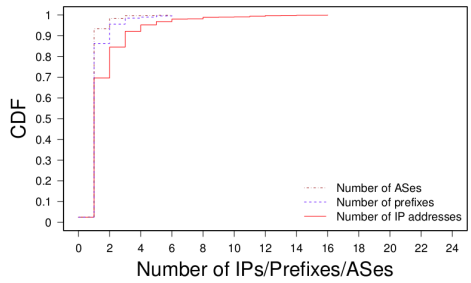
5.2.2 mHTTP

mHTTP is designed with the following three key features in mind:

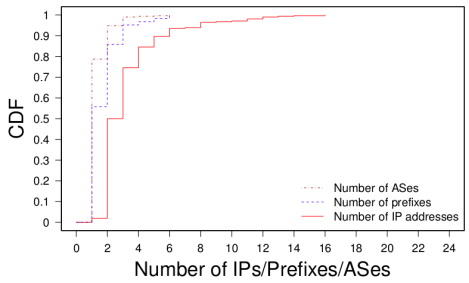
- mHTTP must take advantage of multiple built-in interfaces, multiple paths, and multiple data sources, by establishing simultaneous connections via multiple interfaces to multiple data servers where the identical content is stored.
- mHTTP must not make any modifications on the server-side infrastructure or the protocol stack.
- The client-side implementation must be transparent to the application, *i. e.*, modifications must be limited only to socket APIs.

The key idea of mHTTP is to use the HTTP range request feature to fetch different content chunks from different servers. We define a chunk as a block of content delivered within one HTTP response message. mHTTP includes two components, *multiHTTP* and *multiDNS* as shown in Figure 5.1(b). These components extend the functionality of HTTP and the DNS resolver. The main purpose of multiHTTP is to handle chunked data delivery between the application and multiple servers; and that of multiDNS is to collect IP addresses of available content sources. Figure 5.2 briefly illustrates the process for a 2-connection mHTTP session in a CDN.

multiHTTP is the core component of mHTTP. It is responsible for the management of data chunks; taking advantage of multiple content servers and therefore of path and network diversity; and scheduling chunk requests. multiHTTP intercepts all



(a) Top-1000 domain names



(b) Top-300 domain names

Figure 5.3: CDF of IP addresses, prefixes, and AS numbers for top-1000/top-300 domain names obtained from a single query to the local DNS.

messages sent from the application to the remote end-host (*e. g.*, server). When a TCP connection is identified as an HTTP connection, on reception of an HTTP request from the application, the multiHTTP module modifies the header of the HTTP request by adding a range field. The HTTP response header includes the file size. Thus mHTTP can issue multiple range request to multiple servers that serve the same object and their IP addresses are known via multiDNS.

If the connection is not an HTTP connection, mHTTP falls back to regular socket APIs. Also, for an HTTP connection, if the server does not reply with a partial data response to the HTTP request, or if multiDNS only returns a single IP address and the client is single homed mHTTP may still decide to fall back to regular HTTP.

multiDNS obtains different IP addresses by performing multiple lookups of the same server name by contacting different DNS servers (*i. e.*, the local DNS servers of the upstream ISPs for each of its interfaces, a Google DNS server, an OpenDNS server, etc.). Additionally, it can use the eDNS extension to uncover many more servers in a CDN infrastructure.

5.3 Implementation of multiDNS

In this subsection, we describe multiDNS the core element designed for discovering IP addresses of servers that hold replicas of a content. Note that mHTTP also works with a single server when multiple paths are available between the receiver and the server (refer to Section 5.6.3 for an example).

5.3.1 Data Source Diversity

Before we discuss details of the multiDNS implementation, we analyze how many IP addresses we receive from a single query for a hostname. We choose the top-1000

hostnames provided by *Alexa.com* and request the resolution of these hostnames by sending DNS queries to the local DNS server of a client residing in a university campus. As illustrated in Figure 5.3(a), even with a single query to the local DNS server, approximately 30 % of the total hostnames are associated with more than one IP address and respectively 10 % and 5 % of the total hostnames are in different network prefixes and reside in different ASes (Autonomous Systems). When performing two lookups, one query to the local DNS and another to the Google DNS, these numbers increase to 35 % (IP addresses), 17 % (prefixes), and 7 % (ASes). This can be seen as an evidence that CDNs can provide a different set of IP addresses to a user depending on the choice of a DNS server. More evidence of the content diversity can be found in the work of Poese *et al.* [91].

In Figure 5.3(b), we narrow down the scope to the top 300 hostnames and our result shows that almost all hostnames are associated with at least two IP addresses. Given the fact that the major fraction of the total traffic originates from a small number of popular content providers [4, 13] and the fact that the top 15 domains account for 43 % of the total HTTP traffic in a large European ISP [67], the fraction of the traffic contributed by providers through multiple servers should be significant.

5.3.2 Getting an IP address by a Hostname

When an application needs to obtain an IP address from a human-readable URL, it invokes name resolvers such as *gethostbyname()* or *getaddrinfo()*. The resolver, then, creates a request message and sends it to the local DNS server usually provided by the local ISP. Depending on the content sources, if content is only available at a single server, the DNS returns the IP address of that particular server so that the request can be routed to the server. However, if content is available at multiple places (*e. g.*, a server farm or CDNs), DNS returns a list of IP addresses. In the case of multiple IP addresses, a typical behavior of an application is to choose the first IP address in order to establish the connection and to discard the rest. multiDNS, however, keeps the rest of the IP addresses for later use.

5.3.3 Getting more IP addresses

As mentioned above, different DNS servers may provide different sets of IP addresses. Therefore, it is worthwhile querying multiple DNS servers in order to obtain more IP addresses. multiDNS plays the role of managing the identities of different resolver of different access networks, whenever an interface is activated and the IP address is assigned. It also handles the DNS query by validating the availability of a local DNS server in each access network for each interface[2]. If the local DNS is still available at the point of a name translation, a query to that content is made to the local DNS

[2]The local DNS information is collected when a DHCP request is completed

of that particular access network. For each interface, multiDNS receives a list of IP addresses from each access network, and chooses desired number of IP addresses from every list. Hence, if the desired contents are available at CDNs, mHTTP does not only retrieve them from the CDNs accessible to the public, but from the CDN nodes in the CDN server farm known to the local DNS resolvers.

5.4 Implementation of multiHTTP

The main task of multiHTTP is to interpret mHTTP for regular HTTP speakers such as web servers and client-side applications.

5.4.1 HTTP Byte Range Request

RFC2616 [30] specifies the use of a *byte range request* which enables the partial delivery of content. A client initiates such a request by adding a *range* field within the header of an HTTP request message including offsets of the first byte and the last byte of the partial content. If the server supports this operation, it replies with 206 as the status code (on acceptance of the request message) followed by sequences of bytes. Otherwise, the server replies with a different status code (*e. g.*, 200 OK on success). Note that a block of partial content is referred to as a chunk in this paper. Although [30] defines this operation as an optional feature, our tests on well-known web servers during the development of mHTTP show that almost all web servers accept *range* requests.

Partial content delivery is widely employed by HTTP-based downloaders in order to continuously resume fetching a transferred file. Another common usage of this feature is multi-threaded downloading implemented in some software, *i. e.*, [126] and [131]. Such software boosts download speed by fetching different parts of the content over different connections using the partial content delivery. At first sight, mHTTP is similar to those software; however mHTTP operates in the Socket API thus helps existing HTTP software to utilize the bandwidth more effectively. As mHTTP is designed to communicate with multiple servers containing identical copies of the content, it is clearly distinguished from multi-thread and downloader approaches.

5.4.2 mHTTP Buffer

multiHTTP initializes *mHTTP buffer* and creates a file descriptor associated with the buffer when *socket()* is called by the application. The buffer consists of a *queue* and a *pool* of content blocks (chunks). The queue is a large memory block that is continuously read by the application. Thus, the file descriptor plays the role of a communication channel between the application and the mHTTP buffer (Figure 5.4).

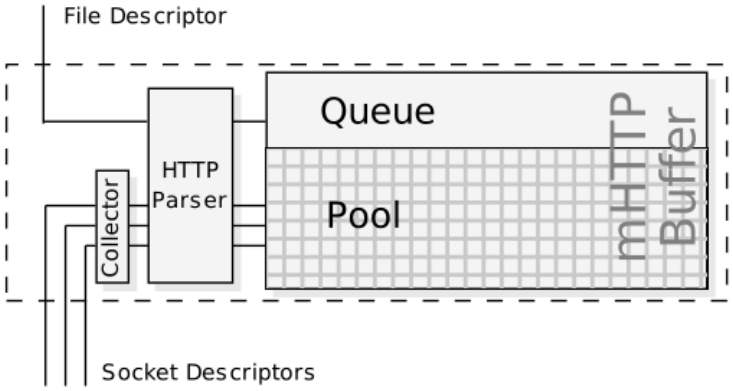


Figure 5.4: multiHTTP design: a) *mHTTP buffer* stores out-of-order received data b) *HTTP parser* examines and modifies HTTP headers c) *collector* gathers data from individual TCP connection buffers.

The pool maintains multiple content blocks in which chunked data collected from individual TCP buffers is stored. A content block can be indexed by the combination of the socket descriptor and the starting byte of the chunk. Data within content blocks is moved to the queue as soon as it is continuous from the last byte that is stored in the queue. The size of the queue does not grow greatly since it is continually drained by the application. However, the size of the pool needs to be sufficient to store out-of-order received chunks. We study the required size of the mHTTP buffer through measurements in Section 5.6. If mHTTP decides to fall back to the regular HTTP, or if the connection is not an HTTP connection, one of the socket descriptors replaces the file descriptor and the mHTTP buffer is discarded.

5.4.3 Manipulating HTTP Headers

Once the connection is identified as an HTTP connection, multiHTTP enables an HTTP parser, which examines HTTP messages during the content delivery period. The tasks of the HTTP parser are mainly three-fold:

- **HTTP request manipulation** The HTTP parser adds the *range* field to the end of the header with the specified chunk size when the initial HTTP request is sent by the application. When the response message to the initial request arrives back to the application, multiHTTP knows the size of the file and whether or not the server accepts a byte range request.
- **Parsing HTTP headers** The HTTP parser extracts and stores important information from the request and response headers such as availability of content, support for the partial content delivery, content size, and the byte range of the content block.

- **Response header management** In order to allow applications to use mHTTP without modification, the behavior of mHTTP must be the same as that of a regular HTTP communication from an application's point of view. To this end, the HTTP parser replaces the initial response header (206) with a header that indicates the acceptance of the request (200). All subsequent headers are discarded by the HTTP parser.

5.4.4 Connections

Upon confirmation of the complete delivery of the initial response message, multiHTTP establishes additional TCP connections using different IP addresses provided by multiDNS. In order to obtain another IP address, multiHTTP invokes *get_ip()* from multiDNS (see 6 and 7 in Figure 5.1). The mechanism used by multiDNS to select IP addresses is independent of the operation of multiHTTP. The current version of multiDNS hands IP addresses over to multiHTTP in the order that they are retrieved. Similarly, the number of connections to be used is configurable.

multiHTTP operates *collector*, a background process that collects data from individual TCP connection buffers. Each new connection must be attached to the *collector* as soon as it is successfully established. Likewise, a connection can be detached from the *collector*.

Determining what content chunk to request over each connection is another important task of multiHTTP. It keeps track of the requested chunks and decides which chunk to ask on the next request message after the previous chunk on the same connection is completely fetched. The mechanism used for such decision is further explained in the next section.

5.5 Scheduling

Different connections may exhibit different level of performance, in terms of latency, capacity, and loss rate. This may cause reordering of the chunks received at the mHTTP buffer. Figure 5.5 illustrates an example of such reordering. We have two connections: one slow and one fast (in terms of download time). The 4th chunk is downloaded over the slow connection and the 5th, 6th, 7th, and 8th chunks are downloaded over the fast connection. As the download of the 4th chunk is not finished yet, these later chunks cannot be moved to the queue. Hence, a mechanism that allocates chunks to different connections plays a critical role in multiHTTP. In this section, we explore mHTTP's design choice with regard to chunk *scheduling*.

For simplicity, we assume a client with two interfaces (*e. g.*, e_0 and e_1). Let $S_0 = \{s_{01}, s_{02}, \cdots, s_{0N1}\}$ and $S_1 = \{s_{11}, s_{12}, \cdots, s_{1N1}\}$ be respectively sets of servers available to the client through e_0 and e_1. Note that S_0 and S_1 are not necessarily disjoint

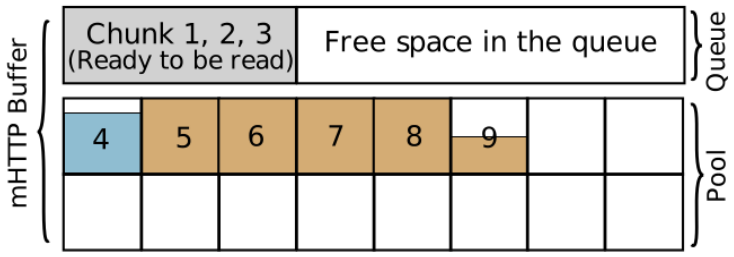


Figure 5.5: A bottleneck in mHTTP buffer. Chunk 4 is downloaded over a slow connection. Chunks 5, 6, 7 ,8 are downloaded over a fast connection. These chunks cannot proceed to the queue before chunk 4 is completely received. A scheduler is needed to better allocate chunks across different connections.

sets. N_1 and N_2 are the numbers of discovered servers over e_0 and e_1. Let P_0 and P_1 denote the connections established through e_0 and e_1 to servers in S_0 and S_1. In this paper, we limit the number of established connections over each interface to one and the total number of connections to two. However, our implementation can set up an arbitrary number of connections per interface.

We measure the instantaneous throughput of each connection by measuring the number of bytes received at by mHTTP every 20 ms. We use a moving average to estimate the average throughput of each connection:

$$\overline{THR}_{new} = 0.8 * THR + 0.2 * \overline{THR}_{old}$$

where THR is the instantaneous throughput measured every 20ms and $\overline{THR}$ is the estimated average throughput. We denote by $\overline{THR}_0$ and $\overline{THR}_1$ the estimated average throughput measured over connections P_0 and P_1.

Let L denote the size of the object and C the chunk size, both measured in bytes. We denote by $N = \lfloor L/C \rfloor + 1$ the number of chunks to be fetched by the client and by $1, 2, 3, \cdots, N$ the chunk numbers. Here, $\lfloor x \rfloor$ is the largest integer not greater than x. Also, let D be the set of chunks that have not yet been requested for download. D is a sorted set based on the chunk numbers.

The scheduling algorithm decides what chunk to request over each connection. For example, if a chunk is successfully fetched over connection P_0, the next chunk over this connection must be carefully chosen in order to avoid a bottleneck situation such as presented in Figure 5.5. mHTTP decides the next chunk over P_0 using the following mechanism:

1. calculate $T_0 = \left\lfloor \max\left(\overline{THR}_0, \overline{THR}_1\right) / \overline{THR}_0 \right\rfloor$;
2. ask for the T_0'th chunk from the set D. If $T_0 > |D|$, no chunk is requested. $|D|$ is the size of set D.

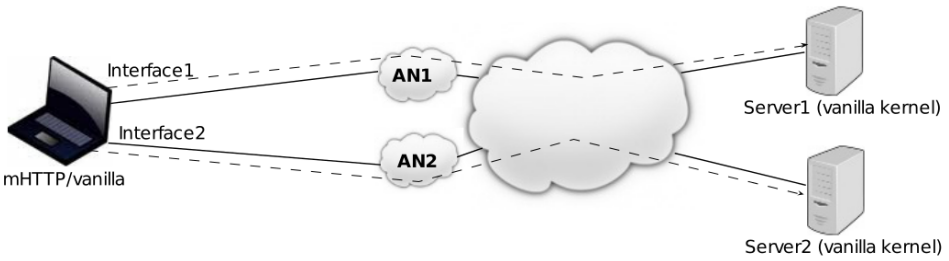


Figure 5.6: Scenario 1: 2 interfaces at the client; 2 servers; 2 paths (dashed lines). In our indoor testbed, AN1 and AN2 are Ethernet routers with a nominal rate of 100Mbps. In our outdoor testbed, the client is a mobile device (laptop) with one WiFi and one LTE interface.

3. Remove the requested chunk from the set D.

T_0 predicts the number of chunks that can be delivered over the best connection among P_0 and P_1 while one chunk is transmitted over P_0. If P_0 is the best connection, then $T_0 = 1$. When mHTTP needs to issue a new request over P_0, it does not request the next chunk but skips to the T_0'th chunk from the set D.

mHTTP uses similar mechanism to decide the next chunk to be requested over connection P_1, with the modification that T_0 is replaced with $T_0 = \lfloor \max(\overline{THR}_0, \overline{THR}_1)/\overline{THR}_1 \rfloor$

In Section 5.6, we compare the performance of our scheduler with a baseline that multiHTTP simply requests the next chunk in D, whenever it needs to issue a new chunk request over a connection. We show that our scheduler can efficiently reduce the the size of the mHTTP buffer without affecting the performance of mHTTP.

5.6 Performance Evaluation

In this subsection, we study the potential benefit of using mHTTP in different indoor and outdoor scenarios through measurements. We study how mHTTP takes advantage of different types of diversity in the Internet and compare its performance to that of regular HTTP operating over single-path TCP and MPTCP.

We use the download completion time as the performance metric in our evaluation. It is defined as the duration between the first SYN packet from the client and the last data packet from the servers. The download completion times are measured for different file sizes, *i. e.*, 4MB, 16MB, and 64MB. We run each measurement 30 times and show the median, $25-75\,\%$ percentiles (boxes), and dispersion (lines, $5-95\,\%$ percentiles). In each round of measurement, we randomize the configuration sequence to account for traffic dependencies and/or correlation from time to time and from size to size. Specifically, we randomize the order of file sizes, the choice of protocol (*e. g.*, single-path, mHTTP, and MPTCP), and the choice of chunk sizes for mHTTP.

The servers run an Apache2 web server on port 80 and hold copies of the same files. The client uses wget in order to retrieve the files from the servers and runs on the Linux operating system with the kernel version 3.5.7. We use 10 MSS as the initial size of the congestion window. Furthermore, TCP Cubic [37] is used as the default congestion control at the server. It is the default congestion control used in the current version of the Linux kernel.

Our measurements are performed on two testbeds: an easily configurable indoor testbed that emulates different topologies with different characteristics; and an outdoor testbed using one commercial Internet service provider and a major cellular carrier in the US. Our outdoor testbed represents real world scenarios.

For the scenarios in which we enable MPTCP, we use the stable release (version v0.86) downloaded from [117]. To provide a fair comparison between MPTCP and mHTTP, we also use uncoupled congestion control with Cubic for MPTCP. Uncoupled Cubic represents the case where regular TCP Cubic is used on the subflows. It increases the size of the congestion window of each subflow irregardless of the congestion state of the other subflows that are part of the MPTCP session. We set the maximum receive buffer to 6MB to avoid potential performance degradation to MPTCP [96]. Our testbed configuration is optimized for MPTCP. Hence, we observe the best performance we can achieve using MPTCP. Our results show that mHTTP performs very close to this baseline.

We first analyze the overhead of mHTTP and study its effect on the performance of downloading small objects; we then show the benefits of mHTTP when downloading large objects.

5.6.1 Overhead analysis for small objects

mHTTP suffers a performance degradation each time that a connection performs a range request. We evaluate this degradation by measuring the download completion time of a file over a single path connection using HTTP and mHTTP. The client is connected via an Ethernet interface to an Ethernet router with a nominal rate of 100Mbps. The server is also connected to the Ethernet router via an Ethernet interface. A round trip time of 50ms of the round-trip time is generated on the link using a built-in traffic control module of the Linux kernel (qdisc [6]). We evaluate the overhead of mHTTP with different chunk sizes assuming the transfer over regular HTTP as the baseline. We show the results in Figure 5.7. We observe that the overhead is more significant with a small chunk size such as 256KB than with a large chunk size (*e. g.*, 512KB or 1024KB). When the chunk size is 1024KB, we observe that the overhead is around $5 - 10\,\%$. The poor performance of mHTTP with small chunk sizes is puzzling and a topic for future investigation.

We now study the performance of mHTTP for downloading small objects. We evaluate the download completion time of downloading files of various sizes (from 8KB

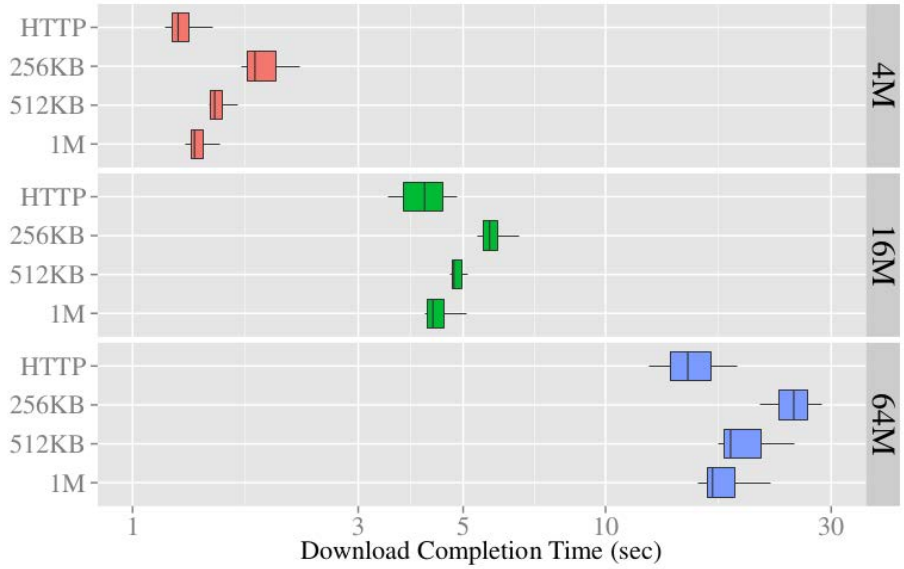


Figure 5.7: Overhead analysis: HTTP vs. mHTTP over a single connection.

to 2MB) over mHTTP using 512KB as the chunk size. We consider a scenario where the client has two interfaces and downloads an object from two servers as illustrated in Figure 5.6. We emulate AN1 and AN2 with Ethernet routers with a nominal rate of 100Mbps. The servers and the client are connected via Ethernet interfaces to the routers. The round-trip times over the connections are set to 50ms. The measurement results are depicted in Figure 5.8. We observe that mHTTP does not provide any performance gain for small object downloads but does no harm either. For object downloads larger than the chunk size (512KB in this measurement), mHTTP provides good performance by utilizing the diversity in the network.

Our results in this section show that mHTTP with large chunk sizes, such as 512KB and 1024KB, provides good performance for small file downloads and introduces negligible overhead when used over a single-path connection. In the rest of the paper, we focus our analysis on the performance of mHTTP for large object downloads.

5.6.2 mHTTP vs regular HTTP

Now, we consider a scenario where the client has multiple interfaces and downloads a file from multiple servers. We assume a 2-server case in this scenario. As illustrated in Figure 5.6, the client is equipped with two interfaces connected to different access networks (ANs). Thus, the client can establish two different connections to two different servers that contain identical copies of the same content. Note that MPTCP cannot be used in this scenario as it is a single-server-oriented protocol.

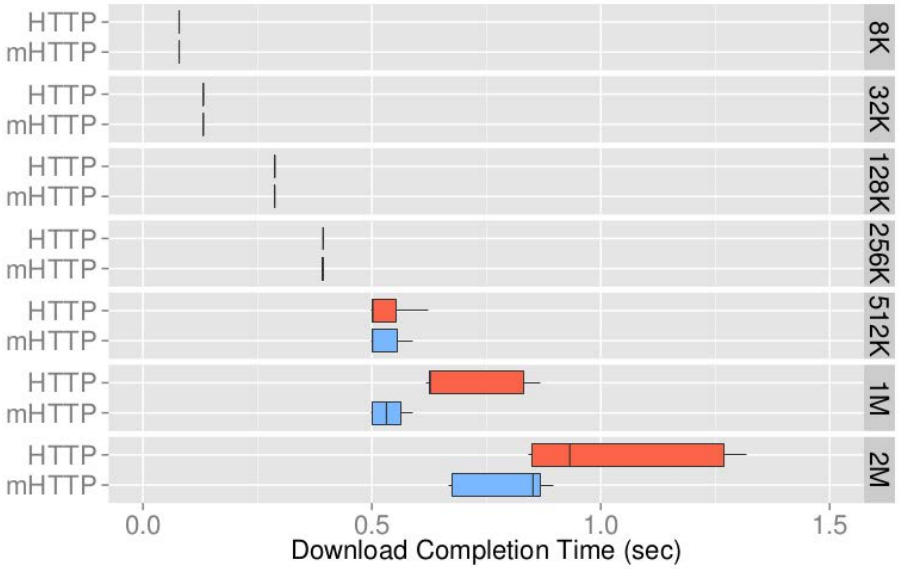


Figure 5.8: The performance of downloading small objects in Scenario 1 (mHTTP chunk size: 512KB).

As the first step, we emulate the above scenario in our indoor testbed where AN1 and AN2 are Ethernet routers with nominal rates of 100Mbps each. Each server is connected via an Ethernet interface to a corresponding router. The client has two Ethernet interfaces that connect to the routers. In order to emulate different link latencies in the scenario, we set round-trip times to 10ms and 50ms on the first link and the second link, respectively. The measurement results are depicted in Figure 5.9. We show download completion times of file sizes 4MB, 16MB, and 64MB. Each figure compares the performance of regular HTTP over a single-path connection with that of mHTTP that uses both connections. The results are presented for different mHTTP chunk sizes. We observe that (1) the connection over the eth0 interface has a much better performance than the one over the eth1 connection; (2) mHTTP greatly benefits from the existing diversity in the network, the performance gain from using mHTTP is larger for larger file sizes; and (3)a chunk size of 1024KB provides the best performance across different file sizes.

In Section 5.5, we proposed a scheduler that decides what chunk to request over each connection to avoid a bottleneck situation such as presented in Figure 5.5. Here, we compare the performance of this scheduler with a baseline that mHTTP simply requests for the next chunk in D, whenever it needs to issue a new chunk request over a connection. Recall that D is the set of chunks that have not yet been requested for download. The experiment is done in our indoor testbed and for file size of 16MB.

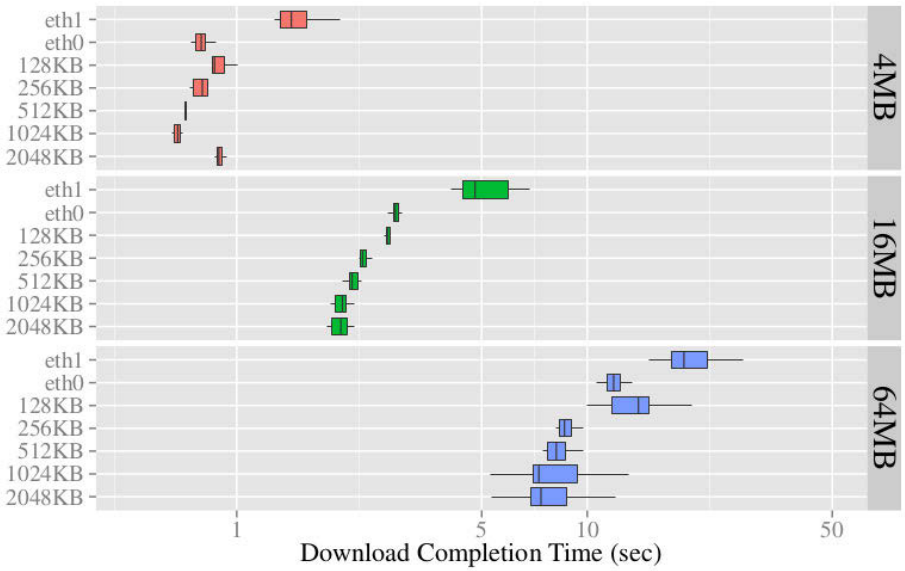
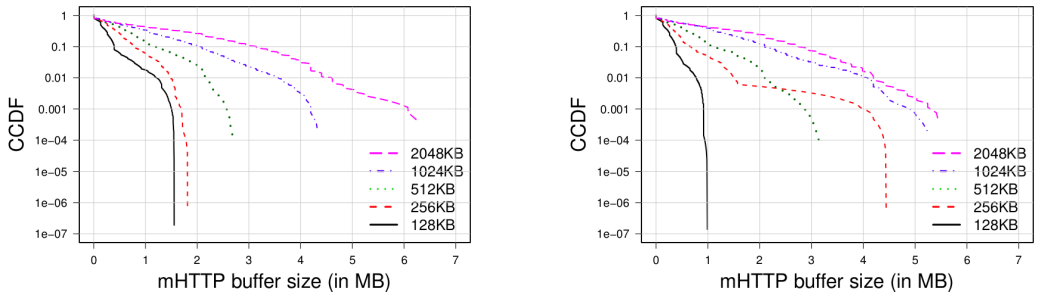


Figure 5.9: Scenario 1 (indoor testbed): download completion time of regular HTTP vs. mHTTP for different file and chunk sizes. mHTTP can efficiently use the bandwidth available to the client and outperforms the best performing connection among regular HTTP connections. 1024KB of the chunk shows the optimal performance in all file sizes.

Measurements show that (1) mHTTP with and without scheduler exhibit similar performance (in term of download completion time). Hence our scheduler does not affect the performance of mHTTP. As the results for mHTTP without scheduling are similar to Figure 5.9, we do not show them in this paper. (2) Our scheduler efficiently reduces the mHTTP buffer size. Figure 5.10 depicts the CCDF (Complementary Cumulative Distribution Function) of mHTTP buffer sizes for both cases. We observe that mHTTP without scheduling requires larger buffer sizes. The results for 2048 chunk size are identical. As in this case we have 8 chunks to be requested over the connections, the scheduler would not have any impact. (3) Furthermore, we observe that the mHTTP buffer size is smaller than 1MB in more than 50 % of the cases. The maximum buffer occupancy is 7 MB. Note that mHTTP buffer uses user level memory and not the the kernel space memory.

Now, we move our measurements to a more realistic network using our outdoor testbed. We configure two servers in a university campus and a mobile device (laptop) as the client. The servers are connected to the Internet via 1Gbps Ethernet cables (*i. e.*, the bottleneck is not at the server). The client device is equipped with two wireless interfaces (WiFi and LTE) that respectively connect to a WiFi network and a cellular network. We show the results of our measurements in Figure 5.11 for file sizes of 4MB, 16MB, and 64MB and for different chunk sizes. We observe from the



(a) mHTTP buffer size with scheduling.

(b) mHTTP buffer size without scheduling.

Figure 5.10: Scenario 1: mHTTP buffer size with and without scheduling. Measurements are done in our indoor testbed and for file size of 16MB.

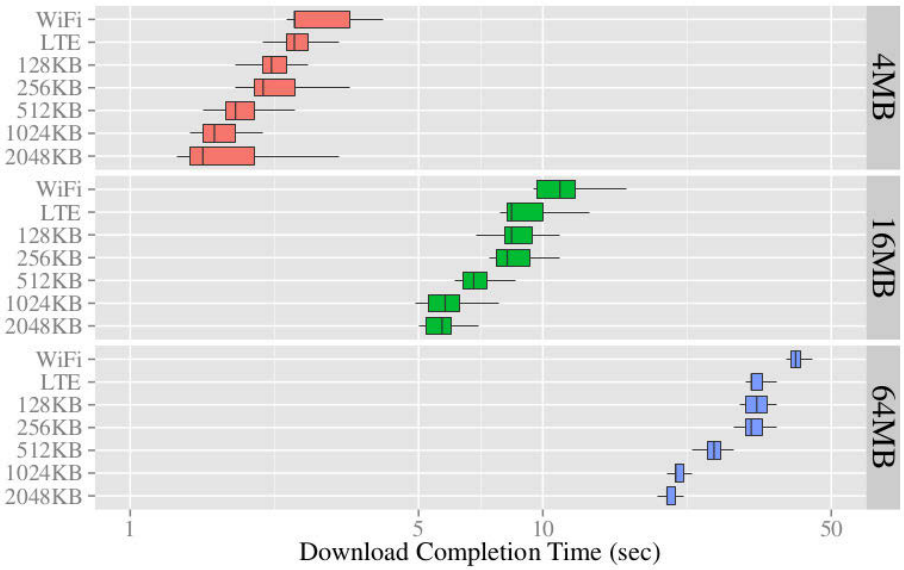


Figure 5.11: Scenario 1 (outdoor testbed): download completion time of regular HTTP vs. mHTTP for different file and chunk sizes. mHTTP can efficiently take advantage of the diversity exists in the network. We observe that mHTTP shows a relatively low performance when using small chunk sizes compared to the measurements with large chunk sizes which is due to the fact that our server configuration is not optimized for mHTTP.

results that (1) LTE and WiFi exhibit very similar performance; and (2) mHTTP can efficiently use the available bandwidth, especially when the chunk size is 1024KB. In this case, we observe that mHTTP's throughput equals the sum of the throughput of LTE and WiFi. Hence, mHTTP fully utilizes the available capacity and shows a

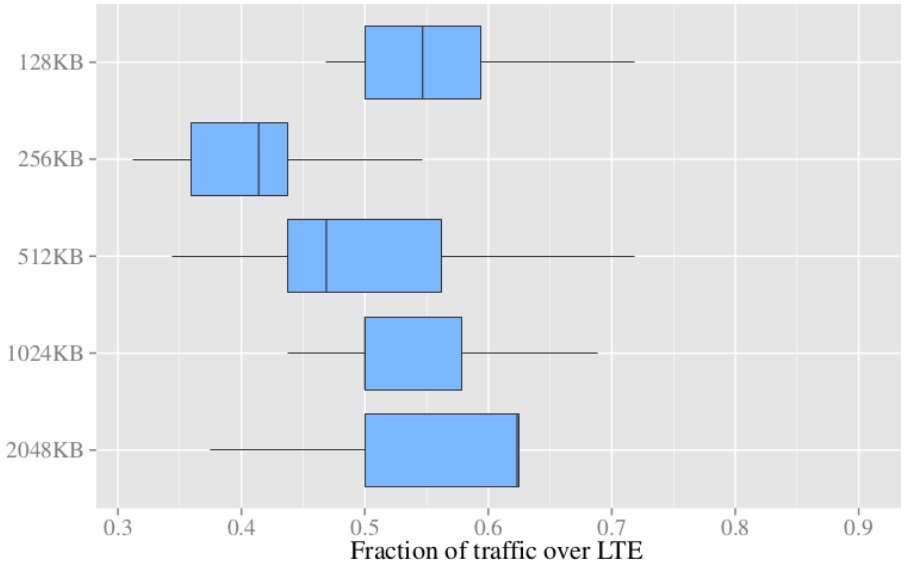


Figure 5.12: Scenario 1 (outdoor testbed): fraction of traffic carried over a LTE connection for file size of 16MB.

substantial performance by reducing the completion time by 50 %. For smaller chunk sizes, we observe a lower performance than that for large chunk sizes. This is mainly due to the overhead of range requests as analyzed in Section 5.6.1. Improving the performance of mHTTP for small chunk sizes is a future research topic.

Figure 5.12 depicts the fraction of traffic carried over the LTE interface using mHTTP. We show the results for different chunk sizes and for a file size of 16MB. We observe from Figure 5.11 that LTE exhibits a slightly higher throughput than WiFi. Hence, we expect mHTTP to send more or less the same amount of traffic over LTE and WiFi. Our results in Figure 5.12 confirm our expectation specifically for large chunk sizes.

5.6.3 mHTTP vs MPTCP for a single server case

Our second scenario focuses on comparing the performance of mHTTP and MPTCP in the multi-homed and single data source environment as illustrated in Figure 5.13.

As in the previous subsection, we first report measurements on our indoor testbed. The topology of the testbed is slightly changed in this scenario: there is only one server, and thus no data source diversity. The server and the client are booted with the MPTCP-enabled kernel when we measure the performance of MPTCP. The results are shown in Figure 5.14. As stated before, we configure our testbed in such a way that it is optimal for MPTCP. Hence, we expect MPTCP with independent cubic

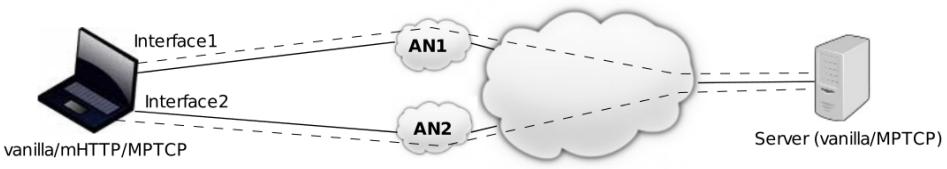


Figure 5.13: Scenario 2: 2 interfaces at the client; 1 server; 2 paths (dashed lines). As in the previous section, we emulate this testbed both indoor and outdoor.

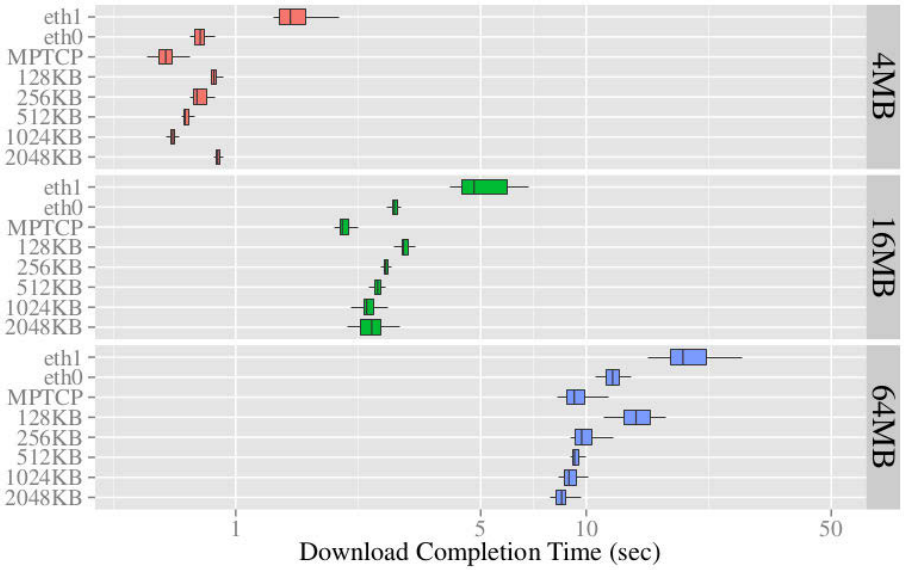


Figure 5.14: Scenario 2 (indoor testbed): download completion time of mHTTP vs. MPTCP and regular HTTP. Our testbed configuration is optimized for MPTCP. Hence, MPTCP is able to fully utilize the available capacity and provide a good performance. We observe that mHTTP perform very close to MPTCP and always outperforms regular HTTP over the best path.

be able to fully utilize the available capacity and provide good performance. The results confirm this: the MPTCP throughput equals to the sum of the throughput of two connections. Moreover, we observe that mHTTP performs closely to MPTCP when the chunk size is 1024KB.

Furthermore, we observe for 64MB file size, and for large chunk sizes, that mHTTP outperforms MPTCP. This is due to the fact that MPTCP uses a shared TCP receive buffer which can limit its performance when paths have different characteristics [96].

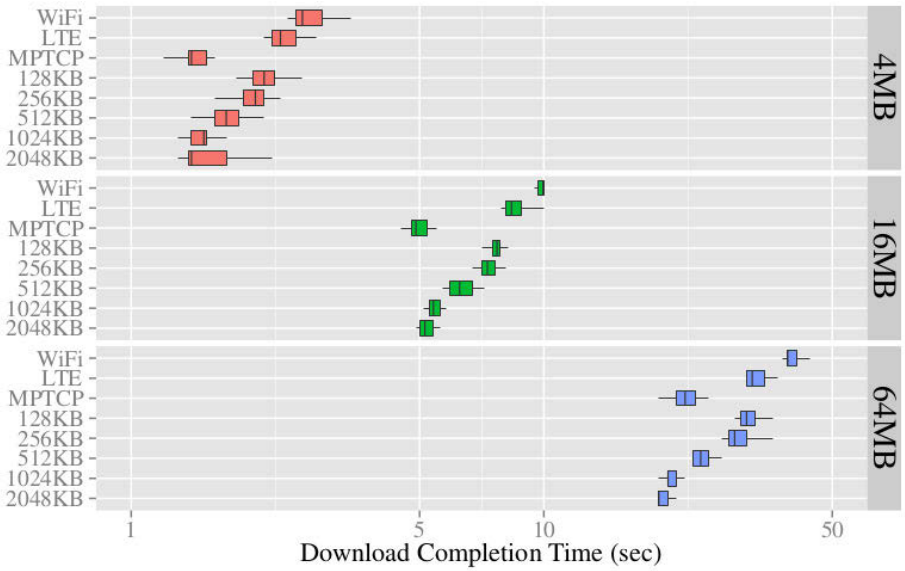


Figure 5.15: Scenario 2 (outdoor testbed): download completion time of mHTTP vs. MPTCP and regular HTTP. We observe that MPTCP is able to fully use the available bandwidth and mHTTP performs close to MPTCP.

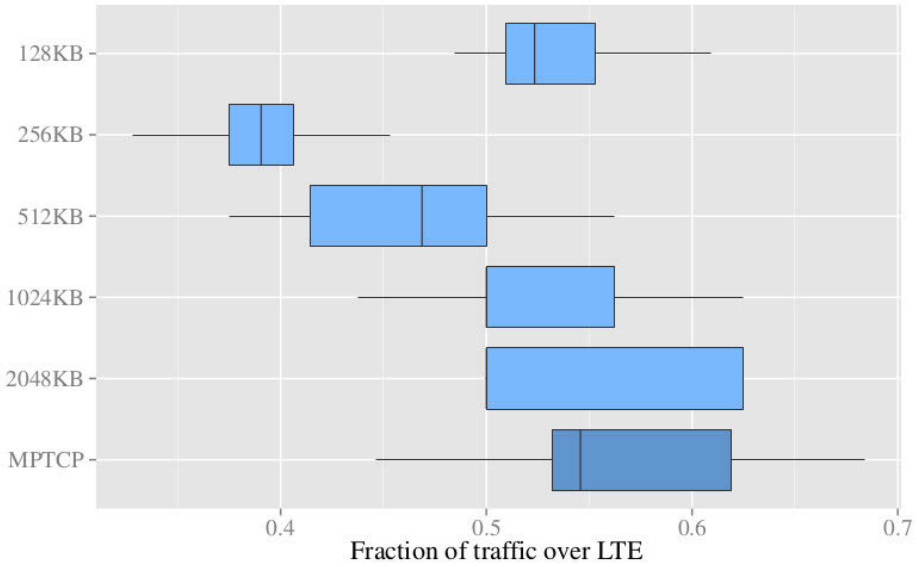


Figure 5.16: Scenario 2 (outdoor testbed): fraction of traffic carried over a LTE connection for 16MB file using mHTTP as well as MPTCP. We show the results for 16MB file.

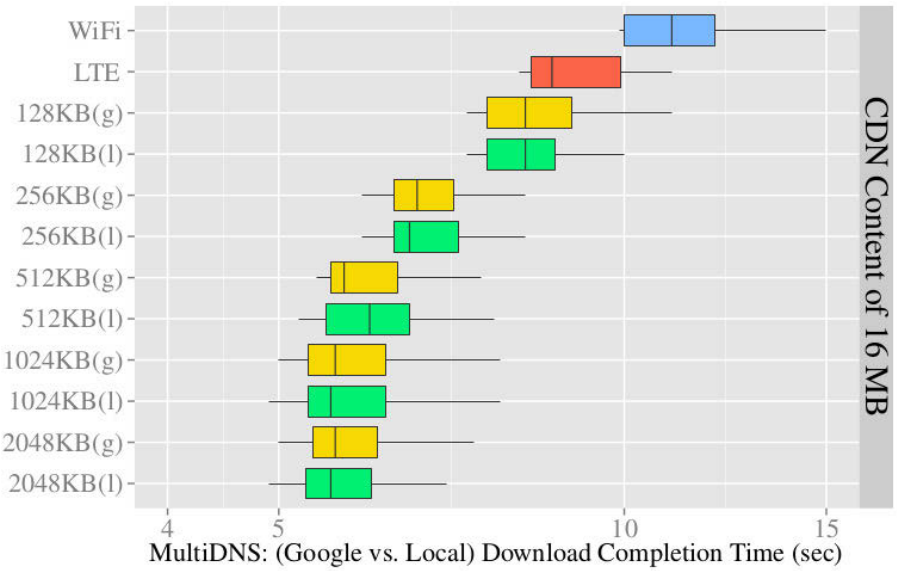


Figure 5.17: Measurement on a CDN infrastructure. The results are shown for two cases: the first case uses IP addresses obtained from DNS queries sent to Google DNS (shown with (g) in the y-axis); and the second case that uses IP addresses obtained from the local DNS of each of the interfaces networks (shown with (l) in the y-axis).

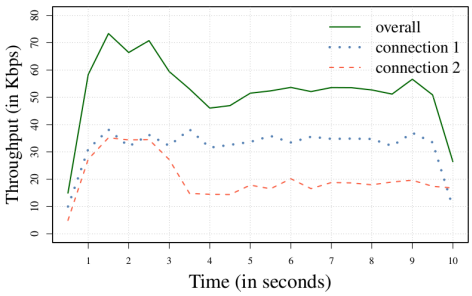
However, mHTTP uses a separated TCP receive buffer for different established connections and hence can perform well in such a situation.

Now, we show measurement results from our outdoor testbed: a server residing at a university campus and a client equipped with LTE and WiFi network interfaces. The results are depicted in Figure 5.14. Again, we observe that MPTCP fully uses available capacity and mHTTP performs close to MPTCP, especially for large file sizes and 1024KB as the chunk size.

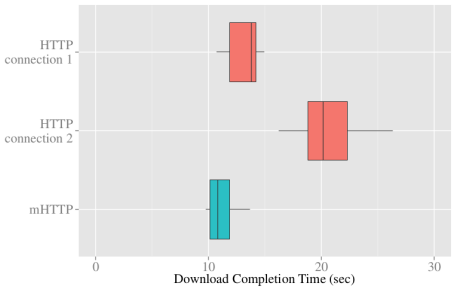
Figure 5.16 depicts the fraction of traffic transmitted over the LTE connection for both mHTTP and MPTCP. We show the results for 16MB file size. As WiFi and LTE connections exhibit similar performance, we expect that MPTCP and mHTTP will transmit more and less the same amount of traffic over each of these connections as observed in the results. Moreover, we observe some differences between using different chunk sizes for mHTTP.

5.6.4 mHTTP in a multi-source CDN

Finally, we conduct performance measurements on an existing CDN infrastructure. We choose a 16MB file from a well known site on *Alexa.com*'s top-50 list, where



(a) Throughput over each of the connection



(b) Download completion time

Figure 5.18: The RTT of the second connection is changed to 100ms after 3 second. mHTTP is robust to such a latency change and leverages his possible resources over each of these connections.

the content is hosted in a CDN. We evaluate the performance of mHTTP when multiDNS uses the following two approaches: to simply use Google's public DNS or to leverage separate local DNS resolvers.

For the first approach, multiDNS queries Google's DNS over each interface separately, and uses the set of IP addresses returned for each interface. For the second approach, multiDNS sends a DNS query over each interface to the local DNS of that access network to obtain content sources' IP addresses.

We depict the download times of the file using single-path or mHTTP with different chunk sizes in Figure 5.17. Note that for single-path TCP, each interface by default queries its local DNS resolver. We observe that mHTTP reduces download times by up to 50 % when compared to the single-path case and performs very well across a wide range of chunk sizes. Moreover, no significant differences are observed for both approaches that multiDNS uses. Our results in Figure 5.17 confirms that mHTTP can benefit from the path diversity in the Internet and can fully utilize the available bandwidth.

5.6.5 mHTTP is robust to the changes

Finally, we show through an example how mHTTP performs when one of its connections experiences performance drops (due to the congestion either on the path or at the server). We use a scenario similar to what is depicted in Figure 5.6. We emulate this scenario in our indoor testbed. AN1 and AN2 are Ethernet routers with nominal rates of 100Mbps. Each server is connected via an Ethernet interface to a corresponding router.

The RTTs of both connections are initially set to 50 ms. The RTT of the second connection is configured to be changed to 100ms 3 seconds after the transfer begins.

We investigate how mHTTP reacts to this change. We show the results for 64MB file downloads when 1024KB chunk size is used.

Figure 5.18(a) depicts the throughput on each of the connection of mHTTP and the overall throughput of mHTTP. We show the results for one experiment run. We observe that upon the RTT change of the second connection, the throughput over this connection decreases. However, mHTTP is robust to such a change and takes advantage of the diversity in the network.

Figure 5.18(b) depicts the download completion time of mHTTP and compares its performance with when we use single-path HTTP over each of these connections (recall that the performance of the second connection drop after 3 second). We show the results from 30 rounds of measurement. We observe that mHTTP provides a significant performance gain, especially when we compare it with single-path HTTP over the second connection.

5.7 Related Work

The goal of our study is to boost the speed of the HTTP-based content delivery. Indeed, the need for such a latency reduction in the Internet has already been acknowledged by network communities.

5.7.1 Multipath Approaches

One of the closest siblings of mHTTP is MPTCP [31,96] which is an extension of the regular TCP that enables a user to spread its traffic across disjoint paths. Although MPTCP focuses on the path diversity between a single server and a single receiver and requires the modification at both end-hosts, the fundamental idea behind these two protocols is the same. Furthermore, mHTTP sheds light on solving a middlebox conundrum [12] that MPTCP currently struggles with. Kaspar [54] thoroughly studies the path diversity in the Internet and discusses use cases on the transport layer as well as on the application layer. His work and mHTTP have many features in common except that his work is limited on a single client/server scenario and it does not take scheduling into the design consideration.

5.7.2 Multi-source Approaches

Content Distribution Network (CDN) is a key technology for reducing the delivery latency in today's Internet and the performance of CDN has been evaluated by many studies [44, 52, 60, 99]. CDNs provide widely distributed servers with multiple copies of the content available at different locations. CDN typically selects the content server based on the IP address of client's DNS server and it often makes an incorrect

suggestion due to the use of a public DNS [3] or the malfunction of the IP geolocation database [91]. To improve the performance of the server selection, mechanisms such as PaDIS [90] or ALTO [107] have been discussed in the community. Our goal is to leverage such content distribution infrastructures. However, mHTTP does not limit the communication to a single server. Instead, mHTTP connects to multiple content servers for utilizing path and server diversity of the Internet. Tian *et al.* [112] has proposed a mechanism that is an extension of DASH [1] video streaming protocol, but heavily relies on specific features of DASH. mHTTP, on the other hand, can be used for any HTTP-type traffic, including streaming contents.

5.7.3 Single Path Approaches

Google has recently proposed a new protocol, SPDY [11, 139], which shares the ultimate goal with mHTTP, *i. e.*, reducing the user latency. These two protocols (mHTTP and SPDY) have a similar architecture that does not need any modifications in existing applications. SPDY uses only one server and one interface at a time and utilizes a single TCP connections as if there are multiple connections in it. To achieve this, server-side socket APIs must support SPDY and that clearly distinguishes SPDY from mHTTP. However, features of mHTTP and SPDY are mutually exclusive, thus these two protocols may even be merged in the same platform.

5.7.4 Application Specific Approaches

BitTorrent implements a sophisticated mechanism which enables users to download the same content from multiple sources [19]. However, BitTorrent is an application specific protocol and it needs modification both at the sender's and at the receiver's side. Download managers, often run as add-on software in a web browser or as standalone software, can be other examples of application specific approaches, *e. g.*, [126, 131].

5.8 Summary

Advantages of simultaneously utilizing multiple paths over a network communication are widely evaluated and understood [10, 16, 78, 94]. Given the fact that HTTP accounts for more than 60 % [67] of today's Internet traffic and that the major fraction of the total web servers are operated on content distribution infrastructures [44, 113], it is meaningful to broaden the benefit via globally replicated content sources. However, convincing application developers and content providers to modify/update their software is practically infeasible within a reasonable amount of time. mHTTP's key

contribution is to bring significant benefits to the end-to-end content delivery by utilizing the path diversity in the Internet without any changes on existing applications and the server-side network stack.

By evaluating it in testbeds and in real-world experiment, we present that mHTTP shows a significant performance increase in large content delivery as compared to the normal HTTP. Furthermore, mHTTP exhibits the similar performance to that of MPTCP even with a single data source, note that MPTCP is restricted to the use of a single server. For this reason, mHTTP can be seen as a quickly deployable alternative of MPTCP for HTTP traffic.

Our results show that the performance gain of mHTTP is relatively low when small chunk sizes are used. Part of the problem is due to our testbed configuration. Additionally, our implementation is still in the testing phase. Optimizing the HTTP parser and restructuring the mHTTP buffer will provide substantial performance increase. This is a topic for future investigation.

For small object downloads, we observe that mHTTP does not provide a high performance gain, but does not harm either. Moreover, we can modify mHTTP such that it does not establish multiple paths if the object size is relatively small. Hence, for the small flows, mHTTP will fall back to regular HTTP. Furthermore, we can integrate ideas like socket intense proposed in [104] in our implementation to better deal with small flows. This is part of our future research agenda.

In regard to the comparison with MPTCP in single server scenarios, we observe that mHTTP exhibits similar performance as MPTCP for large chunk sizes, *e. g.*, 1024KB, and for downloading large objects. Moreover, previous studies show that MPTCP, similarly to mHTTP, does not provide a high performance gain for small object downloads [16]. Hence, we consider mHTTP to be a viable alternative when running HTTP. Note that MPTCP requires changes to the kernel, both at the sender and receiver. mHTTP, on the other hand, requires only receiver-side modifications which are restricted to the socket interface.

6

Conclusion and Future Work

The trend shows that the number of Internet users and network-enabled end-devices grows at increasing rate [129, 130], thus, adds complexity to the Internet every day. Furthermore, as the volume of digital content is constantly growing (*e. g.*, the size of high definition videos), its delivery applications (*e. g.*, YouTube, NetFlix, or One-Click Hosters) are becoming more and more bandwidth demanding. To cope with the increasing demand for content, providers started to replicate their content in multiple data centers across the world (*i. e.*, CDNs) which increases the network complexity yet again.

Viewed from a different standpoint, the complexity is an outcome of various types of diversity in the Internet such as address space diversity, upstream link diversity, interface diversity, path diversity, and data source diversity. To better understand the problems and opportunities enabled by this complexity, research communities are actively exploring network diversity from different perspectives. In this thesis, which is inspired by such a movement, three of the most pressing problems in today's Internet are tackled.

6.1 Summary of Thesis Contributions

As the first key contribution of this thesis, we evaluate the adoption level of IPv6 from an IXP's viewpoint in order to assess the current situation of the transition in address spaces. The outcomes of the measurement show that, in terms of the traffic contribution, we are still standing near the starting line (*i. e.*, less than 1 % of the

total global traffic). Considering that this transition has already started two decades ago, it is hard to deny that, in innovation, the Internet Protocol (IP) is the slowest Internet technology. Despite its slow transition speed, some results of analysis give us a positive outlook for IPv6 deployment. Mainly, two observations support our anticipation.

First, the growth of IPv6 traffic shown within one year between two global IPv6 events (World IPv6 Day and World IPv6 Launch Day) is almost corresponding to the one that has happened for the last two decades, meaning that network operators are well aware of the necessity of moving towards the next version of the Internet Protocol and that they are willing to participate in collaborative endeavors.

Second, the application mix in IPv6 traffic shows the clear domination of HTTP with more than a half share which is similar to the application mix in IPv4 traffic. This is a positive phenomenon since it means that the IPv6 traffic actually transfers content and is not used as an experiment.

Along with the assessment study of IPv6 adoption, two application protocols (NNTP and World of Warcraft) that have antithetical characteristics (bandwidth intensive vs. latency intensive) are picked up and analyzed within IPv4. Through these two measurement studies, we examine how the Internet is utilized by end-users today.

One of the most relevant outcomes of our study is that the major bulk of content delivered in the Internet is high-volume binary data, *e. g.*, video/audio and compressed binary data, encapsuled in traditional text-friendly protocols, *e. g.*, NNTP (studied in this thesis) and/or HTTP (partly covered in this thesis, but mainly studied elsewhere [49, 67, 92, 106]). This result leads us to the choice of HTTP as the platform of the technique we propose to increase the performance of end-to-end data transfer, *i. e.*, mHTTP.

The second major contribution of this thesis is the thorough analysis on a future routing protocol, *i. e.*, the Locator/ID Separation Protocol (LISP), that deals with the scalability problem mainly caused by certain types of Internet diversity, *e. g.*, interface diversity and upstream link diversity. Specifically, our interests in this protocol are largely concentrated on the LISP Cache that temporarily stores mappings between its two orthogonal address spaces (Routing Locator and End-point Identifier).

By reproducing the behavior of the LISP Cache based on the real traffic driven emulation, we find that even a cache timeout as short as 60 seconds provides a high performance (*i. e.*, hit ratio) and that the cost for increasing the security level of LISP is affordable.

The performance degradation caused by an initial cache-miss of LISP is mainly presented as the delay on a TCP communication between applications. This is due to TCP's retransmission mechanism bound to the LISP Cache's common policy to drop a packet that causes a cache-miss.

In case of a LISP router failure in a multihomed stub network, another LISP router overtakes EID-to-RLOC mapping tasks for the encapsulation (and decapsulation in case of the *symmetric LISP*) from the failed router. Thus, this causes a storm of cache-misses since the newly involved router does not have the knowledge of how to map IDs to RLOCs. The same phenomenon happens when the router comes back from the failure. To this end, we evaluate the impact of such a failure (and a recovery), then suggest a solution by synchronizing mapping information within a small group of ingress tunnel routers (ITRs). Our evaluation shows that the overhead due to the synchronization is acceptable, while it greatly suppresses the loss of packets upon a failure and recovery of a router.

Finally, as the third contribution of this thesis, we design and implement the Multi-source Multipath HTTP (mHTTP). mHTTP is an easily deployable solution for increasing the performance of the end-to-end delivery within Content Distribution Networks (CDNs) by utilizing various types of network diversity. As the name suggests, mHTTP is an extension of HTTP that is designed to pull content from multiple different data sources (*i. e.*, seen as data source diversity in this thesis). Our approach is thoughtfully designed not to make any changes at the server-side or at applications, thus the modification is only to socket APIs at the receiver-side. This makes mHTTP easy to adopt in the current content delivery infrastructure.

Our evaluation shows that mHTTP has a remarkable performance enhancement (in terms of download completion time and throughput) in large content delivery as compared to the normal HTTP over TCP. Furthermore, mHTTP exhibits a similar performance to that of MPTCP even with a single data source. For this reason, mHTTP can be seen as a quickly deployable alternative of MPTCP for HTTP traffic.

As mHTTP does not provide a high performance gain for short flows, it is sensible to disable the multi-connection feature and to fall back to the normal HTTP as soon as the size of content is known to the receiver. Indeed, for flows smaller than the size of a chunk, *i. e.*, a segment to be delivered over one connection at each time, there will be no need to establish the second connection anyway. According to the result of our measurement, the optimal size of a chunk is around 1 MB.

6.2 Future Work

Future measurement study needs to cover more application protocols to better understand the usage pattern of today's Internet. Moreover, it is crucial to track IPv6 deployment since finding out when IPv6 transition can really happen is of great interest to everyone in the research community. Furthermore, we will continue observing the alteration, *e. g.*, application mix, throughput, topology, etc., in traffic and examine more factors that add complexity and diversity to the Internet.

In order to estimate time required for the full deployment of the Locator/ID Separation Protocol (LISP), we plan to study its deployment techniques. Moreover, we will develop methods to maximize the performance of LISP and to minimize the impact of the initial cache-miss. To achieve this, we plan to address the two unanswered questions.

First, our performance analysis on TCP communication was largely directed at determining how to optimize the TCP parameters for Locator/ID Split technologies. We will take various TCP parameters, *e. g.*, delayed ACK time and initial congestion window size, consideration into account to determine the optimal TCP parameters. In addition, we will closely observe the impact of the optimization to find out whether or not it hinders the communication between a TCP speaker in a LISP-enabled site and a normal TCP speaker.

Second, we will extend the synchronization mechanism to ETRs and develop detailed specifications for implementation and experimentation in the *lisp4.net* testbed. Furthermore, the question of what is the best trade-off between tight synchronization and signaling overhead is still unanswered.

Although the Multi-source Multipath HTTP (mHTTP) is a solely receiver-oriented mechanism, it is not easy task to gain users and keep them engaged without the support of operating systems. Thus, along with the main branch of mHTTP that is implemented on the linux operating system, we plan to port it to operating systems of widely-used mobile hand-held devices such as *Android*.

For the further development of mHTTP, we will implement more sophisticated scheduling mechanism such as coupled-control [95] or OLIA [55]. As our design goal is not to modify servers, this implementation can be realized by modifying the TCP kernel of the receiver. The idea is that we can limit/regulate the transmission rate of a connection by adjusting the receive window size advertised by the receiver.

We also plan to extend mHTTP to other use cases such as a streaming content delivery (*e. g.*, YouTube and/or NetFlix) or One-Click Hosters (OCHs). For this, the data chunking mechanism (currently only based on HTTP's ranged request feature) will be independently developed from the core of the mHTTP implementation, hence applications that have an individual data chunking mechanism will be supported by mHTTP in the future. Specifically, we are interested in studying if using mHTTP can reduce the start-up latency of streaming contents [98]. The performance study of mHTTP in high Bandwidth-Delay-Product environments is another future research topic.

Some academic circles study the cross-layer cooperation between Locator/ID Separation Protocol (LISP) and Multipath TCP (MPTCP) [21]. The potential behind this cooperation lies on the improved path discovery by sharing topology learned via LISP operations. In principle, the same mechanism can be applied to mHTTP. Thus, in our future work, we will go on examining opportunities of such a cooperation in order to ensure the quality of mHTTP's path selection.

At this point, our long journey through network diversity is coming to an end. As we witnessed throughout this thesis, network diversity has two faces. One face shows us several scalability issues that the Internet community worries about, *e. g.*, address space scarcity and routing table saturation. The other face sheds light on the performance of end-to-end data transfer. Our contributions in this thesis are made for a better understanding of such challenges and will help the further evolution of the Internet.

Acknowledgements

Looking back on the past, the moment that my advisor, Anja Feldmann, offered me a doctoral student position was one of the most important turning points in my life. Who, among my family members and friends, could possibly imagine that I will persue a doctoral degree? Moreover, during the course of my Ph.D., Anja always helped me out of a short-sighted approach to a problem and guided me with valuable feedback to see a big picture of the problem. Therefore, I would like to thank Anja foremost.

I am deeply grateful to Franziska Dreke, for everything she has done to support me. Without her endless encouragement, motivation, patience, and love, I never could achieve this today. Besides, even with her busy schedule she always found time to proof-read my thesis.

I thank Prof. Don Towsley who inspired me (and other mHTTPers) with the great interest and enthusiasm. I am especially grateful to him for his tireless and continuous input on problem solving.

I would also like to thank all my co-advisors and collaborators, particularly, Luigi Iannone who helped me with his extensive knowledge and rich experience in research. I also thank Ramin Khalili for the countless brainstorming sessions and valuable discussion.

Now, I wish to express my special gratitude to everyone at FG-INET for the great time I had in our group.

Last but not least, I would like to acknowledge the following people, to whom I offer my heartfelt gratitude for making Germany my second home.
Detlef Dreke, Roselind Dreke, Luisa Dreke, Patricia Dreke, Sebastian Blume, Tim Kauermann, Alexandra Kramarz, Christina Brückmann.

List of Figures

List of Tables

Bibliography

[1] Adhikari, V., Guo, Y., Hao, F., Varvello, M., Hilt, V., Steiner, M., and Zhang, Z. Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery. In *Proc. IEEE INFOCOM* (2012).

[2] Ager, B., Chatzis, N., Feldmann, A., Sarrar, N., Uhlig, S., and Willinger, W. Anatomy of a Large European IXP. In *Proc. the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication* (2012).

[3] Ager, B., Muhlbauer, W., Smaragdakis, G., and Uhlig, S. Comparing DNS Resolvers in the Wild. In *Proc. ACM Internet Measurement Conference* (2010).

[4] Ager, B., Schneider, F., Kim, J., and Feldmann, A. Revisiting Cacheability in Times of User Generated Content. In *Proc. IEEE Global Internet* (2010).

[5] Allbery, R., and Lindsey, C. Netnews Architecture and Protocols. RFC 5537, Nov 2009.

[6] Almesberger, W., et al. Linux Network Traffic Control - Implementation Overview, 1999.

[7] Antoniades, D., Markatos, E., and Dovrolis, C. One-click Hosting Services: a File-sharing Hideout. In *Proc. ACM Internet Measurement Conference* (2009).

[8] Atkinson, R. ILNP concept of operations. *Work in Progress* (2011).

[9] Azzouna, N., and Guillemin, F. Analysis of ADSL Traffic on an IP Backbone Link. In *Proc. IEEE GLOBECOM* (2003).

[10] Barre, S., Paasch, C., and Bonaventure, O. Multipath TCP: from Theory to Practice. In *Proc. International IFIP TC 6 Networking Conference* (2011).

[11] Belshe, M., and Peon, R. SPDY Protocol. `http://tools.ietf.org/html/draft-mbelshe-httpbis-spdy-00`, 2012.

[12] Bonaventure, O. Multipath TCP Tutorial at IEEE Cloudnet 2012. `http://www-phare.lip6.fr/cloudnet12/Multipath-TCP-tutorial-cloudnet.pdf`, 2012.

[13] Breslau, L., Cao, P., Fani, L., Phillips, G., and Shenker, S. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. IEEE INFOCOM* (1999).

[14] Chatzis, N., Smaragdakis, G., Böttger, J., Krenc, T., and Feldmann, A. On the benefits of using a large IXP as an Internet vantage point. In *Proc. ACM Internet Measurement Conference* (2013).

[15] Chen, K., Huang, P., Huang, C., and Lei, C. Game Traffic Analysis: An MMORPG Perspective. In *Proc. the international workshop on Network and operating systems support for digital audio and video* (2005).

[16] Chen, Y., Lim, Y., Gibbens, R., Nahum, E., Khalili, R., and Towsley, D. A Measurement-based Study of Multipath TCP Performance over Wireless Networks. In *Proc. ACM Internet Measurement Conference* (2013).

[17] Claffy, K., Hyun, Y., Keys, K., Fomenkov, M., and Krioukov, D. Internet Mapping: from Art to Science. In *Proc. Conference For Homeland Security, Cybersecurity Applications & Technology* (2009).

[18] Clayton, R. Internet Multi-homing Problems: Explanations from Economics. In *Proc. workshop on the Economics of Information Security* (2009).

[19] Cohen, B. Incentives Build Robustness in BitTorrent. In *Proc. workshop on Economics of Peer-to-Peer systems* (2003).

[20] Colitti, L., Gunderson, S., Kline, E., and Refice, T. Evaluating IPv6 Adoption in the Internet. In *Proc. Passive and Active Measurement Conference* (2010).

[21] Coudron, M., Secci, S., Pujolle, G., Raad, P., and Gallard, P. Cross-layer Cooperation to Boost Multipath TCP Performance in Cloud Networks. In *Proc. IEEE Cloud Networking* (2013).

[22] Damas, J., Graff, M., and Vixie, P. Extension Mechanisms for DNS (EDNS (0)). RFC 6891, 2013.

[23] Dhamdhere, A., Luckie, M., Huffaker, B., Elmokashfi, A., Aben, E., et al. Measuring the Deployment of IPv6: Topology, Routing and Performance. In *Proc. ACM Internet Measurement Conference* (2012).

[24] Dreger, H., Feldmann, A., Mai, M., Paxson, V., and Sommer, R. Dynamic Application-layer Protocol Analysis for Network Intrusion Detection. In *Proc. Usenix Security Symp.* (2006).

[25] Farinacci, D., Fuller, V., Meyer, D., and Lewis, D. Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT). RFC 6836, 2013.

[26] Farinacci, D., Fuller, V., Meyer, D., and Lewis, D. Locator/ID Separation Protocol (LISP). RFC 6830, 2013.

[27] Feather, C. Network News Transfer Protocol (NNTP). RFC 3977, Oct 2006.

[28] Feldmann, A. Internet Clean-slate Design: What and Why? *ACM Computer Communication Review* (2007).

[29] Feldmann, A., Cittadini, L., Mühlbauer, W., Bush, R., and Maennel, O. HAIR: Hierarchical Architecture for Internet Routing. In *Proc. workshop on Re-architecting the Internet* (2009).

[30] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, 1999.

[31] Ford, A., Raiciu, C., Handley, M., Barre, S., and Iyengar, J. Architectural Guidelines for Multipath TCP Development. RFC 6182, 2011.

[32] Freed, N., and Borenstein, N. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, Nov 1996. Updated by RFCs 2184, 2231, 5335, 6532.

[33] Fuller, V., and Farinacci, D. LISP Map Server Interface. `http://www.ietf.org/proceedings/82/id/draft-ietf-lisp-ms-15.txt`, 2012.

[34] Gao, L., Yang, J., Zhang, H., Qin, D., and Zhang, B. What's Going on in Chinese IPv6 World. In *Proc. IEEE NOMS* (2012).

[35] Geography, G. I. TeleGeography Research. `http://www.telegeography.com/product-info/gb/download/executive-summary.pdf`, 2009.

[36] Gryaznov, D. Malware in Popular Networks. In *Proc. Virus Bulletin International Conference* (2005).

[37] Ha, S., Rhee, I., and Xu, L. CUBIC: A New TCP-friendly High-speed TCP Variant. *ACM SIGOPS Operating System Review 42* (2008).

[38] Hafsaoui, A., Collange, D., and Urvoy-Keller, G. Revisiting the Performance of Short TCP Transfers. In *Proc. International IFIP TC 6 Networking Conference* (2009).

[39] Helbing, J. yEnc - Efficient Encoding for Usenet and eMail. `http://www.yenc.org/`, 2003.

[40] Hinden, R. New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG. RFC 1955, 1996.

[41] Horton, M. UUCP Mail Interchange Format Standard. RFC 976, Feb 1986. Updated by RFC 1137.

[42] Horton, M., and Adams, R. Standard for Interchange of USENET Messages. RFC 1036, Dec 1987. Obsoleted by RFCs 5536, 5537.

[43] Hsieh, H., Kim, K., Zhu, Y., and Sivakumar, R. A Receiver-centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces. In *Proc. ACM MOBICOM* (2003).

[44] Huang, C., Wang, A., Li, J., and Ross, K. Measuring and Evaluating Large-scale CDNs. In *Proc. ACM Internet Measurement Conference* (2008).

[45] Huitema, C. Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs). RFC 4380 (Proposed Standard), Feb 2006.

[46] Iannone, L., and Bonaventure, O. On the Cost of Caching Locator/ID Mappings. In *Proc. ACM CoNEXT* (2007).

[47] Iannone, L., and Levä, T. Modeling the Economics of Loc/ID Split for the Future Internet. In *Future Internet Assembly* (2010).

[48] Iannone, L., Saucez, D., and Bonaventure, O. Locator/ID Separation Protocol (LISP) Map-Versioning. RFC 6834, 2013.

[49] Inc., S. 009 Global Broadband Phenomena. `http://www.sandvine.com/news/global_broadband_trends.asp`, 2009.

[50] Jain, A., Lewis, D., Fuller, V., and Ermagan, V. LISP Delegated Database Tree.

[51] Jakab, L., Cabellos-Aparicio, A., Coras, F., Saucez, D., and Bonaventure, O. LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System. *IEEE J. Selected Areas in Communications 28* (2010).

[52] Karagiannis, T., Rodriguez, P., and Papagiannaki, K. Should Internet Service Providers Fear Peer-assisted Content Distribution? In *Proc. ACM Internet Measurement Conference* (2005).

[53] Karpilovsky, E., Gerber, A., Pei, D., Rexford, J., and Shaikh, A. Quantifying the Extent of IPv6 Deployment. In *Proc. Passive and Active Measurement Conference* (2009).

[54] Kaspar, D. *Multipath Aggregation of Heterogeneous Access Networks.* PhD thesis, University of Oslo, 2012.

[55] Khalili, R., Gast, N., Popovic, M., Upadhyay, U., and Le Boudec, J. MPTCP is not Pareto-optimal: Performance Issues and a Possible Solution. In *Proc. ACM CoNEXT* (2012).

[56] Kihl, M., Aurelius, A., and Lagerstedt, C. Analysis of World of Warcraft Traffic Patterns and User Behavior. In *Proc. IEEE ICUMT* (2010).

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

[58] Kim, J., Iannone, L., and Feldmann, A. A Deep Dive into the LISP Cache and What ISPs Should Know about It. In *Proc. International IFIP TC 6 Networking Conference* (2011).

[59] Kim, J., Schneider, F., Ager, B., and Feldmann, A. Today's Usenet Usage: NNTP Traffic Characterization. In *Proc. IEEE Global Internet* (2010).

[60] Krishnamurthy, B., Wills, C., and Zhang, Y. On the Use and Performance of Content Distribution Networks. In *Proc. ACM Internet Measurement Workshop* (2001).

[61] Labovitz, C., McPherson, D., and Iekel-Johnson, S. NANOG 47: 2009 Internet Observatory Report. `http://www.nanog.org/meetings/nanog47/abstracts.php?pt=MTQ1MyZuYW5vZzQ3&nm=nanog47`, 2009.

[62] Lauinger, T., Kirda, E., and Michiardi, P. Paying for Piracy? An Analysis of One-click Hosters' Controversial Reward Schemes. In *Proc. Research in Attacks, Intrusions, and Defenses* (2012).

[63] Lewis, D., and Meyer, D. Architectural Implications of Locator/ID Separation. `http://tools.ietf.org/html/draft-meyer-loc-id-implications-01`, 2009.

[64] Liebenau, J., Elaluf-Calderwood, S., and Kärrberg, P. European Internet Traffic: Problems and Prospects of Growth and Competition.

[65] Luciani, J., Armitage, G., Doraswamy, N., and Halpern, J. Server Cache Synchronization Protocol. RFC 2334, 1998.

[66] Madhyastha, H., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., and Venkataramani, A. iPlane: An Information Plane for Distributed Services. In *Proc. OSDI* (2006).

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

[68] Maier, G., Sommer, R., Dreger, H., Feldmann, A., Paxson, V., and Schneider, F. Enriching Network Security Analysis with Time Travel. In *ACM Computer Communication Review* (2008).

[69] Malone, D. Observations of IPv6 Addresses. In *Proc. Passive and Active Measurement Conference* (2008).

[70] Massar, J. AYIYA: Anything in Anything. `http://tools.ietf.org/html/draft-massar-v6ops-ayiya-02`, Jan 2005.

[71] Mathy, L., and Iannone, L. LISP-DHT: Towards a DHT to Map Identifiers onto Locators. In *Proc. ACM CoNEXT* (2008).

[72] Meng, X., Xu, Z., Zhang, B., Huston, G., Lu, S., and Zhang, L. IPv4 Address Allocation and the BGP Routing Table Evolution. *ACM Computer Communication Review 35* (2005).

[73] Miller, J., and Crowcroft, J. Avatar Movement in World of Warcraft Battlegrounds. In *Proc. ACM NetGames* (2009).

[74] Mockapetris, P. DOMAIN NAMES - CONCEPTS AND FACILITIES. RFC 1034, 1987.

[75] Mockapetris, P. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. RFC 1035, 1987.

[76] Moskowitz, R. Host Identity Protocol Architecture. RFC 4423, 2012.

[77] Murchison, K., Lindsey, C., and Kohn, D. Netnews Article Format. RFC 5536, Nov 2009.

[78] Nguyen, S., and Nguyen, T. Evaluation of Multipath TCP Load Sharing with Coupled Congestion Control Option in Heterogeneous Networks. In *Proc. IEEE Global Information Infrastructure Symposium* (2011).

[79] Nikkhah, M., Guérin, R., Lee, Y., and Woundy, R. Assessing IPv6 Through Web Access a Measurement Study and its Findings. In *Proc. ACM CoNEXT* (2011).

[80] Nordmark, E., and Bagnulo, M. Shim6: Level 3 Multihoming Shim Protocol for IPv6. RFC 5533, 2009.

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

[82] Obraczka, K., and Danzig, P. Evaluating the Performance of Flood-d: A Tool for Efficiently Replicating Internet Information Services. *IEEE J. Selected Areas in Communications 16* (1998).

[83] Odlyzko, A. M. Internet Traffic Growth: Sources and Implications. `http://www.dtc.umn.edu/mints/home.php`, 2003.

[84] Ohmori, M., Okamura, K., Tanizaki, F., and HAYAKAWA, K. Analyses on First Packet Drops of LISP in End-to-end Bidirectional Communications. In *Proc. Internet Conference* (2011).

[85] Pang, R., Paxson, V., Sommer, R., and Peterson, L. binpac: A yacc for Writing Application Protocol Parsers. In *Proc. ACM Internet Measurement Conference* (2006).

[86] Parchive. Parchive: Parity Archive Tool. `http://parchive.sourceforge.net/`.

[87] Paxson, V. Bro: a System for Detecting Network Intruders in Real-time. *Computer Networks 31* (1999).

[88] Petersen, K., Spreitzer, M., Terry, D., Theimer, M., and Demers, A. Flexible Update Propagation for Weakly Consistent Replication. In *ACM SIGOPS Operating Systems Review* (1997).

[89] Pittman, D., and GauthierDickey, C. A Measurement Study of Virtual Populations in Massively Multiplayer Online Games. In *Proc. ACM NetGames* (2007).

[90] Poese, I., Frank, B., Ager, B., Smaragdakis, G., and Feldmann, A. Improving Content Delivery Using Provider-aided Distance Information. In *Proc. ACM Internet Measurement Conference* (2010).

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

[92] Popa, L., Ghodsi, A., and Stoica, I. HTTP as the Narrow Waist of the Future Internet. In *Proc. ACM HotNets* (2010).

[93] Quoitin, B., Iannone, L., de Launois, C., and Bonaventure, O. Evaluating the Benefits of the Locator/Identifier Separation. In *Proc. Mobility in the Evolving Internet Architecture* (2007).

[94] Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and Handley, M. Improving Datacenter Performance and Robustness with Multipath TCP. In *Proc. ACM SIGCOMM* (2011).

[95] Raiciu, C., Handly, M., and Wischik, D. Coupled Congestion Control for Multipath Transport Protocols. RFC 6356, 2011.

[96] Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., and Handley, M. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *Proc. USENIX NSDI* (2012).

[97] Raicu, I., and Zeadally, S. Evaluating IPv4 to IPv6 Transition Mechanisms. In *Proc. IEEE International Conference on Telecommunications* (2003).

[98] Rao, A., Legout, A., Lim, Y., Towsley, D., Barakat, C., and Dabbous, W. Network Characteristics of Video Streaming Traffic. In *Proc. ACM CoNEXT* (2011).

[99] Ratnasamy, S., Handley, M., Karp, R., and Shenker, S. Topologically-aware Overlay Construction and Server Selection. In *Proc. IEEE INFOCOM* (2002).

[100] Saito, Y., and Shapiro, M. Optimistic Replication. *ACM Computing Surveys (CSUR) 37* (2005).

[101] Saltzer, J. On the Naming and Binding of Network Destinations. RFC 1498, 1993.

[102] Sarrar, N., Maier, G., Ager, B., Sommer, R., and Uhlig, S. Investigating IPv6 Traffic: What Happened at the World IPv6 Day? In *Proc. Passive and Active Measurement Conference* (2012).

[103] Saucez, D., Iannone, L., and Bonaventure, O. LISP Threats Analysis. `http://tools.ietf.org/html/draft-ietf-lisp-threats-08`, 2013.

[104] Schmidt, P., Enghardt, T., Khalili, R., and Feldmann, A. Socket Intents: Leveraging Application Awareness for Multi-Access Connectivity. In *Proc. ACM CoNEXT* (2013).

[105] Schulze, H., and Mochalski, K. ipoque Internet Study 2007. `http://www.ipoque.com/resources/internet-studies/` (need to register), 2007.

[106] Schulze, H., and Mochalski, K. ipoque Internet Study 2008/2009. `http://www.ipoque.com/resources/internet-studies/` (need to register), 2009.

[107] Seedorf, J., and Burger, E. Application-layer Traffic Optimization (ALTO) Problem Statement. RFC 5693, 2009.

[108] Smith, J., Klein, M., and Nelson, M. Repository replication using NNTP and SMTP. In *Research and Advanced Technology for Digital Libraries* (2006).

[109] Stewart, R. Stream Control Transmission Protocol. RFC 4960, Sep 2007.

[110] Suznjevic, M., Dobrijevic, O., and Matijasevic, M. MMORPG Player Actions: Network Performance, Session Patterns and Latency Requirements Analysis. *Multimedia Tools Appl. 45* (2009).

[111] Szabó, G., Veres, A., and Molnár, S. On the Impacts of Human Interactions in MMORPG Traffic. *Multimedia Tools Appl. 45* (2009).

[112] Tian, G., and Liu, Y. Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming. In *Proc. ACM CoNEXT* (2012).

[113] Triukose, S., Al-Qudah, Z., and Rabinovich, M. Content Delivery Networks: Protection or Threat? In *Proc. Computer Security–ESORICS* (2009).

[114] Varvello, M., Picconi, F., Diot, C., and Biersack, E. Is There Life in Second Life? In *Proc. ACM CoNEXT* (2008).

[115] Vasseur, J., Pickavet, M., and Demeester, P. *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Elsevier, 2004.

[116] Vogt, C. Six/One: A Solution for Routing and Addressing in IPv6. `http://tools.ietf.org/html/draft-vogt-rrg-six-one-02`, 2009.

[117] MultiPath TCP - Linux Kernel Implementation. `http://mptcp.info.ucl.ac.be/`.

[118] 802.1AX-2008 - IEEE Standard for Local and Metropolitan Area Networks– Link Aggregation. `http://standards.ieee.org/findstds/standard/802.1AX-2008.html`.

[119] APNIC's IPv4 Pool Usage. `http://www.apnic.net/community/ipv4-exhaustion/graphical-information`.

[120] Archive: 2011 World IPv6 Day. `http://www.internetsociety.org/ipv6/archive-2011-world-ipv6-day`.

[121] Avaya. `http://www.avaya.com/`.

[122] BGP Routing Table Analysis Report. `http://bgp.potaroo.net`.

[123] CIDR Report. `http://www.cidr-report.org/`.

[124] CISCO EtherChannel Technology. `http://www.cisco.com/en/US/tech/tk389/tk213/technologies_white_paper09186a0080092944.shtml`.

[125] Cisco LISP. `http://lisp.cisco.com/`.

[126] FlashGet. `http://www.flashget.com/`.

[127] Global IPv6 Deployment Progress Report. `http://bgp.he.net/ipv6-progress-report.cgi`.

[128] Google IPv6 Statistics. `http://www.google.com/ipv6/statistics.html`.

[129] Internet Connected Devices Approaching 10 Billion, to Exceed 28 Billion by 2020. `http://imsresearch.com/press-release/Internet_Connected_Devices_Approaching_10_Billion_to_exceed_28_Billion_by_2020`.

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

[131] JDownloader. `http://jdownloader.org/`.

[132] LISP-Lab. `http://lisplab.openlisp.org/consortium`.

[133] LISP Site Status. `http://www.lisp4.net/lisp-site/`.

[134] Locator/ID Separation Protocol (LISP) Working Goup. `http://datatracker.ietf.org/wg/lisp/charter/`.

[135] Port Aggregation Protocol. `http://www.ieee802.org/3/trunk_study/april98/finn_042898.pdf`.

[136] RouteViews Project. `http://www.routeviews.org`.

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

[138] sFlow Version 5. `http://www.sflow.org/sflow_version_5.txt`.

[139] SPDY: An Experimental Protocol for a Faster Web. `http://www.chromium.org/spdy/spdy-whitepaper`.

[140] TCPDUMP & LIBPCAP. `http://www.tcpdump.org/`.

[141] The Internet Society. `http://www.internetsociety.org`.

[142] The OpenLISP Project. `http://www.openlisp.org`.

[143] World IPv6 Launch. `http://www.worldipv6launch.org`.

[144] Total MMORPG Subscriptions and Active Accounts. `http://users.telenet.be/mmodata/Charts/TotalSubs.png`, 2010.

[145] World of Warcraft Surpasses 11 Million Subscribers Worldwide. `http://us.blizzard.com/en-us/company/press/pressreleases.html?id=2847881`, 2011.

[146] Zander, S., Andrew, L., Armitage, G., Huston, G., and Michaelson, G. Mitigating Sampling Error When Measuring Internet Client IPv6 Capabilities. In *Proceedings of the 2012 ACM conference on Internet measurement conference* (2012).

[147] Zhang, H., Chen, M., and Zhu, Y. Evaluating the Performance on ID/Loc Mapping. In *Proc. IEEE GLOBECOM* (2008).

[148] Zhang, M., Lai, J., Krishnamurthy, A., Peterson, L., and Wang, R. A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths. In *Proc. Usenix ATC* (2004).