# Use of Locator/Identifier Separation to Improve the Future Internet Routing System

Loránd Jakab

Advisor: Albert Cabellos-Aparicio, PhD

Co-Advisor: Prof. Jordi Domingo-Pascual, PhD

Department of Computer Architecture

Technical University of Catalonia

A Thesis presented to the Technical University of Catalonia in partial fulfillment of the requirements for the degree of

*Doctor in Computer Science*

July 2011

*To my parents*

# Acknowledgements

"When you have completed 95 percent of your journey, you are only halfway there", says a Japanese proverb. I certainly felt the paradoxical meaning of that nugget of wisdom during my long journey towards this destination. But having great people preparing me for the challenges ahead, or guiding me throughout, or simply being there when I needed them, made it both possible and worthwhile. The following paragraphs are a modest attempt to show my appreciation for that.

First and foremost I am most grateful to my advisor, Dr. Albert Cabellos-Aparicio, who's clear research vision and pragmatic advice contributed greatly to this thesis, and kept me motivated throughout. Thank you for the countless brainstorming and followup meetings, our collaboration was a great experience for me.

I would also like to thank my co-advisor, Prof. Jordi Domingo-Pascual, for the patient guidance and continuous support I received during the development of this thesis, both professional and personal.

It was a great privilege for me to work together with outstanding researchers. One of the most exciting time for me during the elaboration of this thesis was the collaboration with Prof. Olivier Bonaventure, Damien Saucez and Dr. Pierre Françoise from Université catholique de Louvain; discussing results, changing simulation parameters, rethinking architectures, sometimes late at night, was an inspiring and motivating experience. Florin Coraş was an undergrad student doing his diploma work with us during this period, but he was an equal footing with everyone else in terms of contribution value. Dr. Thomas Silverston provided data and insight into the inner workings of peer-to-peer live streaming systems, a key ingredient of my work on next generation live streaming. The discussions I had with my professors and colleagues at UPC, Prof. Josep Solé-Pareta, Dr. Pere Barlet, Dr. René Serral, and Josep Sanjuàs, were also very helpful.

While working on the thesis, I enjoyed the full support of the great administrative staff, but there is one person whom I am most indebted to: Trinidad Carneros. She is our guiding light in the maze of bureaucracy, who we all rely on to find the best solution to our administrative problems. Thank you for your patience and hard work! José Núñez and Albert López, along with the LCAC staff, were also key people for this work, as they provided access and support to all the necessary laboratory resources to carry out experiments and analyze their results.

Dr. Luigi Iannone was kind enough to welcome me as visiting researcher to Deutsche Telekom Laboratories, enabling me to interact with some of Europe's best networking experts, and collaborating with him on the open source implementation of LISP. Thank you for being a great host those four months, and also for showing me the ropes at IETF meetings! I'm also grateful to the LISPers, Dino Farinacci, Dave Meyer, Vince Fuller, and Darrel Lewis for creating a great protocol, that provided some of the research challenges in this thesis.

A big thank you to all the teachers who helped me prepare for this journey, long before I knew I will embark on it. They showed me how to organize and apply the knowledge I gained, and most importantly, how to think. To my elementary teacher, Czirmay Irén, who built a solid foundation. To my math teacher during the following 8 years, Ferencz Sára, who's passion for her discipline certainly made learning a fun experience. To the great scholars at the Technical University of Cluj Napoca, who turned me into a telecommunications engineer. I'm especially grateful to Prof. Vasile Bota , Prof. Mircea Dăbâcan, Dr. Daniel Zinca, and Dr. Romulus Terebeş, who's deep knowledge and attention to detail was inspiring and made me look forward to their lectures; and to my mentor at this institution, Prof. Virgil Dobrotă, who guided my diploma work and paved the way for my graduate studies.

Most of the work in this thesis used open source software developed by volunteers, for which I am grateful, and I do my best to contribute back to the community.

I would also like to mention Randall Munroe and Jorge Cham, the names behind `xkcd` and *PhD Comics* respectively, for all the geek humor that PhD candidates can identify with.

Several organizations provided very important financial support. First of all, this thesis would not have been possible without the generous support of the Generalitat de Catalunya through the 2006FI-00935 grant. Additionally, the CONTENT (Excellence in Content Distribution Network Research: FP6-0384239) and EuQoS (End-to-end Quality of Service support over heterogeneous networks: FP6-004503) European projects and the European Doctoral School of Advanced Topics In Networking (SATIN) contributed significantly as well.

On a more personal note, I could never have done this without the support of my friends and family, who often had to put up with my moods and lack of presence. Finally, this work is dedicated to my parents, who made the sacrifices necessary for me to have access to good education and technology, no matter the circumstances. *Köszönöm szépen!*

# Abstract

The Internet evolved from its early days of being a small research network to become a critical infrastructure many organizations and individuals rely on. One dimension of this evolution is the continuous growth of the number of participants in the network, far beyond what the initial designers had in mind. While it does work today, it is widely believed that the current design of the global routing system cannot scale to accommodate future challenges.

In 2006 an Internet Architecture Board (IAB) workshop was held to develop a shared understanding of the Internet routing system scalability issues faced by the large backbone operators. The participants documented in RFC 4984 their belief that "routing scalability is the most important problem facing the Internet today and must be solved."

A potential solution to the routing scalability problem is ending the semantic overloading of Internet addresses, by separating node location from identity. Several proposals exist to apply this idea to current Internet addressing, among which the Locator/Identifier Separation Protocol (LISP) is the only one already being shipped in production routers. Separating locators from identifiers results in another level of indirection, and introduces a new problem: how to determine location, when the identity is known.

The first part of our work analyzes existing proposals for systems that map identifiers to locators and proposes an alternative system, within the LISP ecosystem. We created a large-scale Internet topology simulator and used it to compare the performance of three mapping systems: LISP-DHT, LISP+ALT and the proposed LISP-TREE. We analyzed and contrasted their architectural properties as well.

The monitoring projects that supplied Internet routing table growth data over a large timespan inspired us to create LISPmon, a monitoring platform aimed at collecting, storing and presenting data gathered from the LISP pilot network, early in the deployment of the LISP protocol. The project web site and collected data is publicly available and will assist researchers in studying the evolution of the LISP mapping system.

We also document how the newly introduced LISP network elements fit into the current Internet, advantages and disadvantages of different deployment options, and how the proposed transition mechanism scenarios could affect the evolution of the global routing system. This work is currently available as an active Internet Engineering Task Force (IETF) Internet Draft.

The second part looks at the problem of efficient one-to-many communications, assuming a routing system that implements the above mentioned locator/identifier split paradigm. We propose a network layer protocol for efficient live streaming. It is incrementally deployable, with changes required only in the same border routers that should be upgraded to support locator/identifier separation. Our proof-of-concept Linux kernel implementation shows the feasibility of the protocol, and our comparison to popular peer-to-peer live streaming systems indicates important savings in inter-domain traffic.

We believe LISP has considerable potential of getting adopted, and an important aspect of this work is how it might contribute towards a better mapping system design, by showing the weaknesses of current favorites and proposing alternatives. The presented results are an important step forward in addressing the routing scalability problem described in RFC 4984, and improving the delivery of live streaming video over the Internet.

# Resumen

La Internet evolucionó desde ser una pequeña red experimental en sus inicios hasta llegar a ser una infraestructura crítica de la que muchas organizaciones y personas dependen. Una dimensión de su evolución es el incremento continuo del número de participantes en la red, mucho más allá de lo que los diseñadores iniciales habían imaginado. Aunque siga funcionando hoy en día, está ampliamente aceptado que el diseño actual del sistema global de enrutamiento no puede escalar para acomodar los desafíos del futuro.

En 2006 se organizó un taller del Internet Architecture Board (IAB) para alcanzar una comprensión mutua de los problemas de escalabilidad del sistema de enrutamiento en Internet a los que se enfrentan los grandes operadores troncales. Los participantes documentaron en el RFC 4984 su convicción de que "la escalabilidad del enrutamiento es el problema más importante de la Internet hoy en día que se ha de solucionar."

Una posible solución al problema de escalabilidad del enrutamiento es eliminar la sobrecarga semántica de las direcciones de Internet, separando la localización de los nodos de su identidad. Varias propuestas aplican esta idea al direccionamiento de la Internet actual. De entre ellas el Locator/Identifier Separation Protocol (LISP) es la única disponible para routers de producción. La separación de los localizadores de los identificadores resulta en un nivel adicional de indirección, e introduce un nuevo problema: cómo determinar la localización cuando se conoce la identidad.

La primera parte de nuestro trabajo analiza las propuestas existentes de sistemas que mapean identificadores a localizadores y propone un sistema alternativo dentro del ecosistema LISP. Hemos creado un simulador de topología Internet de gran escala y lo hemos utilizado para comparar el rendimiento de tres sistemas de mapeo: LISP-DHT, LISP+ALT y el propuesto LISP-TREE. También hemos analizado y contrastado sus propiedades arquitecturales.

Los proyectos de monitorización que han proporcionado datos importantes sobre el crecimiento de las tablas de enrutamiento durante un largo intervalo de tiempo nos ha inspirado a crear LISPmon, una plataforma de monitorización con el fin de coleccionar, guardar y presentar datos recogidos en la red piloto LISP, primer despliegue del protocolo LISP. La web del proyecto y los datos recogidos son públicos y ayudarán a los investigadores en estudiar la evolución del sistema de mapeo LISP.

Documentamos también cómo encajan los nuevos elementos de red introducidos por LISP en la Internet actual, las ventajas y desventajas de las diferentes opciones de despliegue, y cómo los mecanismos de transición propuestos pueden afectar la evolución del sistema global de enrutamiento. Este trabajo está actualmente disponible como un draft activo en la Internet Engineering Task Force (IETF).

La segunda parte aborda el problema de las comunicaciones eficientes uno-a-muchos, asumiendo un sistema de enrutamiento que implementa el paradigma locator/identifier split mencionado antes. Proponemos un protocolo a nivel de red para streaming eficiente. Es desplegable de manera incremental, con cambios necesarios solo en los mismos routers borde que se actualizarán para soportar la separación localizador/identificador. Nuestra implementación prueba de concepto para el kernel de Linux muestra la viabilidad de nuestro protocolo, y la comparación con sistemas de streaming en vivo peer-to-peer populares presenta ahorros importantes del trafico inter-dominio.

En nuestra opinión LISP tiene un potencial importante para ser adoptado, y un aspecto importante de este trabajo es como puede contribuir a un mejor diseo de sistema de mapeo, mostrando las debilidades de los sistemas favoritos actuales y proponiendo alternativas. Los resultados presentados son un paso importante para hacer frente al problema de escalabilidad del enrutamiento descrito en el RFC 4984, y mejorar el servicio de streaming video en vivo en Internet.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **AAA** | Authentication, Authorization, and Accounting |
| **API** | Application Programming Interface |
| **CDN** | Content Delivery Network |
| **CSV** | Comma Separated Values |
| **DFZ** | Default-free Zone |
| **DHT** | Distributed Hash Table |
| **DNS** | Domain Name System |
| **EID** | Endhost IDentifier |
| **ETR** | Egress Tunnel Router |
| **FIB** | Forwarding Information Base |
| **GSE** | Global, Site, and End-system |
| **GUI** | Graphical User Interface |
| **IAB** | Internet Architecture Board |
| **IETF** | Internet Engineering Task Force |
| **IP** | Internet Protocol |
| **IPv4** | Internet Protocol version 4 |
| **IPv6** | Internet Protocol version 6 |
| **IRTF** | Internet Research Task Force |
| **ITR** | Ingress Tunnel Router |
| **LIG** | LISP Internet Groper |
| **LISP** | Locator/Identifier Separation Protocol |

## LIST OF ACRONYMS

| | |
|---|---|
| **LPN** | LISP Pilot Network |
| **LTS** | LISP-TREE Server |
| **MR** | Map-Resolver |
| **MS** | Map Server |
| **MTU** | Maximum Transmission Unit |
| **NAT** | Network Address Translation |
| **P2P** | Peer-to-peer |
| **PA** | Provider-aggregatable |
| **PI** | Provider-independent |
| **PoP** | Point of Presence |
| **QoE** | Quality of Experience |
| **QoS** | Quality of Service |
| **RDBMS** | Relational DataBase Management System |
| **rDNS** | Reverse DNS |
| **RFC** | Request for Comments |
| **RIB** | Routing Information Base |
| **RIR** | Regional Internet Registry |
| **RLOC** | Routing LOCator |
| **RRG** | Routing Research Group |
| **RTT** | Round-Trip Time |
| **TCP** | Transmission Control Protocol |
| **TTL** | Time-to-Live |
| **UCL** | Université catholique de Louvain |
| **UDP** | User Datagram Protocol |
| **UPC** | Universitat Politècnica de Catalunya |
| **xTR** | Tunnel Router |

# 1

# Introduction

The Internet is a great invention, which since its inception never stopped evolving, growing to a size and popularity not imagined by its initial designers. The question of how far will the original design continue to scale puzzled researchers and the operational community since the Internet became a commercial success and growed beyond expectations. Concerns about its scalability increased in recent years [67], both due to growing routing table size and dynamics, and live video streaming traffic. This chapter introduces some of the current challenges of the scalability of the Internet routing system, presenting the motivation behind this work, and lays out the contributions brought to the advancement of the networking field.

## 1.1 Motivation

We begin by exploring the current understanding of the problems faced by today's Internet, to explain the motivations behind this work. The next section sets the context for the following chapters, by demonstrating the global routing table scalability problem, presenting some of the solutions proposed to date, and asking the specific questions that will later be answered. Section 1.1.2 dives into the problems faced by live video streaming within the current Internet architecture, and hints at the window of opportunity created by solving the first issue, to address those problems.

### 1.1.1 Locator/Identifier Split and Routing Scalability

The continuous monitoring of the the IPv4 default-free zone (DFZ) routing table size by Geoff Huston dating back to 1994 [42] reveals superlinear growth, which, together with the tables' increased dynamics, raised concerns over the long term scalability of the Internet routing system. To improve the understanding of the issues surrounding this problem, the Internet Architecture Board organized a workshop in November 2006, and published its findings in RFC 4984 [67]. *"Routing scalability is the most important problem facing the Internet today and must be solved"* was one of the key conclusions of the document.

Opinions on how to solve this problem, and in general, how to improve the Internet architecture, are divided into two camps: one side prefers short-term incremental fixes, while the other a long-term major overhaul of the existing design.

In a point/counterpoint article [72], Rexford advocates for a clean-slate design for the future Internet architecture, while Dovrolis sides with the more evolutionary approaches. Quoting American baseball legend Yogi Berra –"You've got to be very careful if you don't know where you're going, because you might not get there." –, the former argues that clean-slate research can produce valuable designs to set long-term targets for the more evolutionary approaches, which are acknowledged as equally useful. The latter mostly dismisses clean-slate, reasoning that the large deployed base of today's Internet calls for improvements that are backwards compatible and incrementally deployable. The reader is left leaning towards the evolutionary approach, which is more likely to lead to deployment, but without dismissing clear directions set by long-term goals. In the following, we analyze the causes of the routing scalability problem, pinpoint one of the long-term goals set by clean-slate research, but advocate for a less disruptive, incremental, solution.

Routing table growth is driven by several causes beyond the organic growth of the Internet. In order to avoid *renumbering*, many sites prefer provider-independent (PI) addresses. Since this address space is not covered by their provider's prefixes, it has to be announced separately, thus increasing routing table size. For better redundancy *multihoming* is a common networking practice. This is an additional reason for which PI addresses are preferred by end sites. While using provider-aggregatable (PA) addresses to achieve multihoming is feasible and is sometimes done, it leads to routing table increase nonetheless, since a more specific prefix has to be announced from the other provider(s) (i.e., not the PA space owner). *Traffic engineering* is another driving force behind DFZ growth, because it leads to prefix deaggregation. To better control incoming traffic, end sites announce more specifics on different links. In some cases, the more specifics serve another purpose: to avoid prefix hijacking, thus improving *security*.

A quote by Yakov Rekhter, often referred to as “Rekhter's Law”, sums up the routing scalability dilemma quite well: “Addressing can follow topology or topology can follow addressing. Choose one.” That is, the way to keep routing table size down is by means of topological aggregation of addresses. In practice, however, the need for PI addressing to avoid renumbering and enable multihoming precludes aggregation. Multihoming is unavoidable for networks with business critical nodes, because a multihomed network can be reached through more than one provider network, greatly improving availability.

The projected exhaustion of the IANA IPv4 address pool for early 2011 [43] (which occurred on February 3) boosted IPv6 deployment [13], with some important carriers either fully supporting the protocol [4] or already running trials to do so [2]. The migration path to IPv6 adopted so far by transit network providers is a *dual stack* strategy: running both IPv4 and IPv6 stacks on all routers. This means that in addition to the IPv4 routing table, routers maintain an IPv6 routing table at the same time, requiring additional fast memory capacity. On top of this, per-entry memory use is higher in IPv6 than in IPv4. RFC 4984 discusses why relying on Moore's law (i.e., transistor density on silicon doubles every 18-24 months) is not a solution for keeping up with table size growth.

For all the above reasons, solutions for slowing down table growth, or, if possible, decreasing table sizes is an important goal of the Internet operational community.

One approach to lower routing table memory requirements is to aggregate entries that point to the same output interface, whenever possible. Algorithms proposed in this solution space [19, 22, 59] can achieve up to an order of magnitude size reduction, but every routing update will trigger a rerun of the algorithm in use. On the downside, these algorithms impose extra load on the routers, and a routing table update rate higher than the algorithm execution time leads to 100% CPU resource consumption.

Even without these reduction techniques in today's BGP routers, table update frequency (BGP churn) is a major concern [31], because it leads to increased CPU usage and to routing convergence times on the order of several minutes on average [50]. During that time, the prefixes that have been updated may not have global reachability.

Another approach to reduce routing table sizes is to improve aggregation. One of the first attempts was the Global, Site, and End-system (GSE) architecture by Mike O'Dell [70], which divides an IPv6 address into a routing goop, site topology partition and end system designator. It eliminates the need for PI, because of the hierarchical addressing scheme. Multihoming is achieved by using a site local routing goop withing the site, and choosing a topologically aggregatable one at the site border, based on the upstream provider. Similarly, Chiappa's Nimrod architecture [25] imagined a hierarchy of network clusters, leading to better aggregation. Hinden's ENCAPS protocol [40] uses dynamic IP tunneling at border routers, encapsulating packets from edge networks using aggregatable address space reserved for core routing. While these proposals had several issues [91] and did not see deployment, the main idea of a core network with aggregatable addressing and edge networks in a different numbering space set a long term goal in Internet architecture design.

Endpoints in the current Internet are assigned an IP address, that is used to determine both *who* and *where* they are. This semantic overloading of the IP address, which acts both as an *identifier* and a *locator*, contributes significantly to the routing scalability problem [67]. Because of this, many proposed solutions are based on the separation of locator information from endpoint identity.

After the publication of RFC 4984, the Routing Research Group (RRG) of the Internet Research Task Force (IRTF) was rechartered to address the issues presented in the document and outlined the design goals of a new Internet architecture in RFC 6227 [57]. Several proposals promising to achieve those goals were published and continuously improved by group participants over the years [14, 18, 32, 34, 65, 79, 80, 83, 85, 90], but no consensus was reached over the best choice. However, there was "rough consensus that separating identity from location is desirable and technically feasible", as documented in RFC 6115 [58]. Massey *et al.* advocate for this approach as well, showing both benefits and challenges [63], while Quoitin *et al.* show [71] by simulations (assuming LISP deployment) that such a mechanism helps in significantly reducing the size of the FIB of core Internet routers, because locators are assigned hierarchically.

Introducing an additional level of indirection in the Internet architecture is one of the major challenges faced by approaches based on the locator/identifier split idea [63], and raises the question: “Will it scale? Or does it introduce more problems than it solves?” It is towards answering these questions that the first part of this work embarks.

### 1.1.2 Locator/Identifier Split and Live Streaming

In addition to control plane scalability issues, the increasing live streaming traffic volumes [74] raise data plane scalability concerns. The most efficient delivery method for IPTV’s one-to-many traffic model is network-layer multicast (IP multicast [28, 29]). Due to its complex network management, resulting in high capital and operational expenditure, *inter-domain* multicast has not taken off [30].

The common approach to transmit live content over the Internet is traditional unicast. In the unicast model, each destination will receive the requested stream over a separate connection. As a result, the same content is transmitted over the network several times, wasting capacity.

Since unicast suffers from severe scalability issues, it is often used in combination with Content Delivery Networks (CDNs) [1, 3, 5, 6]. These networks deploy geographically distributed servers in the Internet, replicating the multimedia content. This content is usually distributed using P2P networks [17]. Finally, the users receive the content from the closest copy. This approach can be considered as a hybrid solution, where first the content is distributed using an overlay (application-layer), and then the users access this content directly using unicast (network-layer). Still, CDN’s focus on performance does not address the fundamental issue of efficiency in the network, and is a costly option for the content provider.

During the last years, the industry along with the research community have designed several peer-to-peer (P2P) systems for streaming live events [8, 9, 11, 60, 92] to an ever increasing audience in a more efficient way. These systems operate at the application-layer, and create complex overlay networks to distribute multimedia content. These popular applications are used daily by millions of Internet users [38]. Yet in practice, these live streaming systems fail to provide a satisfactory quality of experience at all times. As a consequence, the research community has thoroughly analyzed the streaming quality of these large-scale P2P networks in search for explanations [16, 38, 39, 55, 56, 76, 77]. Surprisingly, one of the main findings is that the streaming quality *degrades* as the number of peers *increases* if peak hours are considered [87]. Further, server capacity still plays a key role in these systems, and its insufficient supply leads to low streaming quality [87]. The main reasons behind these inefficiencies are the limited upload capacity of the peers, who usually access the Internet through asymmetric links, and churn (peers may join/leave the system at any moment). Because of these reasons, application layer-based systems cannot guarantee a reliable streaming service, and cannot be considered as the long-term solution for broadcasting live multimedia content over the Internet.

The second part of this work, inspired by the locator/identifier separation paradigm, reevaluates one-to-many communications at the network layer, proposing and evaluating a solution.

## 1.2 Thesis Contributions

### 1.2.1 Mapping Systems

LISP is the most advanced locator/identifier separation proposal to date, having a complete implementation for the Cisco NX-OS and IOS platforms[1] and an interoperable open source implementation for the FreeBSD operating system [47], called OpenLISP. A global testbed gives operational feedback to the protocol authors, enabling improvements rooted in hands-on experience. Due to these practical considerations, and the potentially higher impact, we chose LISP as the locator/identifier separation solution to study.

Several mapping systems were already proposed for LISP [23, 36, 52, 64] before this work started. LISP+ALT [36] was chosen for the testbed, not least because of its ease of implementation on routers. While useful, evaluating LISP+ALT in a testbed with about 50 participants cannot reveal its scalability properties. Moreover, the decision of the mapping system of the future Internet should not be determined based on ease of implementation. But comparing the different proposals objectively is challenging, and no tools were available.

Running experimental testbeds to reproduce the dynamics of the Internet Autonomous System (AS) topology is a difficult task, due to economical considerations. Virtualization lowered the barrier to deploy the approximately necessary 30,000 nodes, but we found only one project attempting to do that, with limited success [68]. Even

[1] `http://lisp4.cisco.com/`

if that number of nodes can be created, the management of the experiments is still a difficult task.

Simulation is the other choice. The open source ns-2, ns-3, and OMNeT++ simulators are widely available for networking simulations, while OPNET and NetSim are popular commercial packages. However, they are packet-level event based simulators, which makes large scale simulations challenging due to the high resource requirements. For our purposes, only the first few packets of a flow affect simulation results, but the topology must be Internet-scale. Thus, a different simulation approach is needed.

This is the first challenge addressed by this work. We developed CoreSim, a hybrid trace and event-based Internet-scale simulator with a modular design, allowing to add different mapping system backends.

Using CoreSim, we compared LISP+ALT and LISP-DHT performance, exposing the the shortcomings of these mapping system proposals, showing the need for a more scalable mapping system. This second challenge was addressed by contributing to the design and then evaluating a new mapping system proposal from the LISP team led by professor Olivier Bonaventure at Université catholique de Louvain: *LISP-TREE*.

LISP-TREE is based on the widely used Domain Name System (DNS), with a similar hierarchical organization. Blocks of EIDs are assigned to the layers of the naming hierarchy by following the current allocation rules for IP addresses. The root of the naming hierarchy is maintained by the Regional EID Registries, which allocate EID blocks to local registries. These in turn maintain delegation information for the owners of the provider independent EID prefixes. Levels can be dynamically added to the hierarchy. LISP-TREE nodes can use existing DNS implementations, and benefit from the long operational experience, but to avoid interference with the current domain name system, LISP-TREE should be deployed on a *physically* separate infrastructure. One of the main advantages of LISP-TREE is DNS' proven scalability track record: for instance, at the end of 2010, the *.com* top-level domain stored an estimated 88 million entries [12], while there are only around 350,000 entries in the default-free zone today [42].

The above work resulted in several publications: the CoreSim simulator was presented at the 2009 Trilogy Future Internet Summer School poster session; the talk *"Evaluation of LISP+ALT performance"* was given at the 75th IETF Meeting in Stockholm about the comparison of LISP+ALT and LISP-DHT; and the presentation of

LISP-TREE and the detailed evaluation of all three mapping system was published in the IEEE Journal on Selected Areas in Communications, special issue on Internet routing scalability, with the title *"LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System"*.

### 1.2.2 Live Streaming

As we showed above, there is consensus in the Internet research community that a locator/identifier separation-based approach is needed for future routing scalability. We also argued that practical reasons determined us to choose the LISP protocol from the existing proposals to base our work on. This is true for the second part of our work as well, which focuses on data plane scalability concerns.

In the second part we describe CoreCast, a network-layer live streaming protocol, designed to exploit the window of opportunity offered by the likely evolution of the Internet to LISP. Unlike IP multicast, which requires support on all routers on the data path, CoreCast changes only the same autonomous system border routers that are upgraded to support LISP.

While IP multicast is the most efficient way to deliver live streaming traffic, it failed to take off for ubiquitous inter-domain configurations, as we showed above. The current alternative is either the least efficient unicast, or P2P platforms. To quantify the traffic savings of CoreCast, we compare it to these application-layer solutions, both analytically and based on measurements.

For the measurement based comparison we capture and analyze P2P-TV traffic in four different countries (Japan, France, Spain and Romania) during popular live events. The results give insight on the distribution of clients in autonomous systems, a property which important to the CoreCast protocol.

We analyzed the feasibility of CoreCast to be included on routers by implementing a proof-of-concept for the Linux kernel. This implementation was used to test processing overhead due to CoreCast at Gigabit speeds on commodity hardware. Considering that some operations can be offloaded to specialized ASICs, performance impact on real world productions routers would be even smaller that what our experiments show. The proof-of-concept code and a protocol dissector for Wireshark were released to the public[1].

[1] `http://www.cba.upc.edu/corecast`

This work was published in its early stages in the ACM SIGCOMM 2009 poster session, and once reaching maturity in the Elsevier Computer Networks journal, under the title *"CoreCast: How Core/Edge Separation Can Help Improving Inter-Domain Live Streaming"*. Part of the obtained results were cross-pollinated to the Elsevier Signal Processing: Image Communication journal, as *"Large-scale measurement experiments of P2P-TV systems. Insights on fairness and locality"*.

### 1.2.3 Contributions to the LISP Community

#### 1.2.3.1 LISP Deployment Internet-draft

As active LISP working group participants in the IETF, we presented our mapping system analysis in the July 2009 meeting, participated in mailing list discussions, and are currently working on an Internet-draft. Internet-drafts are working documents of the IETF, made available for informal review and comment within a certain working group and the wider community.

The LISP working group had several discussions about how the expected deployment scenarios may affect the protocol design. Margaret Wasserman and Darrel Lewis gave a talk at the 76th IETF meeting about a few deployment scenarios [54], but there was no formal document to list the scenarios LISP should be targeted to. Our Internet-draft, based on that talk, is a work-in-progress to fill this vacuum, and was presented at the 79th IETF meeting in November 2010, where it received positive feedback. A draft goes over several revisions, until working group participants reach "rough consensus" on its content, at which point it can be published as a Request for Comments (RFC). We are working closely with the LISP working group to reach this important milestone for our work.

Our Internet-draft, "LISP Network Element Deployment Considerations", describes the possible deployment locations of the new network elements introduced by the LISP protocol, and the consequences of deploying in a particular location on client networks, service providers, and, in some cases, the Internet at large. A significant part is dedicated to the transition from the non-LISP Internet to a LISP-enabled one. Having a clear upgrade path is critical for any new protocol that wishes to improve upon (or replace) an existing popular protocol. LISP enhances the current addressing architecture of the Internet routing system, which has changed very little since its original design.

It does so in a fully backwards compatible way, but there are still some issues to be discussed with regard to the interoperation of non-LISP and LISP sites. The draft documents the plausible scenarios for the deployment of the network elements necessary for the transition, and offers perspectives on the deployment incentives.

#### 1.2.3.2 LISPmon

To gain operational experience with the protocol, the LISP early adopters built a testbed in the current Internet. This testing ground uses the LISP+ALT mapping system, the only one that has been implemented to date. There are many aspects that are interesting from an operational point of view, and monitoring the testbed is a good way to study them.

These early adopters are most likely interested in the stability and maturity of the code, but also in tuning protocol parameters, finding corner cases, and so on. Monitoring the testbed is of vital importance to gather the necessary performance and other statistical data.

To date, only one project exists that provides *public* LISP testbed monitoring data[1], maintained by one of the testbed participants. However, this project needs manual configuration and maintenance, which is not scalable. LISPmon aims to provide a scalable design, by automatic discovery of testbed elements. Additionally, it monitors a greater number of parameters, stored in a flexible way, to enable future creative use of old historical data. The gathered live data is presented in an interactive graphical interface, while historical data is provided in computer readable format.

#### 1.2.3.3 LISP Protocol Dissector

Once the specification of a network protocol is mostly stable, the implementation phase can begin. To debug the code, it is necessary to capture the packets produced by the implementation. While it is possible to manually check protocol field offsets in the captured data, debugging is more productive when a network protocol analyzer is available. This software is able to recognize different protocol fields in network packets and display their name and value in a human readable format.

[1]`http://smokeping.lisp.intouch.net/cgi-bin/smokeping.cgi`

Taking advantage of the modular design of the most popular open source packet analyzer, Wireshark, we implemented a packet dissector for both LISP control packets and the encapsulation header of data packets. The implementation is public[1], and feedback received shows that it is widely used in the LISP development community.

## 1.3 Thesis Structure

**Chapter 2** provides the necessary background on the Locator/Identifier Protocol (LISP) and the existing mapping systems, to facilitate understanding of the following material.

**Part I** is dedicated to the study of the LISP control plane, specifically the LISP mapping systems. **Chapter 3** presents LISP-TREE, our proposed mapping system that is based on the DNS protocol. The chapter begins with the advantages of using DNS, describes the protocol itself, and how it should be deployed.

The *CoreSim* Internet topology simulator is presented in **Chapter 4**. CoreSim was built to simulate the effects of migrating the current IPv4 Internet to LISP, and to estimate some of the performance metrics of existing mapping system proposals.

In **Chapter 5**, two experimental datasets (traffic traces) are introduced, after which the chapter is devoted to the thorough analysis of simulation results using the LISP+ALT, LISP-DHT, and LISP-TREE *CoreSim* modules. After analyzing the cache miss rate, the mapping lookup latency, and the node load for each mapping system, they are compared based on operational considerations as well.

**Part II** begins with an overview of live streaming systems in **Chapter 6**, based on existing unicast, network-layer multicast and application-layer multicast delivery methods. This chapter then describes *CoreCast*, our novel network-layer multicast live streaming method, exploiting the window of opportunity offered by the likely deployment of LISP.

In **Chapter 7** the savings in terms of inter-domain traffic compared to application layer approaches are quantified using both analytical and metrological methods, including P2P live streaming measurements from different vantage points. The feasibility of CoreCast is demonstrated using a prototype implementation.

[1] `http://lisp.cba.upc.edu/wireshark/`

**Part III** presents contributions to the LISP community, including an IETF Internet Draft discussing the deployment scenarios of new LISP network elements in **Chapter 8**, a the LISPmon monitoring platform to gather and visualize statistics about the LISP pilot network in **Chapter 9**, and the open source Wireshark dissector plugin to decode LISP data and control packets in **Chapter 10**.

Finally, **Chapter 11** concludes the thesis, and offers perspectives for future work.

# 2

# Background

## 2.1 The Locator/Identifier Separation Protocol

LISP is one of the proposals implementing the locator/identifier separation idea. IP addresses (both version 4 and version 6) are replaced by the syntactically identical new numbers: Routing LOCators (RLOCs) or Endpoint IDentifiers (EIDs). EIDs are assigned to devices in edge networks independently of the network topology, while RLOCs follow topology, and are used for transit in the Internet.

There are two main design decisions each locator/identifier split proposal has to take:

1. the topological location of the boundary where locator usage ends and identifier usage begins;

2. the method employed to get identifier-based packets across the locator-based networks.

The first design decision was to use administrative boundaries (such as the border routers of stub autonomous systems) as the delimiter between identifier and locator space. The main advantage of this choice is that only a small number of Internet devices have to be upgraded to support the new protocol, as opposed to a host-based boundary, where all endpoints are required to change. All other nodes continue to operate as they do today, without changes to their protocol stacks, making transition less costly.

For the second design decision, the choice had to be made between *map-and-encap* and *address rewriting*. The former uses encapsulation for transporting the original packets across locator space, after a mapping function determines the RLOC of the destination EID (see Section 2.2 for more details on mapping functions). The latter rewrites the source and destination fields in the packet headers at the identifier/locator and locator/identifier boundaries, and is a mechanism similar to Network Address Translation (NAT). LISP employs map-and-encap, which allows mixing any combination of address families for EIDs and RLOCs and preserves the original source EID of packets (and conforms to the end-to-end model). It's main disadvantage is the tunneling overhead, which may lead to path maximum transmission unit (MTU) issues.

To do the map-and-encap operation, LISP introduces two new functional units: the Ingress Tunnel Router (ITR) and the Egress Tunnel Router (ETR). They terminate the ends of a unidirectional tunnel, connecting to EID domains. Since with few exceptions bidirectional operation is desired, the two functional units can be combined, creating a Tunnel Router (xTR). These functional units are deployed as software updates to the routers on the administrative boundaries discussed above, or the routers themselves are replaced by new hardware with this functionality built-in, in order to enable a site to use EIDs.

As the name suggests, the *Ingress* Tunnel Router is the tunnel ingress point, where packets are encapsulated. EID-to-RLOC mappings are used to determine the *Egress* Tunnel Router that will be the exit point of the tunnel. The destination EID of packets is examined and used as a lookup key in a distributed mapping database to determine a set of candidate ETRs. To discriminate between the RLOCs returned by the distributed mapping database, LISP defines *priorities* and *weights*, which are associated to each RLOC in the mapping.

First, the RLOCs with the lowest priority value are selected, eliminating the unreachable ones from that list. Next, packets are load-balanced over the remaining RLOCs, based on the weights specified for each.

Finally, the ETR checks if it is responsible for the destination EID of the encapsulated packet, then decapsulates and forwards it to the intended endpoint.

Consider the example simple LISP scenario from Figure 2.1. The bottom site (that initiates the communication in our example) has two xTRs: `R1` and `R2`. In this example, we use IPv6 addresses as end host identifiers (EIDs) and IPv4 addresses as RLOCs.

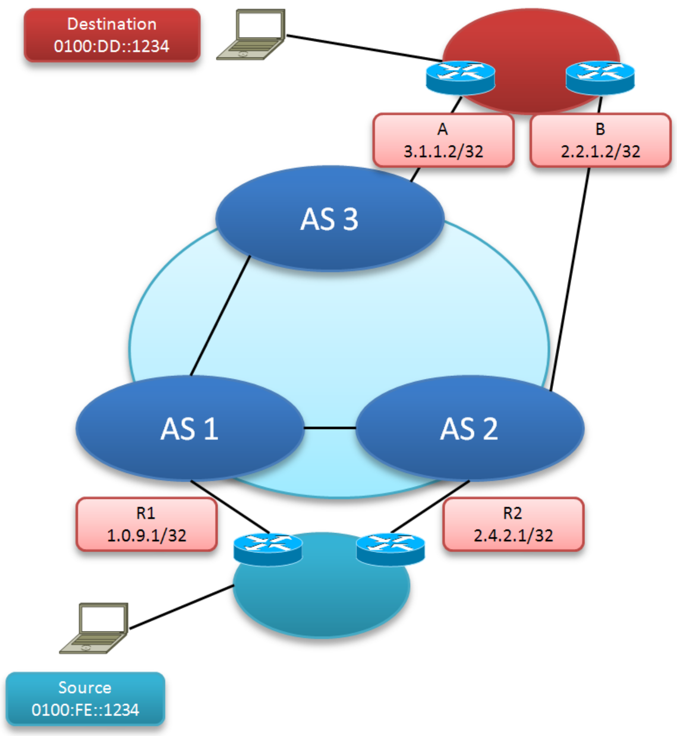


**Figure 2.1:** Simple LISP scenario

When an end host in the bottom LISP site, e.g. `0100:FE::1234`, needs to contact a remote end host, e.g. `0100:DD::1234`, it sends a normal (IPv6 in this case) packet with the destination EID as destination address. This packet reaches one of the site's ITRs (i.e., R1 or R2). To forward the packet, the ITR needs to obtain at least one of the RLOCs of the destination ETR. For this, the ITR queries the mapping system by sending a Map-Request LISP control packet.

Several mapping systems [23, 36, 52, 64] are being developed for LISP (see next section). The mapping system will respond with a Map-Reply control packet which contains a set of RLOCs of the ETRs which the ITR can use to forward packets to the destination. A priority and a weight are associated with each RLOC to allow a LISP site to control its ingress traffic. RLOCs with the lowest priority value are preferred. Load balancing between the same priority RLOCs is achieved with the weight. A time-to-live (TTL) is also associated to these mappings, indicating their maximum lifetime in a cache. A mapping can be evicted from a cache before its TTL expires, if so required by the particular cache's eviction policy, but must be either refreshed or deleted when its maximum lifetime is reached. Furthermore, the mapping system will usually return a mapping that is valid for the entire prefix containing the destination EID. For example, in Figure 2.1, the mapping system could return a mapping that associates `0100:DD/48` to `3.1.1.2` and `2.2.1.2`. Once the mapping has been received, it is installed in the mapping cache of the ITR and the packet is encapsulated and sent to the RLOC of the destination ETR. The destination ETR decapsulates the packet and forwards it to the destination end host. Subsequent packets sent to this EID will be forwarded based on the cached mapping.

## 2.2 LISP Mapping Systems

The mapping system is a major component of the Locator/Identifier Separation Protocol as it provides the association between an identifier and its locators.

From an architectural standpoint there are two possible ways in which a mapping system could supply mapping information. It can either provide individual answers to specific requests (pull), or distribute (push) all the mappings onto listeners. In pull-based mapping systems, the ITR sends queries to the mapping system every time it needs to contact a remote EID and has no mapping for it. The mapping system then returns a mapping for a prefix that contains this EID. Pull-based mapping systems have thus similarities with today's DNS. Proposed pull-based mapping systems include LISP+ALT [36], LISP-CONS [23] and LISP-DHT [64]. In push-based mapping systems, the ITR receives and stores all the mappings for all EID prefixes even if it does not contact them. Push-based mapping systems have thus similarities with today's BGP. To the best of our knowledge, NERD [52] is the only proposed push-based mapping system.

### 2.2.1 LISP+ALT

The LISP Alternative Topology (LISP+ALT) [36] is a mapping system distributed over an overlay. All the participating nodes connect to their peers through static tunnels. BGP is the routing protocol chosen to maintain the routes on the overlay. Every ETR involved in the ALT topology advertises its EID prefixes making the EID routable on the overlay. Note though, that the mappings are not advertised by BGP. When an ITR needs a mapping, it sends a Map-Request to a nearby ALT router. It starts by constructing a packet with the EID, for which the mapping has to be retrieved, as the destination address, and the RLOC of the ITR as the source address. The ALT routers then forward the Map-Request on the overlay by inspecting their ALT routing tables. When the Map-Request reaches the ETR responsible for the mapping, a Map-Reply is generated and directly sent to the ITR's RLOC, without using the ALT overlay.

### 2.2.2 LISP-DHT

LISP-DHT [64] is a mapping system based on a Distributed Hash Table (DHT). The LISP-DHT mapping system uses an overlay network derived from Chord [78]. Choosing this particular structured DHT over others (e.g., CAN, Pastry, Tapestry or Kademlia) was motivated by the algorithm used to map search keys to nodes containing the stored values. In a traditional Chord DHT, nodes choose their identifier randomly. In LISP-DHT, a node is associated to an EID prefix and its Chord identifier is chosen at bootstrap as the highest EID in that associated EID prefix. This enforces mapping locality that ensures that a mapping is always stored on a node chosen by the owner of the EID prefix, see [64] for details. When an ITR needs a mapping, it sends a Map-Request through the LISP-DHT overlay with its RLOC as source address. Each node routes the request according to its finger table (a table that associates a next hop to a portion of the space covered by the Chord ring). The Map-Reply is sent directly to the ITR via its RLOC.

### 2.2.3 LISP-CONS

The Content distribution Overlay Network Service for LISP, LISP-CONS [23], is a hierarchical content distribution system for EID-to-RLOC mappings. It is a generalization of LISP+ALT, which does not use the BGP routing protocol. On the other hand, it adds support for caching in intermediary nodes. In this work LISP-CONS is not considered for the mapping system comparison as it does not seem to evolve anymore.

### 2.2.4 NERD

A Not-so-novel EID to RLOC Database (NERD) [52] is a flat centralized mapping database, using the push-model. Because any change requires a new version of the database to be downloaded by all ITRs, this approach is unlikely to scale to the needs of a future global LISP mapping system. The main advantage of NERD is the absence of cache misses that could degrade traffic performance.

# Part I

# Evaluation of LISP Mapping Systems

# 3

# A DNS-based LISP Mapping System: LISP-TREE

## 3.1 LISP-TREE Overview

LISP-TREE is a hierarchical mapping system that has a clear separation of the storage of mappings and their discovery. The *mapping storage* is under the responsibility of the ETRs while the *discovery* mechanism is built on top of the DNS protocol. The role of the discovery mechanism is to provide a list of ETRs that respond authoritatively for the mappings associated to the queried EID.

When a requester needs to obtain a mapping for an EID, it first sends a request to the discovery part that answers with a list containing the locators of the authoritative ETRs for the requested EID. The requester then sends a Map-Request to one of these ETRs and receives a Map-Reply containing a mapping for the identifier. The mappings are provided by the ETR to let them control their traffic by setting the priorities and weights.

## 3.2 LISP-TREE Model

The bindings between the EIDs and the locators are kept on the authoritative egress tunnel routers of customer domains. These ETRs manage and distribute these mappings with the aim of sinking all self owned EID-prefix mapping requests. All the mapping databases are combined to form the *Mapping Storage*. It will not be further detailed here as its functionality has already been defined in the LISP specification [32].

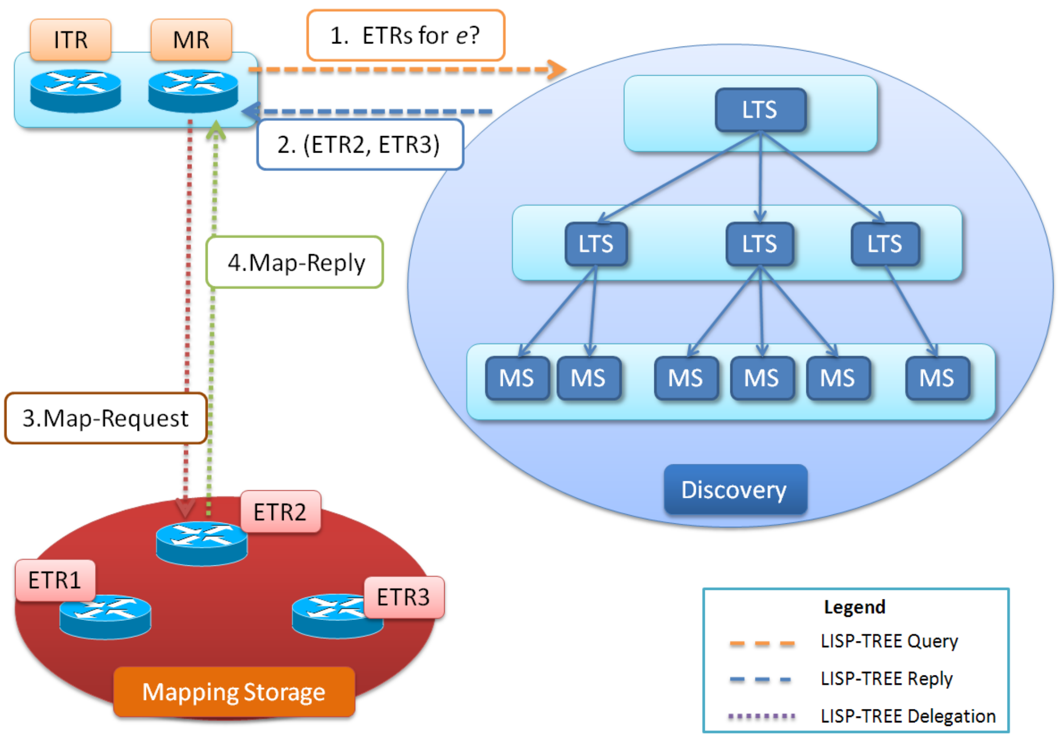


**Figure 3.1:** Global overview of LISP-TREE. 1. The requester asks the discovery part to have the locator of some authoritative ETRs for an EID $e$. 2. The discovery part provides this list. 3. The Map-Resolver sends a Map-Request to one of these ETRs. 4. The ETR sends a Map-Reply back with a mapping for $e$.

ETRs respond with Map-Replies only for the EID space they are authoritative for [32]. Because of this, it is the responsibility of the inquiring node, the Map-Request originator, to find the locators of the ETRs authoritative for the queried identifier. Such functionality is provided by the *discovery part* of LISP-TREE. It is implemented on top of the DNS protocol [69] and builds its tree by logically linking *LISP-TREE Servers (LTS)*. Although LISP-TREE is built over DNS, we suggest to deploy it independently to keep the separation between the DNS names and the identifier resolutions.

All the LTSes are *responsible* for an EID prefix and build a hierarchy determined by their intrinsic relationship. Therefore, an LTS responsible for EID prefix $p_1$ is ancestor of an LTS responsible for EID prefix $p_2$ if and only if $p_1 \preceq p_2$[1]. Moreover, any LTS of prefix $p_1$ is a parent of an LTS responsible for prefix $p_2$ if and only if there exists no

[1]$p \preceq q$ if and only if prefix $p$ is shorter (less specific) than $q$

LTS of prefix $p_3$ such that $p_1 \prec p_3 \prec p_2$. All these strict ordering relations are stored by parent LTS as a list of child servers. Exceptions are the lowest level servers, the leaves, which store a list of ETRs that are responsible for their associated prefix. Hence, the search for any EID $e$ ends at a leaf of prefix $p$ that respects $p \preceq e$.

The authoritative ETRs are registered to their responsible leaf LTSes by using the Map-Register procedure defined in [35]. In LISP terminology, the leaf LTSes are called *Map Servers (MS)* [35].

To make LISP-TREE transparent for the ITRs, *Map-Resolvers (MR)* [35] are added to the system. When a Map-Resolver receives a Map-Request from an ITR, it queries the LISP-TREE discovery part to obtain the list of the authoritative ETRs for the EID in the request. Once the MR has the corresponding list of authoritative ETRs, it sends a Map-Request to one of them and subsequently forwards the received Map-Reply to the requesting ITR. Thanks to this functionality, the ITRs do not need to know anything about LISP-TREE, they just need to send a Map-Request to an MR.

To avoid circular dependencies in the addresses, every node involved in the mapping system (i.e., MRs, LTSes, MSes and xTRs) is addressed with a locator.

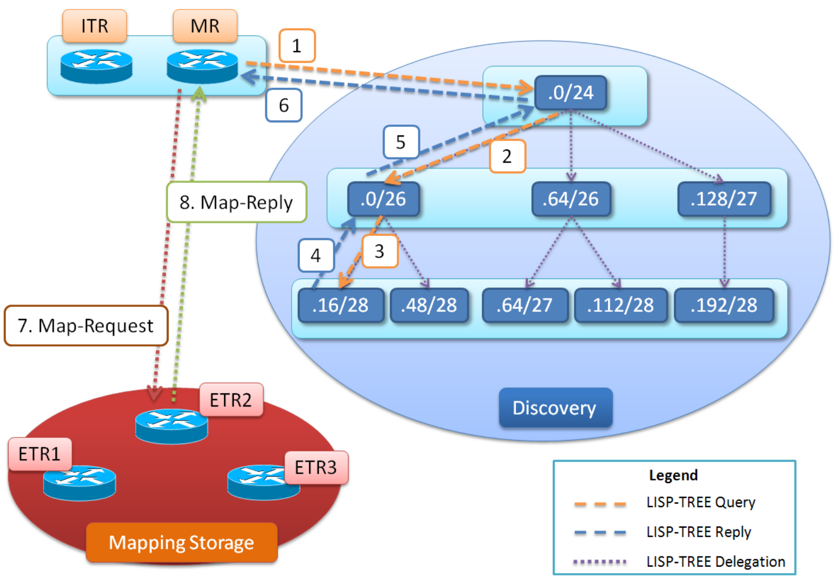
## 3.3 LISP-TREE Modes

Like DNS, LISP-TREE can run in two distinct modes: (*i*) *recursive* and (*ii*) *iterative*. Figure 3.2 illustrates the difference between the two and presents a requester who wants to obtain a mapping for the EID `192.0.2.20`. To simplify the figure, only the last EID octet's is shown.

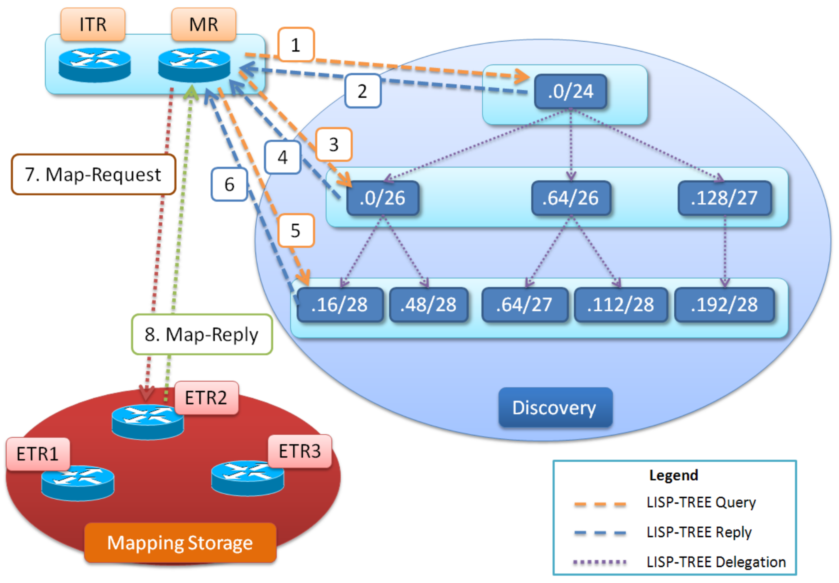
### 3.3.1 Recursive Mode

In the recursive mode (Figure 3.2(a)), the MR requests a mapping for EID $e$ from a root LTS[1]. The root LTS sends a request to one of its children responsible for $e$ and so on until a MS is reached. The MS then generates a list of $e$'s authoritative ETRs locators and this list is back-walked until the answer arrives the root LTS. At that stage, the root LTS sends the reply back to the MR.

[1]A root LTS is an LTS that serves the root of the tree



(a) Recursive mode: queries in the discovery parts are progressively transmitted to a MS. The MS answer then back-walk the tree up to the root that sends the answer to the MR.



(b) Iterative mode: queries are moving back and forth from the MR and the LTSes, starting at a root LTS until a MR is reached. The MS then provides the answer to the MR.

**Figure 3.2:** LISP-TREE works with different mode: (*a*) recursive and (*b*) iterative.

### 3.3.2 Iterative Mode

In the iterative mode (Figure 3.2(b)), the MR requests a mapping for EID $e$ from a root LTS. The LTS then sends back the locators of its children responsible for $e$ to the MR. The MR then sends a request for the mapping to one of those children. The child does the same and answers with a list of locators of its children responsible for $e$, and so on until a MS is reached. The MS then generates a list of $e$'s authoritative ETRs locators and sends it to the MR.

In both modes, when the MR receives the ETR locators list, it sends a Map-Request to one of them to get a mapping for $e$ and eventually receives a Map-Reply. It is possible to optimize for latency in the last step, by allowing the MS to do DNS-to-LISP protocol translation and have it send the Map-Request to the ETR. We did not consider this scenario in order to keep the architectural separation and with that the good troubleshooting properties of LISP-TREE.

## 3.4 LISP-TREE Deployment Scenario

A key factor for the long-term success of a mapping system is its ability to scale. However, the effective deployment of a mapping system also largely depends on its ability to fit with a business model accepted by the community willing to use it.

We therefore propose a model that relies on the the *delegation* principle which is very common in the Internet. A globally approved entity (e.g., IANA, IETF. . . ) defines the minimum set of rules and delegates the operations to independent service providers that can, at their turn, delegate to other providers and fix extended rules. People wishing to gain access to the system are free to contact any provider and sign agreements with them. The providers are differentiated by the extra services they propose.

The EID prefix delegation follows an approach similar to the current Internet prefix allocation. The IANA divides the EID space in large blocks and assigns them to five different *Regional EID Registries (RER)* that could be the current RIRs. The RERs are responsible for the allocation of prefixes to the *Local EID Registries (LERs)* and large networks. The LERs can then provide EID prefixes to some customers or LERs below, which, at their turn, can delegate the prefixes to other LERs or to customers. In this scenario, we consider that the EID space is independent from the currently assigned IP space. For example, it could be a subset of the IPv6 address space. This separation means that the EID space does not have to support all the legacy decomposition of prefixes observed on the Internet.

In this scenario, the tree has at least three levels. The first level (i.e., the root) is composed of LTSes maintained by the RERs. These servers replicate the same information: the list of all LERs responsible for all the large blocks of EIDs. The number of such blocks is limited (maximum 255 in an /8 decomposition perspective) meaning that the total state to maintain at the root LTSes is limited even if every prefix is maintained by tens of different servers. Level 2 of the tree is composed of the LTSes maintained by the LERs and the lowest level encompasses the map servers responsible for customer ETRs. State at level 2 depends on the deaggregation of the large block of EIDs and implicitly on the number of subscribed customer map servers. To avoid having to support a large number of deaggregated prefixes, an LTS can partially deaggregate its EID block (e.g., divide it by two) and delegate such sub-blocks. In other words, the state to maintain at an LTS can be controlled by adapting the depth of the tree.

It is important to note that the EID prefixes are independent of any given ISP, and that additional levels in the hierarchy can be defined if needed. It is worth to note that this raises the problem of registrar lock-in: once an organization gets an allocation, it cannot move the allocated prefix to a different registrar. It is also important to remark that any LTS (including the MSes) can be replicated and maintained by independent server operators and implementations. Therefore, to limit the impact of the lock-in, we would recommend the registrars to ensure enough diversity in LTS operators for their blocks. No significant lock-in is observed in DNS today thanks to this way of deploying TLDs. The load can also be controlled among the different replicas by deploying them in anycast. Finally, the use of caches, like in the DNS, can reduce the number of LTSes involved in every query.

4

# Simulating Mapping Systems with CoreSim

LISP and LISP+ALT are discussed in the LISP working group of the IETF and an implementation is being developed for Cisco platforms. A world-wide testbed of more than 50 tunnel routers (at the time of this writing) relying on the LISP+ALT mapping system is also deployed[1]. In addition, an open source implementation of the LISP tunnel router functionality is developed by the OpenLISP project [47] for the FreeBSD operating system. However, while these implementations help to validate and gain operational experience with the proposed protocols, they do not allow to estimate the behavior of LISP at a very large scale. We developed *CoreSim* [27] for this purpose, an open source Internet-scale LISP deployment simulator[2]. CoreSim combines a packet trace and Internet latency data to simulate the behavior of an ITR and the mapping system.

CoreSim works on top of a topology built from measurements performed by the iPlane infrastructure[3], which provides Internet topology information and the latency between arbitrary IP addresses. The simulator reports mapping lookup latency, the load imposed on each node of the mapping system and cache performance statistics.

CoreSim is composed of two main building blocks (see Figure 4.1): the first one, `ITR`, simulates an ITR with its associated operations (sending Map-Requests, caching

[1]See `http://www.lisp4.net/` for more details.
[2]Available from `http://www.cba.upc.edu/lisp`
[3]See `http://iplane.cs.washington.edu/`

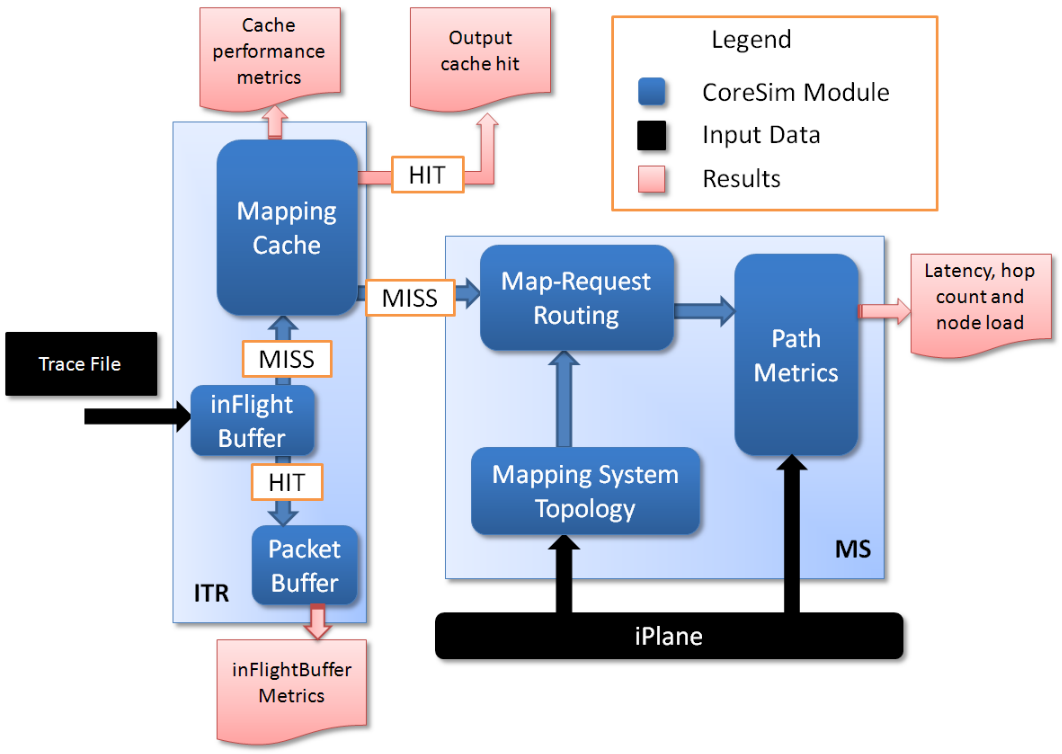


**Figure 4.1:** CoreSim architecture

Map-Replies and forwarding packets), while the second, `MS`, simulates the mapping system (path taken and mapping latency).

A packet trace file is used as input data for the ITR block. The position of the simulated ITR in the topology built by CoreSim (see next section) is determined by the real world location of link where the trace was captured.

## 4.1 Topology Model

The first element to model the behavior of mapping systems in a large scale network is the network itself. For CoreSim, we chose to use the current Internet as the reference topology.

### 4.1.1 Tunnel Routers

More precisely, the topology used in the simulator is composed of all the points of presence (PoPs) as defined by the iPlane measurements set [62]. This dataset associates several IP prefixes addresses with each PoP and provides the links that it has to other PoPs. For our simulations we assume that one LISP tunnel router (xTR) will be deployed at the border router of each autonomous system. We use the PoP connectivity information to select xTRs for our simulated topology, as follows. First, we determine the AS number of all PoPs, and count the number of links that each PoP has to PoPs in other ASes (inter-domain links). We consider as the AS border router and LISP xTR the PoP with the highest number of inter-domain connections. The iPlane dataset yielded 14340 such PoPs, which is close to half of the ASes participating in the Internet today.

### 4.1.2 EID Prefixes

Once the xTRs are selected, they are assigned several EID prefixes. Those prefixes are the IP prefixes currently advertised in the default-free zone (DFZ) by this AS. Since the causes of prefix deaggregation in BGP is not always clear, we removed from the iPlane dataset the more specific prefixes. These prefixes are mostly advertised for traffic engineering purposes [67] and would not be advertised with LISP as it natively supports traffic engineering in the mappings. A total number of $112,233$ prefixes are assigned based on originating AS to their respective xTR[1].

### 4.1.3 Delays

To be able to simulate the time required to obtain a mapping from a mapping system, CoreSim must have information about the delays between the nodes in the modelled topology. For this, we rely on the iPlane platform as well, which combines BGP looking glass information with a large number of daily traceroutes from over 300 vantage points [61, 62]. By combining this data, the iPlane Internet latency lookup service is offered, which returns the measured latency between arbitrary IP addresses. CoreSim relies on this service to simulate the time required to obtain a mapping. Unfortunately, iPlane does not provide all possible delay pairs. Because of this, for 35% of the lookups, we use a latency estimator that correlates the geographical distance between IP addresses as reported by the MaxMind database [7], with the latencies based on an iPlane training dataset (see details in [24]). This approach was found only slightly

[1]Data is from March 2009

less accurate than other more complex algorithms in [15] using measurement data from over 3.5 million globally well distributed nodes.

An important assumption made by CoreSim is that there is no churn in the topology during a simulation run, meaning that the delay between two nodes is constant, and that nodes never fail.

## 4.2 ITR Model

CoreSim studies the behavior of only one ITR at a time, therefore, out of all xTRs, one is selected as the ITR under study (see Figure 4.1). The PoP for the selected ITR is determined by manual configuration, based on the point of capture of the traffic trace that is fed to the simulator. The ITR caches mappings (i.e., resolved Map-Requests) and evicts entries after 3 minutes of inactivity.

The ITR is composed of three modules: the mapping cache module (`mapCache`), the packet buffer module (`packetBuffer`), and the module responsible of tracking ongoing map requests (`inFlightBuffer`). They work as follows:

**mapCache** The mapping cache module maintains a list of all prefixes that are currently in the cache after a successful Map-Request, along with the timestamp when they will expire from the cache. For each hit on a prefix, the timestamp is updated to $t_{packet} + T$ where $t_{packet}$ is the packet timestamp and $T$ the minimum lifetime of an entry in the cache. We used a 3 minutes timeout $T$ for the simulations in this work. This approach permits to maintain entries in the cache if there are packets more than once every $T$ minutes. It is important to notice that we considered that mappings have a TTL longer than the duration of the trace meaning that entries are never refreshed with a Map-Request once they enter the cache.

**packetBuffer** The packet buffer module holds the buffered packets that caused a cache miss and must wait for the Map-Reply to be received before they can be sent. The implementation of the `packetBuffer` treats transport layer protocols differently. In the case of a real-world TCP connection, the sender would transmit the first packet of the three-way-handshake and wait for the acknowledgement, that would only be sent after the Map-Request completed and the buffered TCP SYN packet was successfully transmitted by the ITR. The simulator cannot reproduce this behaviour of the transmitting host, it has to work with the existing trace. In order to have correct values for the number of packets in the `packetBuffer`, we only consider the first TCP packet for each destination IP address that produces a cache miss and drop the following ones, until the Map-Request is answered. In the case of UDP packets the simulator buffers all of them. This may not always be the correct behaviour, but it is highly dependent on the sending application on the originating host, and buffering all provides an upper bound for the size of the buffer.

**inFlightBuffer** This module keeps track of all ongoing mapping requests, so that the second and the following packets destined to the particular prefix can be buffered.

Algorithm 1 describes the details of the ITR module implementation. For each arriving packet the simulator first evaluates the timestamp, removes from the buffer packets for which a Map-Reply arrived in the last inter-packet interval, and writes statistics to disk. Next, the ITR examines if there is an ongoing Map-Request for the destination of the packet, in which case the packet is buffered. Otherwise, the RLOC for the destination is either cached, producing a cache hit or unknown, producing a cache miss. For a cache hit the entry's timeout value is updated to $t_{packet} + T$. In the case of a cache miss, the mapping system module is used to determine the lookup latency and path, an entry is added to the cache, the packet is buffered and the `inFlightBuffer` is updated to save state. Packets with destinations without a valid mapping are dropped.

## 4.3 Mapping System Models

In CoreSim, a LISP Mapping System is modeled as an IP overlay. The overlay is mainly composed of nodes of the topology module, but also includes some mapping system specific ones. As expected the organization of the overlay depends on the considered mapping system.

The simulator routes Map-Requests from the ITR to the node authoritative for the mapping (ETR). The path of the query and the latency of each hop are recorded in order to infer statistics and metrics of interest.

**Algorithm 1** ITR packet processing

```
Prune expired inFlightBuffer entries
Send outstanding packets from buffer
if ∃ ongoing Map-Request for destination EID then
    /* inFlight Hit */
    if Proto = TCP then
        Buffer first packet only
    else
        Buffer packet
    end if
else if $\exists Mapping$ in mapCache for destination EID then
    /* mapCache Hit */
    Update $Timeout \leftarrow t_{packet} + T$
else
    /* mapCache Miss */
    Perform lookup
    if $\exists Mapping$ then
        /* Mapping found */
        Create new mapCache entry
        Create new inFlightBuffer entry
        Set $Timeout \leftarrow t_{packet} + T$
        Buffer the packet
    else
        /* No mapping found */
        Discard packet
    end if
end if
```

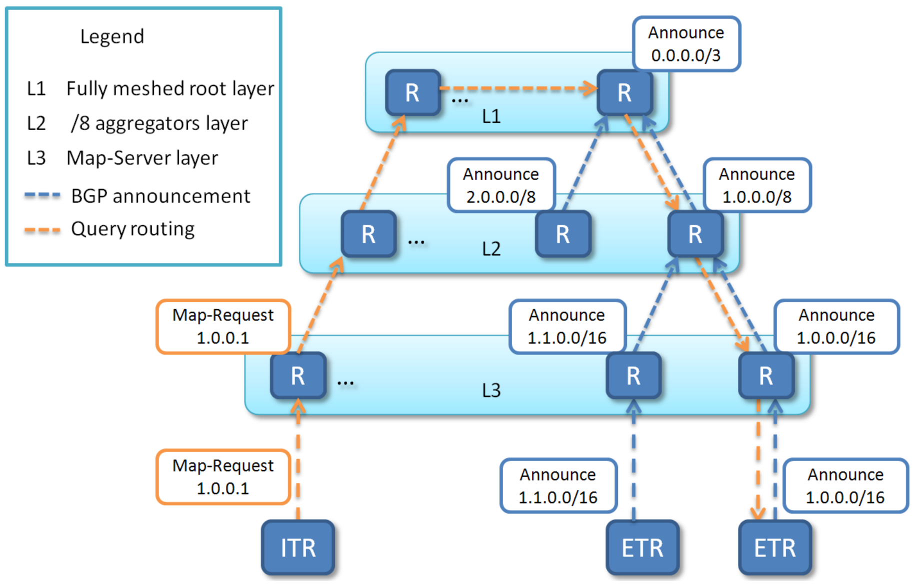


**Figure 4.2:** LISP+ALT architecture

### 4.3.1 LISP+ALT Model

The LISP+ALT draft [36] envisages a hierarchical topology built with GRE tunnels but does not recommend any organization for the overlay. Therefore, among all the possible organizations, a hierarchical overlay structure with strong EID prefix aggregation for advertisements has been chosen.

Based on discussions on the LISP mailing list, a three-level hierarchy was decided (see Figure 4.2). In this hierarchy, the bottom leaf nodes are ALT routers belonging to a certain domain. The upper two levels are dedicated ALT routers, which may be offered as a commercial service by providers or registries. Each of these nodes is responsible for certain aggregated prefixes, and connects to all lower level nodes which advertise sub-prefixes included in those prefixes. The hierarchy consists of 16 root (first) level ALT routers, responsible for the eight /3 prefixes, and 256 second level ALT routers, each responsible for a /8 prefix. For each /3 prefix two ALT routers at different locations are used and each lower level ALT router connects to the one with the lowest latency. All these 16 routers are connected to each other with a fully meshed topology. Please note that LISP+ALT can support other topologies and for instance include intra-level links.

Map-Requests are routed via the shortest path through the hierarchy starting from the bottom layer where the ITR is connected to the ALT topology.

To calculate the lookup latency in ALT, the simulator uses the destination address of the packet that triggers a Map-Request and routes the packet via the shortest path through the hierarchy starting from the bottom layer. Each node through which the Map-Request passes has a FIB with the prefixes that it "sees": all prefixes of the child nodes and aggregated prefixes from the parent node. Packet forwarding proceeds according to the FIB entries and the path is recorded. The topology module then computes the latency of all segments that the Map-Request traversed, and all are added together. A Map-Request that traverses the path $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$ will have the latency calculated as

$$L_{MR} = L_{AB} + L_{BC} + L_{CD} + L_{DE} + L_{EA}, \tag{4.1}$$

where $L_{AB}$ is the latency between routers $A$ and $B$. The last segment, $L_{EA}$, corresponds to the Map-Reply.

### 4.3.2 LISP-DHT Model

LISP-DHT uses a slightly modified Chord protocol, with a 32 bit ChordID namespace for IPv4 address compatibility. Whenever searching for keys (EIDs to be looked up), no hashing of the key takes place unlike in the original protocol. Nodes entering the Chord ring (Mapping Servers) choose as ChordID the highest IP address in their prefix (see Figure 4.3). If a domain has several non-contiguous prefixes, their mapping server will participate in the DHT with a separate instance for each such prefix.

The iPlane dataset contained $112,233$ prefixes after filtering, requiring a Chord ring of the same size. To the best of our knowledge, all available Chord simulators use an event-based approach, which cannot scale to such large number of nodes. Using $112,233$ instances of a real Chord implementation isn't feasible either. To solve the issue, we made the assumption that there are no link and node failures and thus LISP-DHT is modelled as a static Chord overlay. This implies that we consider no LISP site connects to, or is removed from, the LISP-DHT overlay during a simulation. Handling churn in such a large scale Mapping System is left for further work.

Since the LISP-DHT is stable for the entire simulation, our LISP-DHT model starts from all the EID prefixes and computes the finger tables of all nodes on the overlay.

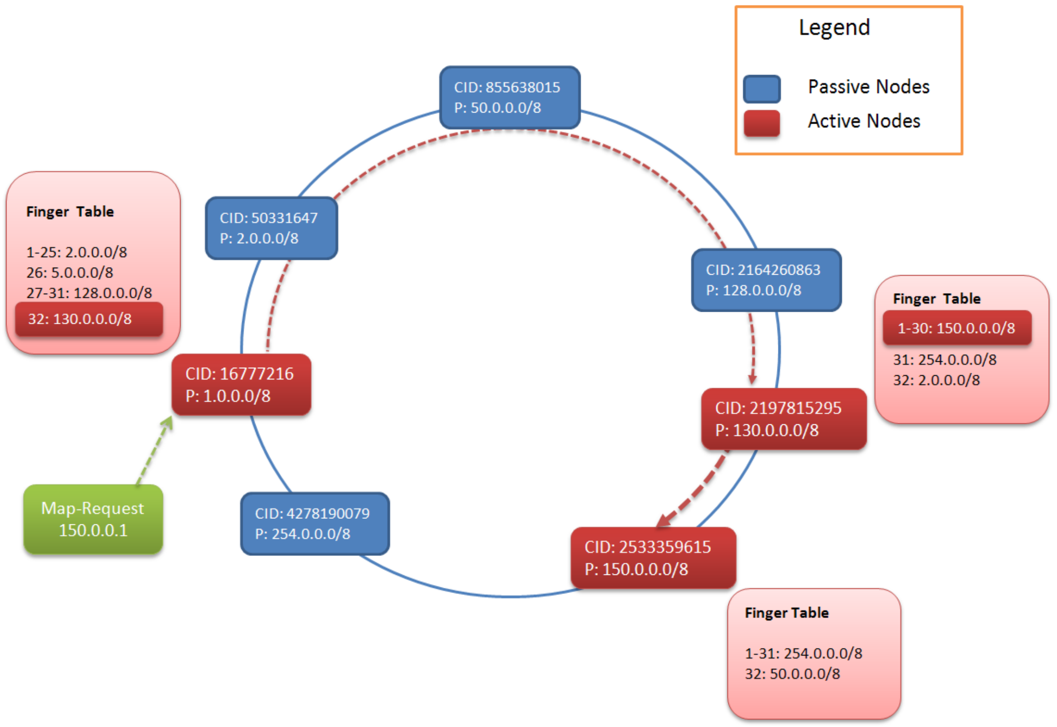


**Figure 4.3:** Modified Chord example

This is possible as in Chord the finger tables only depend on the ChordIDs of the nodes and not on when the nodes have joined the ring.

To look up a destination EID in the DHT, the regular Chord protocol is used: the originating node looks for the successor of the EID in its finger table and sends the Map Request to that node. Our model considers two modes for the queries: *recursive* and *iterative* modes. In recursive node, the originating node sends the Map Request to the closest node according to its finger table. The closest node then forwards the Map Request. This continues until the node responsible for the queried EID prefix receives the Map Request and replies directly. In iterative mode, the same nodes are involved in message routing, but each node replies to the originating node with the ID of the next hop, until it learns the RLOC of the node responsible for the queried EID. In both cases, the path is recorded and the lookup latency is calculated. Suppose the Map-Request traverses nodes $A$, $B$, $C$, $D$, and $E$, with $A$ being the originating node.

For simulating LISP-DHT recursive mode the latency is calculated as

$$L_{MR} = L_{AB} + L_{BC} + L_{CD} + L_{DE} + L_{EA}. \tag{4.2}$$

In the case of iterative mode, we have:

$$L_{MR} = 2 \cdot (L_{AB} + L_{AC} + L_{AD} + L_{AE}). \tag{4.3}$$

As previously noted, the topology is considered symmetrical in terms of latency, and the round-trip times are computed by doubling the latency from the originating node to each node the query passes through.

#### 4.3.2.1 Chord Simulator Validation

To validate our implementation of the Chord protocol, we used a modified version of OpenChord 1.0.5 [1] to generate a topology of 50 nodes. After the DHT stabilized, we saved the finger tables and compared them to the ones generated by our simulator for the same topology. We compared the results for three different topologies and the finger tables were identical in all cases.

### 4.3.3 LISP-TREE Model

CoreSim considers only one ITR at a time, and because the MR always selects the closest root LTS, the one selected by the MR is always the same. Therefore, the simulator considers only one root LTS for the tree. This server connects to the 256 level 2 LTSes that are each responsible for one `/8` EID prefix. In turn, these servers know about all the third level LTSes that are responsible for the prefixes included in their `/8` prefix. These third level servers are the map servers the ETRs subscribe to. The simulator assumes that an ETR registers to a single MS, and that MS is located in the same PoP as the ETR. Since the simulator assigns them to the same PoP, the resulting latency between them is 0. Further, because the latency introduced by DNSSEC is two orders of magnitude lower than typical mapping lookup latencies [84], it is considered negligible and is not accounted for in the simulator. This deployment scenario implemented is an instantiation of the one presented in Section 3.4 which is congruent with the current Internet.

[1] Avalable from `http://open-chord.sourceforge.net/`

# 5

# Comparison of Selected Mapping Systems

Chapter 3 described in detail the advantages of the proposed mapping system from an architectural point of view. This chapter complements that with a quantitative analysis, comparing several performance metrics of three mapping systems: lookup latency, hop count, node load and the amount of state stored in mapping system nodes, using the simulation tool and environment presented in Chapter 4. Low lookup latency improves user experience for new flows, while node load and state affect the scalability of the system. We begin by describing the packet traces that support our evaluation.

## 5.1 Experimental Datasets

In order to evaluate the performance of the mapping systems presented above we used traffic traces collected at the border routers of two university campuses, because border routers are the most likely place to deploy a LISP tunnel router. The first trace was captured at Université catholique de Louvain (UCL) in NetFlow format, and the second is a packet trace from Universitat Politècnica de Catalunya (UPC).

### 5.1.1 UCL

The UCL campus is connected to the Internet with a 1 Gbps link via the Belgian national research network (Belnet). This trace consists of a one day full NetFlow trace collected on March 23, 2009. For this study, only the outgoing traffic is considered,

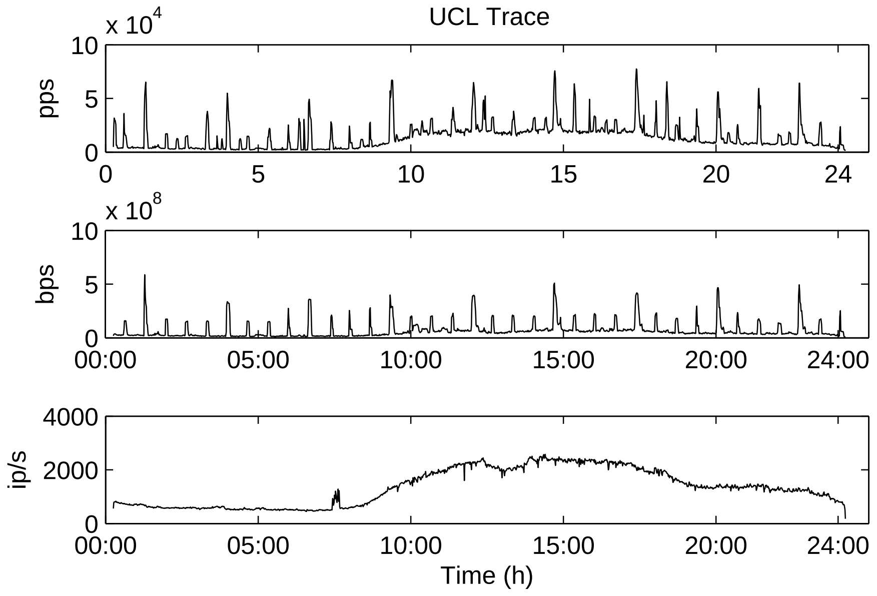


**Figure 5.1:** UCL traffic trace characteristics

representing 752 GB of traffic and 1,200 million packets for an average bandwidth of 69 Mbps. A total number of 4.3 million different IP addresses in 58,204 different BGP prefixes have been contacted by 8,769 different UCL hosts during the 24 hours of the trace. The UCL campus is accessible to more than 26,000 users.

NetFlow generates transport layer flow traces, where each entry is defined as a five-tuple consisting of the source and destination addresses and ports, and the transport layer protocol. The simulator however requires packet traces. This is why the NetFlow trace collected at UCL has been converted into a packet trace: for each flow, we generated the number of packets specified in the NetFlow record, distributed evenly across the flow duration and the size of the flow. Throughout the rest of the study, the term *UCL trace* corresponds to the packet trace obtained from the NetFlow trace collected at UCL.

Figure 5.1 shows he evolution in time of the bandwidth (*bps*), packet rate (*pps*), and the number of different destinations per second exchanging traffic with the UCL campus (*ip/s*). The short bandwidth peaks that appear at regular intervals are caused by periodically scheduled data transfers of well connected scientific servers on the campus. It is worth to note that the working hours start at 8:30 AM (7:30 AM GMT) and finish

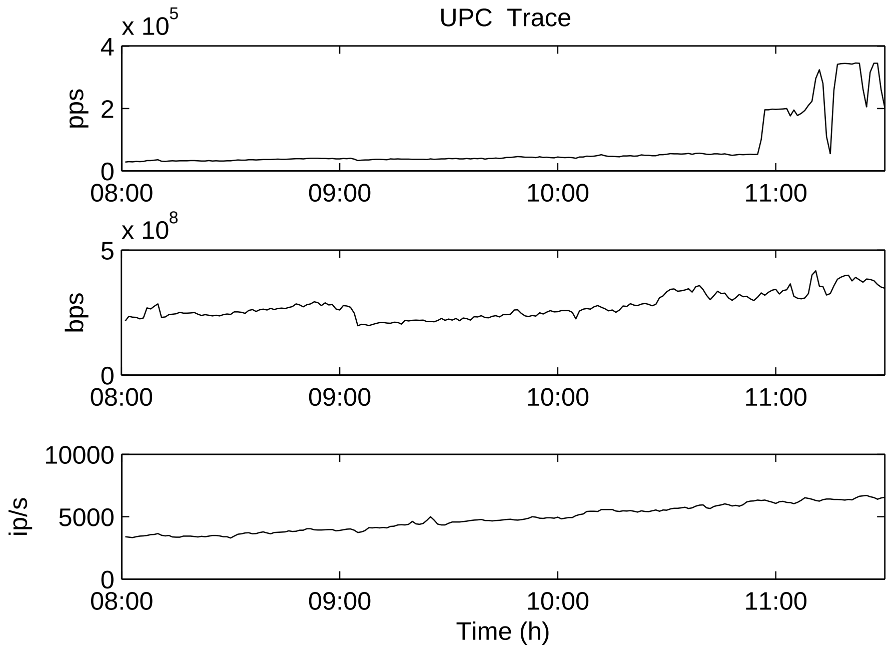


**Figure 5.2:** UPC traffic trace characteristics

at 6 PM (5 PM GMT) at UCL. Traffic exhibits a 1 hour peak at the beginning of the workday. It is caused by tasks that are done once in the morning by the people, like reading the news and replying in batch to their emails.

### 5.1.2 UPC

The second unidirectional trace we used was captured at the 2 Gbps link connecting several campus networks of UPC to the Catalan Research Network (CESCA) with the help of the CoMo infrastructure [20]. It consists of the egress traffic on May 26, 2009 between 08:00–11:49 local time, and contains about 1,200 million packets accounting for 463 GB of traffic with an average bandwidth of 289 Mbps. 4.3 million distinct destination IP addresses from 56,387 BGP prefixes were observed in the trace (see Figure 5.2). UPC Campus has more than 36,000 users.

## 5.2 Cache Miss Rate

The average packet rates at the UCL and UPC border routers are 13 Kpkts/s and 87 Kpkts/s, respectively. However, due to the nature of the traffic, a mapping is already cached by the ITR for most of the packets, with only 0.31% and 0.1% of cache misses for the mentioned vantage points. A cache miss occurs when no mapping is known for an EID to encapsulate. On cache miss, a Map-Request is sent to obtain a mapping, resulting in 2,350,000 Map-Requests sent for UCL and 560,000 for UPC during the 24h and 4h periods of the traces, which are shorter than the default TTL value (1 day) for mappings. As a result, all cache evictions were due to lack of activity to a particular prefix for 3 minutes, rather than expired TTL. These values for Map-Requests have to be compared with the total of 1,200 million packets observed for both traces. The difference between UCL and UPC can be explained by the higher average packet rate of the UPC trace, which keeps the cache more active, resulting in less timed out entries.

During our simulations the maximum number of mapping cache entries reached was 22,993 and 15,011 for the two traces. This is an order of magnitude less than routes in the DFZ. For a detailed mapping cache study, please refer to [44].

It is important to note that the results are obtained for a cache initially empty. Therefore, the miss rate is very important at the beginning of the experiment and thus influences the number of requests. The number of Map-Requests would have been dramatically smaller if we had considered the system at the steady state.

## 5.3 Mapping Lookup Latency

The mapping lookup latency is particularly important in mapping systems as it represents the time required to receive a Map-Reply after sending a Map-Request. When an ITR waits for a mapping for an EID, no packet can be sent to this EID. If this delay is too long, the traffic can be severely affected.

For instance, one of the Internet's most popular resources, the World Wide Web is continuously evolving, delivering web pages that are increasingly interactive. Content on these pages is often from different servers, or even different providers, and presents a multi-level dependency graph [86]. This is already a challenging environment for some applications, and the introduction of a new level of indirection has the potential to introduce additional latencies. However the ATLAS study [51] shows that content is more and more comming from datacenters and CDNs.

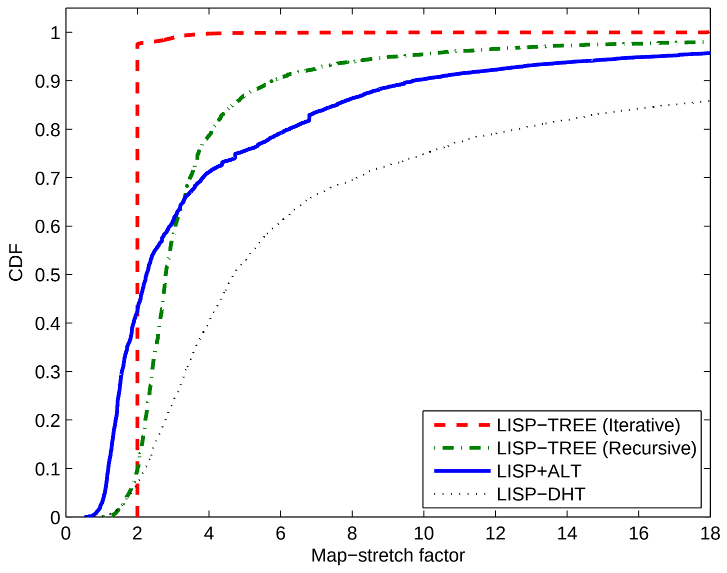
To compare the considered mapping systems, we define the *map-stretch factor* of a lookup as the ratio between the total time required for performing it and the round-trip delay between the ITR and ETR. A low map-stretch indicates that the mapping system introduces a low delay overhead.

Figure 5.3 presents the cumulative distribution function of the map-stretch factor obtained for both the UCL and UPC traces. In most of the cases (over 90%) the iterative LISP-TREE presents a stretch factor of 2. This can be explained as follows. After the initial cache warm up, there is no need to contact the root and level 2 servers, their responses are already cached. Still, the discovery phase cannot be completely avoided and the level 3 MS have to be contacted for the RLOCs of the ETRs. These are not cached in the MR, because it would have a negative impact on the mapping dynamics, limiting the possibilities for supporting fast end-host mobility.

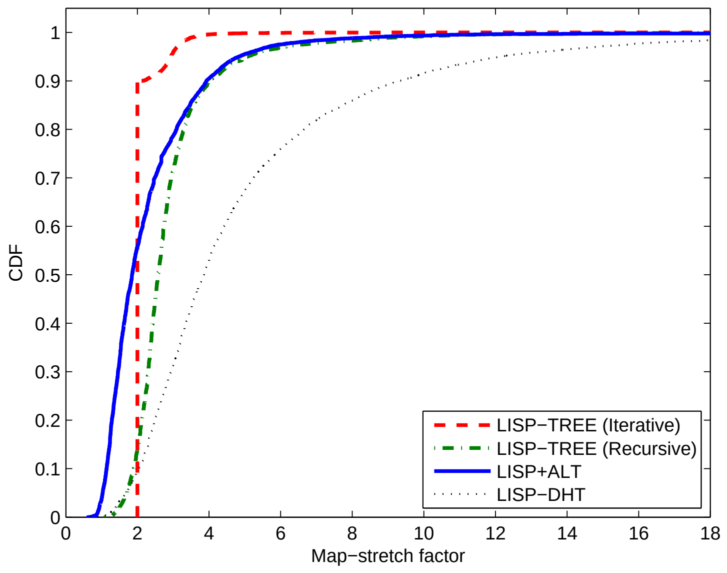
Recursive LISP-TREE is slower than iterative in over 90% of the cases. The caching limitation mentioned in the previous paragraph has particularly negative consequences for this operating mode: since the only response arriving to the MR is the list of RLOCs for the authoritative ETRs, no caching at all can be done. The 10% of the cases when recursive is better than iterative can be attributed to the path via the tree being faster than from the MR to the MS, resulting in map-stretch ratio values below 2.

Concerning LISP+ALT, its latency performance is similar to LISP-TREE iterative. This is because in LISP+ALT, the queries are routed through to the overlay topology, which is composed by core routers. According to our assumptions (see Section 4), these nodes are located in well connected PoPs. However, in iterative LISP-TREE the query flows in almost all the cases between the ITR and ETR, which may not be so well-connected. When comparing LISP+ALT and LISP-TREE recursive, we can see that LISP+ALT performs slightly better. In the recursive mode of LISP-TREE queries are also forwarded through a topology of well-connected nodes, but as we will see, they have to follow a much longer path.

As expected, LISP-DHT has the highest latency because of the longer path taken by the queries routed through the P2P topology. This is a well known problem in P2P networks, and research to tackle this issue is ongoing [89]. However, it is worth to note that LISP-DHT uses a particular Chord ring, where the peers do not choose their



(a) UCL



(b) UPC

**Figure 5.3:** CDF of map-stretch ratio. Use of caching makes LISP-TREE have a constant value for the majority of the lookups.

identifiers randomly, but as the highest EID in the EID prefix. This enforces mapping locality that ensures that a mapping is always stored on a node chosen by the owner of the EID prefix. As a consequence, it may be difficult for LISP-DHT to benefit from existing optimization techniques proposed in the literature.

Average lookup latencies were close to half a second for all mapping systems except LISP-DHT, which had values slightly larger than one second.

Figure 5.4 helps to better understand the mapping latency difference between the mapping systems. It presents a CDF of the number of hops passed by over which a request and reply.

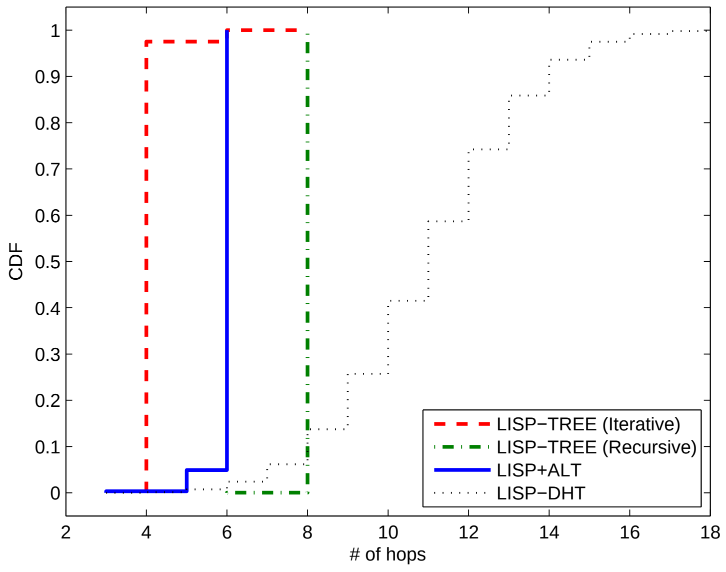
The iterative version of LISP-TREE has the lowest hopcount values, which can be explained, just like for the latency, by caching in the Map-Resolver. Recursive LISP-TREE not helped by caching, and queries have to traverse the full path in each case, for a maximum of 8 hops (Figure 3.2(a)).

The topology chosen for LISP+ALT in the simulator (Figure 4.2) limits the maximum number of hops to 6, but in 95% of the cases, this maximum number of hops is observed. In order to have a shorter path, the destination EID would have to be in one of the `/8` prefixes that doesn't have a more specific part announced separately. As we will see in the next section, this also increases the load on the nodes composing the LISP+ALT mapping system. In fact, Figure 4.2 shows that all queries are forwarded through the root layer. This may result in scalability problems.

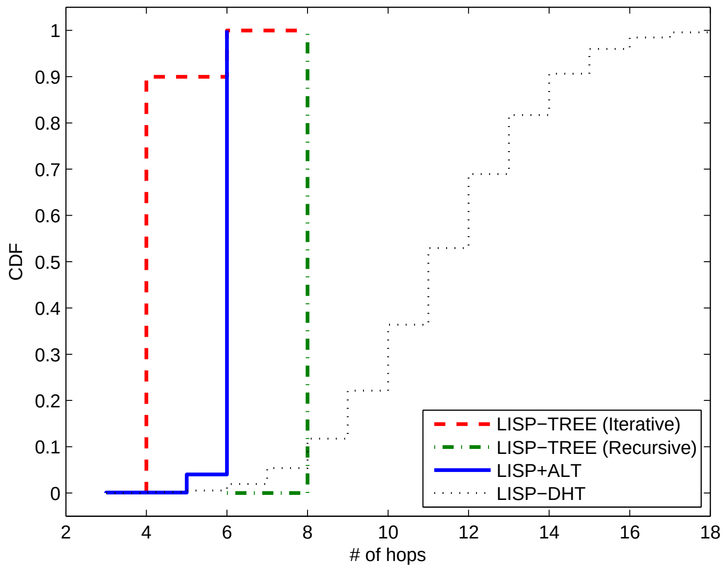
LISP-DHT has a much higher hop count with a maximum of 17 hops. This not only increases the lookup latency, it means that more nodes are involved in the resolution of a mapping than in the case of the other mapping systems, increasing the overall load on the system and the probability of failure.

Summarizing, these results reveal significant differences among the mapping systems under consideration. LISP-TREE and LISP+ALT both use a hierarchical model, where nodes on the query path tend to be congruent with the topology. In contrast, the Chord overlay used to route queries in LISP-DHT does not follow the underlying physical topology. Further, iterative LISP-TREE allows caching and this reduces its mapping latency.

The latency introduced by the mapping system in case of cache misses will likely have a negative impact on the higher layer protocols, in particular on congestion control



(a) UCL



(b) UPC

**Figure 5.4:** CDF of hop count. For hierarchical mapping systems it is almost constant, for LISP-DHT we have a wide array of different values.

**Table 5.1:** Node load in levels 1 and 2

| | Level 1 | | Level 2 | |
|---|---|---|---|---|
| **Mapping System** | **Avg.** | **Max.** | **Avg.** | **Max.** |
| LISP-TREE (Itr.) | 158 | 158 | 372 | 2,354 |
| LISP-TREE (Rec.) | 2,351,815 | 2,351,815 | 14,941 | 70,520 |
| LISP+ALT | 655,600 | 2,348,695 | 29,776 | 2,356,404 |
| LISP-DHT | 147,860 | 1,257,642 | 258 | 2,365,797 |

in the transport or application layer. These issues deserve a dedicated study, and are left for future work.

The findings of this study can be leveraged for further research on the LISP mapping systems because there are several properties that can be modeled based on the latency. We provide on the simulator web page a standalone latency modeling tool, which generates latencies following the distributions that we obtained for each mapping system. The results can help LISP researchers, designers and developers.

## 5.4 Node Load

We define the *node load* as the total number of Map-Request messages processed by nodes of the mapping system during the full run of the trace. For a more thorough analysis, we differentiate between the load caused by messages forwarded by the node (*transit load*) and the load due to the messages for which the node is the final destination (*mapping load*). Mapping load mainly depends on the observed traffic (distribution of destination EIDs) and is mostly the same for all studied mapping systems. On the other hand, transit load is mapping system specific and depends on how a request is sent to the holder of the mapping. Table 5.1 summarizes the load statistics of the UCL trace.

The root LTS is only contacted 158 times in the case of iterative LISP-TREE, that is, the number of times necessary to look up the locators of the level 2 nodes responsible for the contacted `/8` prefixes. Since these locators are cached by the Map-Resolver, they are only requested once. The load on the level 2 servers is also very low, compared to the other mapping systems, where there is no caching on intermediate nodes. In recursive LISP-TREE all Map-Requests have to pass through the root node,

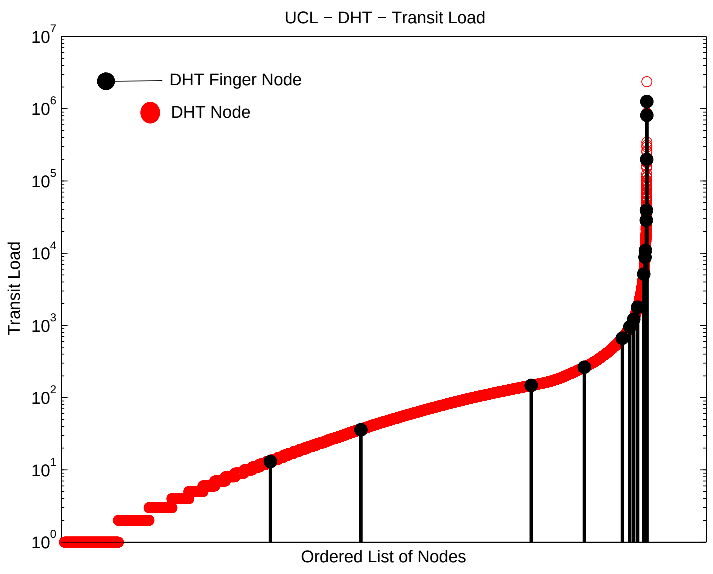


**Figure 5.5:** LISP-DHT load distribution. Fingers of the ITR, represented with the vertical lines, cluster on the right, having a considerable load.

and then get distributed to the lower level nodes, according to the destination prefix. In LISP+ALT, level 1 consists of 7 logical nodes for the seven `/3` covering the routable unicast prefixes. The nodes responsible for IP space with high prefix density sustain a considerably higher load.

For LISP-DHT, level 1 in the table refers to the transit load of the ITR's fingers, while level 2 to the transit load of all other nodes. Figure 5.5 shows the transit load in LISP-DHT. Vertical lines represent fingers of the ITR at UCL, which was originating the Map-Requests. From the 112,233 nodes participating in the DHT, only 74% have routed or received Map-Request messages and are depicted in the figure. We can observe a sharp surge after a load value of 1000, that accounts for about 1.8% of the total number of nodes on the DHT. As expected we find many of the ITR's fingers among these hotspots. Of the 2,365,797 Map-Requests initiated, more than half pass via the last finger and one third via the second last finger. Among the top ten most loaded nodes 7 are not fingers.

Upon further inspection it was discovered that the node with the highest load was the last finger of the ITR. Due to the way Chord lookups work, this node is responsible for half of the key space on the Chord ring, which explains the high load. Further investigation revealed that there is one node which is last finger for 5.6% of the LISP-DHT participants: the one responsible for the prefix 4.0.0.0/8. This is the first prefix from the IPv4 space present in the iPlane dataset. Since Chord is organized as a ring (key space wraps around), this node becomes responsible for the EID space of classes D and E as well. Because this EID space is not represented in the overlay, the result is a disproportional load. The UPC trace produced similar load distribution results.

The likelihood of a node becoming a popular finger in LISP-DHT is proportional to the IP space for which it is responsible in the Chord ring. This IP space, apart from the size of the prefix advertised depends also on the existence of unallocated blocks just before. E.g., if a `/24` prefix is preceded by two unallocated `/8` blocks, it has a higher chance to become a hotspot than a `/8` with an allocated neighboring prefix. Note that even if there is no traffic to the unallocated space, the `/24` from our example is still a hotspot for transit traffic, because it is present in many finger tables.

LISP-DHT's transit traffic distribution characteristics may be desirable for peer-to-peer networks, but are a major disadvantage for a mapping system. Since the transit route is defined only by the Chord routing algorithm, two issues arise: lack of path control may lead to choke points at nodes belonging to small sites, and exposure of mapping traffic to unknown third parties (potentially leading eavesdropping and denial-of-service attacks). This makes LISP-DHT a poor choice as a mapping system.

Due to these architectural differences, the mapping systems under study exhibit different transit load characteristics. Indeed, in the case of LISP-DHT all participating nodes are both transit and destination nodes, while the hierarchical mapping systems have dedicated transit nodes on levels 1 and 2. The iterative version of LISP-TREE has a significantly lower load in those dedicated transit nodes, because of the caching done in the Map-Resolver.

LISP+ALT needs to route all packets through the root nodes, producing a potential hot spot. LISP-TREE on the other hand avoids using the root nodes most of the time, because it is able to cache intermediate nodes. Apart from the obvious scalability advantages, this improves reliability as well, since a total failure of the root infrastructure would still allow partial functioning of LISP-TREE, while no resolutions would be possible in LISP+ALT.

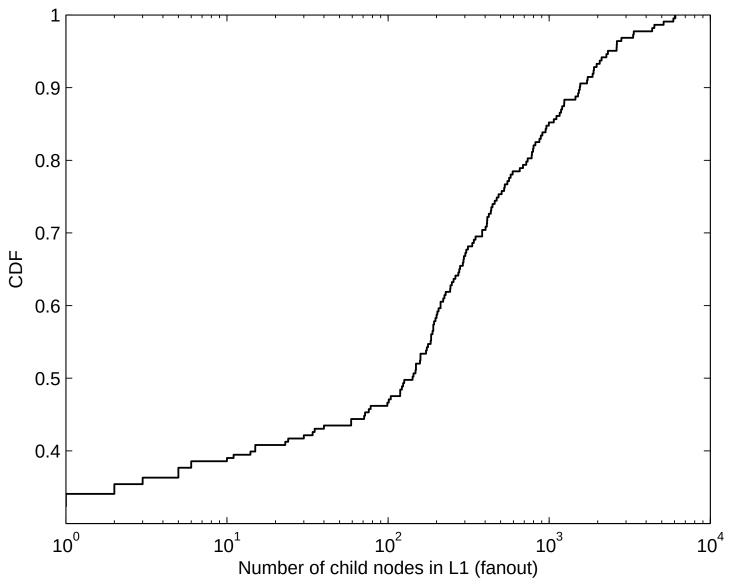


**Figure 5.6:** The cumulative distribution of the amount of state in the level 2 LISP-TREE and LISP+ALT nodes

## 5.5 Operational considerations

The mapping requests are transmitted through the mapping system towards the queried ETR. To achieve this, every node involved in the topology has to store some *state* about its neighbors. The amount of state needed on each node is directly related to the mapping system technology. For example, in an horizontally organized system such as LISP-DHT, all nodes have the same amount of state (32 entries). On the contrary, the amount of state needed on a node in a hierarchical mapping system depends on its position in the hierarchy.

LISP-TREE and LISP+ALT are both hierarchical topologies, and in this study have been deployed according to the same EID prefix distribution, and thus have the same amount of state. The root nodes refer to all the disjoint `/8` prefixes, which amounts to a maximum of 256 entries when all are allocated. The number of nodes that are referred to by level 2 nodes depends on how the prefixes are allocated. Figure 5.6 shows the distribution of the state kept at each level 2 node (both LISP-TREE and LISP+ALT). Finally, the leaves store a small amount of entries, equal to the number of announced prefixes.

Figure 5.6 shows that 15% of the nodes have more than 1000 children and the most connected node has 6072 children. For LISP-TREE this is not an issue, as the nodes only need to keep a database that relates a prefix with the list of locators. It is well known that currently deployed DNS servers scale to much more than thousands of records [81]. However, in the case of LISP+ALT, a BGP session has to be maintained for each child, as well as a tunnel between the two nodes. The costs in terms of configuration, memory, and processing are much higher than for LISP-TREE. A way to alleviate this would be to increase the depth of the tree and thus reduce the number of children of any one node. Unfortunately, this solution stretches the path length within the tree and is likely to increase the mapping lookup latency. Another solution could be to follow an organic deployment, but in that case, the mapping system would eventually have scalability issues because of the lack of aggregability.

## 5.6 Conclusions

LISP-TREE is based on the Domain Name System and is built on a hierarchical topology. From an operational point of view the main advantage of LISP-TREE over LISP+ALT and LISP-DHT is that it enables clients to cache information about intermediate nodes in the resolution hierarchy, and direct communication with them. This avoids resolution request traffic concentration at the root, and as a result iterative LISP-TREE has much better scaling properties. Further, LISP+ALT's scalability also depends on enforcing an intelligent organization that increases aggregation. Unfortunately, the current BGP system shows that there is a risk of an organic growth of LISP+ALT, one which does not achieve aggregation. Neither of the LISP-TREE variants displays this weakness, since their organization is inherently hierarchical (and thus inherently aggregable).

Moreover, the hierarchical organization of LISP-TREE also reduces the possibility for a configuration error which could interfere with the operation of the network (unlike the situation with the current BGP). Existing DNS security extensions can also help produce a high degree of robustness, both against misconfiguration, and deliberate attack. The direct communication with intermediate nodes in iterative LISP-TREE also helps to quickly locate problems when they occur, resulting in better operational characteristics.

Additionally, in LISP+ALT and LISP-DHT, since mapping requests must be transmitted through the overlay network, a significant share of requests can see substantially increased latencies. Our simulation results clearly show, and quantify, this effect. Also, our simulations show that the nodes composing the LISP+ALT and LISP-TREE networks can have thousands of neighbors. This is not an issue for LISP-TREE, but may be problematic for LISP+ALT nodes, since handling that number of simultaneous BGP sessions could be difficult.

# Part II

# Efficient Inter-domain Live Streaming

6

# The CoreCast Protocol

## 6.1 Background

In the first part of the thesis we discussed how the Internet routing architecture can be scaled to support its growth and add desired functionality like improved multihoming, mobility and ingress traffic engineering using LISP. Another area where the current Internet is lacking is efficient one-to-many and many-to-many communications on a global scale. One specific one-to-many communication use case is trasmismission of live streaming data, which is the subject of the second part.

This chapter presents the current state of affairs, describing existing solutions in use today: unicast transmission, network layer multicast (IP multicast), and application layer multicast (P2P-TV).

### 6.1.1 Unicast

The most simple way to transmit live data is the unicast method: the transmitting node maintains a one-to-one connection to each of the destinations, and sends a copy of the payload over each connection (see Figure 6.1).

There are several advantages to the use of unicast:

- *Low barrier to entry:* it requires no special equipment or software support beyond what is already available for typical Internet connectivity
- *QoE flexibility:* data reception is individually acknowledged and data rate can be adapted to each client based on their connection speed, instead of using a lowest common denominator

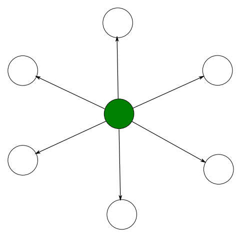


**Figure 6.1:** Unicast transmission architecture. The green node is the source node

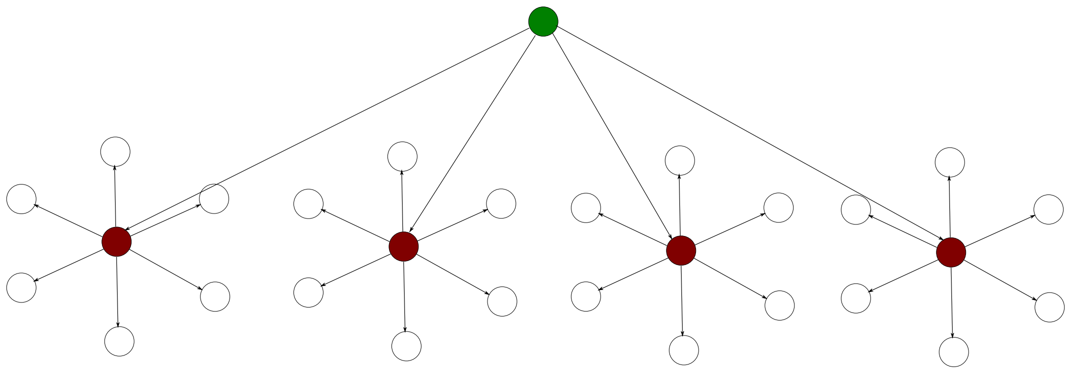
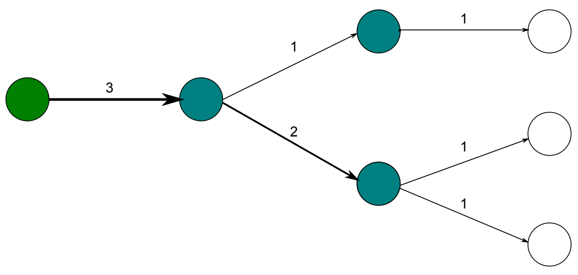


**Figure 6.2:** Content Delivery Network unicast transmission architecture. The green node is the transmission source, the red nodes are CDN points of presence

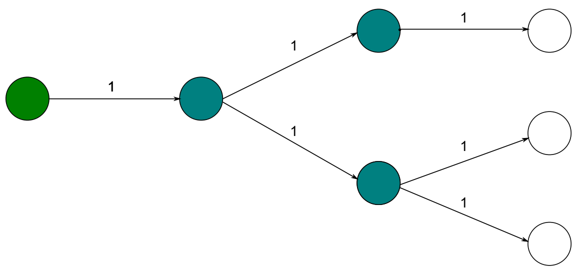
- *Higher security:* access to individual streams can be tightly controlled and encryption or DRM can be individually taylored

On the downside, sending data to each client individually scales poorly, and results in high capital and operational expenditure. Network transmission technologies have an upper limit on bandwidth, requiring the use of clusters of nodes instead of single node, increasing complexity and cost.

Instead of a completely centralized unicast transmission architecture, large streaming providers today use a two-step hierarchy: send one copy of the traffic from the source to geographically distributed Points of Presence (PoPs) over the public Internet or private connections, and distribute to end users from those PoPs (Figure 6.2). This is the business model of Content Delivery Networks (CDNs), who offer their network of PoPs to the streaming content providers. This approach is more scalable and solves the performance issues of the centralized model, however, it is still a costly option.



(a) Unicast



(b) Multicast

**Figure 6.3:** Comparison of the unicast and multicast transmission models. The number above the arrow shows how many copies of the same data is sent over that path segment

### 6.1.2 Network layer multicast

The need for an efficient one-to-many communication solution was recognized over 25 years ago. RFC 988, the initial IP multicast protocol specification was published in 1986, and then refined in RFC 1112 in 1989. The main idea is to create a distribution tree over the network routers leading to the destinations, and over each link in the tree, only one copy of the content is transmitted.

Figure 6.3 illustrates the difference in efficiency between the unicast and multicast transmission methods. In the example there is one source and three clients, connected over a few links, where the path to each destination shares the first link. In the unicast case, a copy of the data is sent for each client over the shared link, while multicast transmission needs a single copy to be sent. Clients wishing to receive the same content join a *multicast group*, a special class D IP address. Multicast signaling then sets up state in intermediary routers on the paths between the sender and all receivers. Incoming packets belonging to a certain multicast groups are sent out on all interfaces where downstream clients exist that joined that group.

IP multicast is used successfully by some ISPs offering “triple-play” services to their customers: data, VoIP and IPTV. They use traditional satellite or terrestrial radio reception methods to receive a large set of television and radio channels at a central location [26]. These channels are then transcoded into digital live streaming data, and distributed *within the ISP network* using IP multicast. The clients use a customized set-top box to view and switch channels, by subscribing to different multicast groups.

This approach saves considerable amount of network capacity to the service provider, but the business model is only applicable within a single network domain. At the other end of the multicast application spectrum are broadcast companies offering multicast streams. While some experiments with this approach have been ongoing for some time, like the BBC’s multicast television and radio channels [21], they are not widely used due to lack of inter-domain multicast support in the global Internet [30].

### 6.1.3 Application layer multicast

Because the lack of success of inter-domain IP multicast, an approach working at the application layer became plausible alternative, since there is no need for cooperation from each network provider on data paths. Applications are controlled by developers, and as long as all users deploy the same application, or they are interoperable, an overlay multicast distribution network can be built.

The most popular application layer streaming solutions [8, 9, 11, 60, 92] chose peer-to-peer technology to create the distrbution overlay network. However, as of today, most solutions are not interoperable, and each application creates its own overlay network. This is mainly due to the lack of standardization in the area, as the IETF only recently began to work on the Peer-to-Peer Streaming Protocol [73], which is still in very early design stages.

The proprietary PPLive system supports over 100,000 simultaneous users, providing about 400 channels with an average data rate of 325 kbps [37]. Once a user runs the PPLive application, it becomes a node in the P2P overlay, and queries the channel server for an updated channel list. Selecting a channel in the GUI results in a query for an online peer list for that particular channel, which includes IP addresses and port numbers. Those peers are then probed for liveness, and further peers are also learned through them. Chunks of streaming data is then downloaded from these peers into two local buffers: one for smooth playback and the other for upload.

Applications layer multicast however has its downsides too. Although it makes inter-domain live streaming more efficient than unicast, it still suffers from scalability problems. According to one study [87], surprisingly the streaming quality *degrades* as the number of peers *increases* if peak hours are considered. Further, server capacity still plays a key role in these systems, and its insufficient supply leads to low streaming quality. The main reasons behind these inefficiencies are the limited upload capacity of the peers, who usually access the Internet through asymmetric links, and churn (peers may join/leave the system at any moment).

Looking at all the above issues of the different streaming architectures, we believe a network layer solution that can work well over inter-domain boundaries is the way forward. The rest of this chapter is dedicated to presenting such a solution.

## 6.2 CoreCast Overview

CoreCast is a simple, one-to-many multicast protocol with low deployment cost, incrementally deployable, exploiting features introduced by LISP in order to reduce redundant traffic. To implement CoreCast, a only small number of modifications to the current LISP specification are required.

Consider the following scenario: a broadcaster has a large number of clients (denoted by $k$) for live streaming content or an IPTV channel. These clients are dispersed over a number of $j$ Autonomous Systems (ASes). When using unicast, the same content has to be sent $k$ times by the source node $S$ to reach all clients. Using CoreCast, the content is sent once to the ITR of the source node ($ITR_S$) along with a list of destinations, which in turn sends one copy to each of the involved ETRs ($ETR_1 \ldots ETR_j$), and these ETRs send out one copy to each destination node inside of their respective ASes. See Figure 6.4 for an example of a small CoreCast streaming deployment with 7 clients distributed in 3 ASes. On each inter-domain link, the multimedia data travels only once, so in the example case from the figure, the core will see it only 3 times instead of 7.

Note that authentication, authorization, and accounting (AAA) is not part of the CoreCast protocol, the source node $S$ should handle that at the application layer, using a framework of its choice. CoreCast is only concerned with the efficient transmission of the actual multimedia streams.

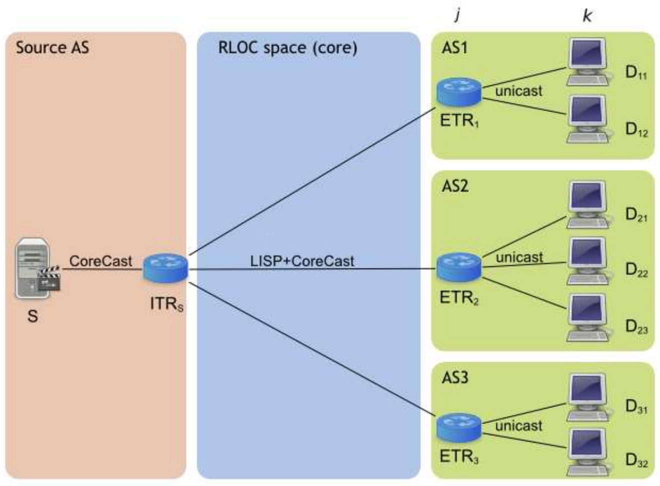


**Figure 6.4:** CoreCast architecture

## 6.3 Packet Types

CoreCast differentiates between two types of packets: *payload packets*, which carry the multimedia payload and *header packets*:

**Payload Packets** Contain the *hash* of the payload, which is the identifier used later by the caching and demultiplexing points, the *length* of the payload and the *payload* data itself.

**Header Packets** The CoreCast protocol data unit (content after the IP header) of this type of packets contains the *destination EID* of the client, and the *hash* that identifies the payload that has to be sent to it.

## 6.4 Source Node

As previously mentioned, $S$ implements AAA in a separate, out-of-band framework (e.g., HTTPS web login, or a custom protocol over TCP), and performs the appropriate mechanism for each connecting client. For each multimedia stream (channel) that it is broadcasting, $S$ maintains a data structure called `chanDstList` which contains a list of EIDs that are currently receiving that stream. After a successful connection, a new client is added to the requested channel's destination list. A client uses the same framework to signal departure from the channel, triggering removal from `chanDstList`. Departure may be caused by switching to a different channel, either on $S$ or a different source, or simply stopping to watch the stream. In order to account for clients not signaling departure, but no longer using the stream (due to system crash, network error, etc.), the AAA framework can implement a heartbeat protocol as well. Clients would be required to send periodically a control packet to show continued interest in the stream. A large interval of several minutes would take care of removing dead clients, while adding very little overhead.

Each domain reserves an EID space with local scope only for CoreCast streams, where an EID designates a channel. This is required because the first demultiplexing point is located on a busy border router, which should forward regular packets at line speed, and only work with CoreCast packets in the slow path. Since the destination EID is always examined, the reserved EID range is used to trigger examination of the CoreCast protocol fields, and leave all other packets in the fast path.

For each channel, $S$ divides multimedia data into chunks of payload in such a way, that packet size is maximized, but the MTU is not exceeded on the path to the destination. For each chunk, it first sends a packet with the payload, setting the destination address in the IP header to the channel's reserved EID. After the payload packet is sent, $S$ iterates through the `chanDstList`, and sends a header packet for each of the destinations listed. The process is then repeated at regular time intervals, determined by the bandwidth required for the stream. For example, sending a 384 Kbps stream, broken down into fixed sized payloads of 1200 bytes would require a payload packet to be sent every 25 ms. Optionally, the source could combine the list of destinations and instead of a header packet for each destination, send up to MTU sized packets with the list of destinations to the ITR. However, this would add complexity that is unlikely to be implementable in hardware at line speed, thereby actually reducing the number of clients supported by the router.

The above mechanism sets an upper limit of how many destinations CoreCast can support. This limit is function of the transmission medium's capacity, the bandwidth required by the stream, and payload size:

$$MaxClients_{CC} \simeq \frac{C \cdot T}{8 \cdot H_{CC}} = \frac{C}{BW} \cdot \frac{P}{H} \tag{6.1}$$

where $C$ is line rate in bits per second, $T$ is time between payloads in seconds, $P$ is payload packet size in bytes, $H$ is header packet size in bytes, and $BW$ the bandwidth required by the stream in bits per second. The same limit in case of unicast transmission is:

$$MaxClients_{UC} = \frac{C}{BW} \tag{6.2}$$

CoreCast's gain in terms of maximum number of supported clients depends on the ratio between the payload size and the header size: $P/H$. Using these formulae the content provider can do capacity planning based on the expected number of clients, and add network and/or server capacity as needed.

For our example in Figure 6.4, the source would send one CoreCast payload packet to the ITR followed by 7 CoreCast header packets for each PDU that has to be transmitted, placing destination EIDs $D_{ij}$ into the header.

When requesting a channel, the client software informs $S$ using the previously mentioned out-of-band AAA protocol if its domain supports CoreCast. For domains without a CoreCast capable ETR, the source will send regular unicast packets.

## 6.5 Ingress Tunnel Router

The ingress tunnel router is the first of the two CoreCast stream demultiplexing points. In order to process CoreCast packets, it maintains a `payloadBuffer` data structure, which reserves one record entry for each supported channel. The entry contains the hash of the payload, the payload itself, and a pointer to a `servedRLOC` buffer. This buffer is created for each payload on arrival, and tracks the locators of the ETRs which already received payload data (see Figure 6.5). When all clients from the domain of the locator have been served with the current payload, the buffer is freed. To avoid keeping too much state, the ITR keeps only one payload for each channel in the `payloadBuffer`.

Algorithm 2 shows the packet processing mechanism in the ingress tunnel router. When a *payload* packet is received, the ITR identifies the channel using the reserved destination EID in the IP header, checks if the hash in the CoreCast header matches the locally calculated hash of the payload, and then overwrites the old payload record in the `payloadBuffer`, also destroying the associated `servedRLOC` buffer and allocating a new one. No packet is sent upon receiving a payload packet.

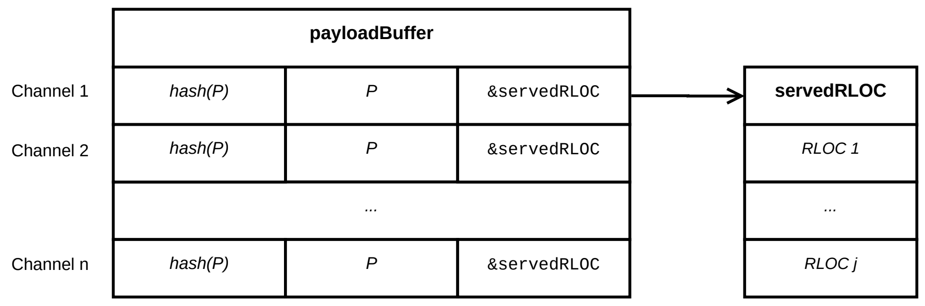


**Figure 6.5:** ITR Memory Structures. There is one `payloadBuffer` entry for each channel; a `servedRLOC` buffer is allocated for each payload on arrival and released after all clients have been served

**Algorithm 2** ITR Demultiplexing

```
if type = Payload then
  /* Payload packet */
  if hash = $hash$(payload) then
    Store to payloadBuffer
  else
    Drop
  end if
else if type = Header then
  /* Header only */
  if hash $\in$ payloadBuffer then
    if RLOC $\in$ servedRLOC then
      Send header to RLOC
    else
      Send payload to RLOC
      Send header to RLOC
      Store RLOC in servedRLOC
    end if
  else
    Drop
  end if
end if
```

For each CoreCast *header* packet, the ITR extracts the client EID and the payload hash from the CoreCast header and checks for the existence of the associated payload data in the `payloadBuffer`. Then, it looks up the locator of the client EID, which is an operation already provided by LISP. If the locator is listed in the `servedRLOC` buffer associated to the payload, the ITR forwards the header to the ETR, doing the usual LISP encapsulation. In the case no payload was yet sent to a particular locator, a payload packet is generated before forwarding the header packet.

Take the example in Figure 6.4. The ITR would store the payload, then send one copy to $ETR_1$ followed by headers for $D_{11}$ and $D_{12}$, another copy to $ETR_2$ and headers to destinations in $AS_2$, and finally a copy to $ETR_3$ and headers to destinations in $AS_3$.

In the case when the AS of the source node does not offer CoreCast protocol support in the ITR, but allows LISP encapsulated packets to be sent from the inside, the functions of the ITR could be performed by $S$ itself: it could send one LISP encapsulated payload to each locator and LISP encapsulated headers for the destinations. For this, it would need access to the LISP mapping system used by tunnel routers, which is allowed by the LISP specification.

Note that CoreCast is a unidirectional *live* streaming protocol; as such there is no feedback from the clients about lost packets and no retransmissions are being made. The expected deployment of the source node close to the ITR reduces the packet loss probability to a minimum. On the other hand, the path between the ITR and ETRs is more prone to losses, and loss of single payload packet will affect all users of the destination AS. To alleviate this problem, the ITR can be configured to interleave an additional payload packet after every $n$ header packet going to the same ETR. That way the loss of a payload packet affects a reduced set of clients.

## 6.6 Egress Tunnel Routers

The ETR is the second and last demultiplexing point, and it works similar to the ITR, storing the payload for each received stream. But instead of forwarding headers, it has to expand them to regular unicast packets that get delivered within the AS to their final destinations, by retrieving and adding the corresponding payload data from the `payloadBuffer`. The packet processing mechanism in the ETR is presented in Algorithm 3.

To complete our example from Figure 6.4, $ETR_2$ stores its copy of the payload, then for the following headers sends a unicast IP packet with the payload to $D_{21}$, $D_{22}$, and $D_{23}$.

**Algorithm 3** ETR Demultiplexing

```
if type = Payload then
  /* Payload packet */
  if hash = $hash$(payload) then
    Store to payloadBuffer
  else
    Drop
  end if
else if type = Header then
  /* Header only */
  if hash $\in$ payloadBuffer then
    Send reconstructed unicast packet to destination
  else
    Drop
  end if
end if
```

Note that the demultiplexing must not necessarily use unicast packets towards the final destinations. The domain may decide to use IP multicast internally instead, especially if live streaming services prove popular. IP multicast is not widely used in inter-domain scenarios, but it has been successfully used for distributing TV channels inside an ISP [26].

Both the ITR and the ETR have to keep some state for CoreCast operations. In the case of the ETR the only CoreCast specific data structure is the `payloadBuffer`, the size of which depends only on the number supported channels. For example, supporting up to 1000 channels, with a maximum payload size of 1200 bytes requires just over 1 MB of memory. The ITR adds one more data structure per channel, `servedRLOC`, to store the locator that received already the current payload. Each entry in this structure is either 4 bytes (IPv4 core) or 16 bytes (IPv6 core). For our example of 1000 channels, clients distributed in up to 1000 ASes and an IPv6 core would add an extra 16 MB to the memory requirements. We believe this is an acceptable amount of memory for a border router.

# 7

# Evaluation of CoreCast

## 7.1 Analytical Evaluation

Along with architectural benefits, CoreCast lowers inter-domain bandwidth requirements for ISPs providing live streaming services to their subscribers. Existing methods to deliver these services include IP multicast, unicast and application layer P2P systems. To evaluate our proposal, we take P2P systems as base for comparison, because unicast is the most inefficient solution and multicast is difficult to deploy in inter-domain scenarios [30]. In contrast, P2P live streaming systems are in wide use today.

In order to compare the data rate of CoreCast with that of current P2P applications we developed a formula for the former and an estimation method for the latter. Furthermore, we focus the analysis on inter-domain traffic, mainly because it is more expensive for ISPs. In fact this is a key issue and recently ISPs have shown their concerns because of the large amount of inter-domain traffic generated by P2P applications [88].

In the case of CoreCast, we consider inter-domain the traffic sent from the ITR to the ETRs and intra-domain the one flowing from all ETRs to all individual clients. In the following, the terms *inter-domain traffic* and *transit traffic* will be used interchangeably; also, *intra-domain traffic* and *local traffic* is to be interpreted as the same.

The total transit traffic generated by sending one payload is as follows:

$$TT_{CC} = (H_L + P_{CC}) \cdot j + (H_L + H_{CC}) \cdot k, \tag{7.1}$$

where $H_L$ is the size of a LISP header, $P_{CC}$ is the CoreCast PDU size for a payload packet, $H_{CC}$ is the size of a header packet, and $j$ and $k$ are the number of RLOCs and clients, respectively, as described in Chapter 6. Thus the corresponding bandwidth is

$$BW_{transit} = \frac{TT_{CC}}{T} \tag{7.2}$$

On the other hand the total local traffic is given by:

$$LT_{CC} = (H_I + D) \cdot k, \tag{7.3}$$

where $H_I$ is the size of an IP header, and $D$ is the size of one multimedia chunk.

$$BW_{local} = \frac{LT_{CC}}{T} \tag{7.4}$$

Due to the way P2P streaming applications work, the total traffic can only be determined if each participating node sends its statistics to a central server or we can capture at each node. The first approach is only available to the owners of the application software, while the second is unfeasible due to its scale.

Instead, we will show that CoreCast inter-domain traffic is very close to the theoretical minimum, thus it is unlikely that any P2P network configuration produces less traffic of this type.

The expression for inter-domain traffic for a single payload in CoreCast has been computed in Equation 7.1. We will compute the inter-domain traffic for a single payload in a P2P network of the same size. Such a network will contain $k + 1$ nodes (including the broadcaster) and $j + 1$ ASes, since the broadcaster is assumed to be an independent AS.

If we model the P2P network as a graph, in order to cast the payload to all nodes, the graph must be connected. Some of these connections between nodes would be inter-domain connections. Let us denote by $i$ the number of inter-domain arcs in the graph. Since all ASes must be connected, the minimum number of inter-domain arcs is $j$, so $i \geqslant j$.

The total transit traffic for a P2P (under LISP) can be written in terms of the $i$ arcs as

$$TT_{P2P} = (H_L + P_{CC}) \cdot i. \tag{7.5}$$

As far as the transit traffic is concerned, CoreCast is more efficient than the P2P network whenever $TT_{CC} \leqslant TT_{P2P}$. Substituting by the corresponding expressions, we obtain the following equivalent formulation:

$$\frac{H_L + H_{CC}}{H_L + P_{CC}} \leqslant \frac{i - j}{k}. \tag{7.6}$$

The difference $i - j$ is the total number of inter-domain arcs in the P2P that are not really essential. To ease the interpretation of the results we rewrite this difference in terms of $k$, so that $i - j = \alpha \cdot k$. The parameter $\alpha \in [0, k-1]$ has a straightforward interpretation: it is the average number of non-essential inter-domain connections per node. Using $\alpha$ in Equation 7.6, CoreCast produces less inter-domain traffic than a P2P when

$$\alpha \geqslant \frac{H_L + H_{CC}}{H_L + P_{CC}}. \tag{7.7}$$

The parameter $\alpha$ depends on the particular P2P system and is difficult to estimate. However for reasonable values of header and payload sizes, only for very small values of $\alpha$ is the inequality in Equation 7.7 not satisfied. For instance, the LISP header is 20 bytes long ($H_L = 20$), the CoreCast header has 60 bytes ($H_{CC} = 60$), and a popular payload size used by live streaming applications is 1200 bytes ($P_{CC} = 1200$, see [38]). In this case we have that $\alpha \geqslant 0.0656$. Any P2P system having more than 0.0656 non-essential inter-domain arcs per node would be less efficient than CoreCast in terms of bandwidth for a 1200 byte payload size. Even if we decrease the payload size to, e.g., 400 bytes, we get a very small $\alpha = 0.16$ lower bound, from which CoreCast is more efficient.

In the next section we present a measurement-based analysis of popular P2P applications from several vantage points, that help us providing plausible values for the parameters of Equation 7.7.

## 7.2 Experimental Evaluation

In this section we first describe the datasets collected for our evaluation and secondly we compare P2P live streaming bandwidth requirements to that of CoreCast.

### 7.2.1 Experimental Datasets

To obtain the experimental datasets needed for the evaluation we performed several measurement experiments using different P2P live streaming systems. We passively measured the network traffic of the application and saved the data for offline analysis. In particular, we collected three different datasets from a wide variety of well-distributed vantage points at two different live events:

**Set 1**: This dataset consists of traces collected at multiple different capture points situated in France and Japan, to obtain a global view of the P2P network. We passively measured the network traffic of TVAnts, which is a very popular P2P live streaming application.

Our measurement testbed is divided into two parts: one in France, the other in Japan. Since a large community of users employ P2P live streaming to watch live soccer games, we performed our measurement experiment during such kind of events that also exhibit a strong interest to be watched live. The measured event was a qualifying soccer match for the Olympic tournament with China vs. Vietnam on August 23, 2007.

During the experiment, all the PCs were running the TVAnts P2P live streaming application and WinDump to collect the packets. All the seven traces we collected have the same duration of 2:45h. This duration is slightly larger than a soccer game (105 minutes) because we wanted to capture all the events that may occur at the beginning or the end of the game. The traces show very similar properties: their average size and number of packets are 2.5 GB and 3 millions of packets respectively. All the traces count approximately 1,850 distinct IPs. More than 95% of the IPs of a trace are also present in the other traces. This suggests that we captured the full population of peers in the measured event. A detailed description of this dataset can be found in [75]. Throughout the study we refer to the individual traces in this dataset as FR1, FR2, FR3, FR4, JP1, JP2 and JP3.

**Set 2**: To obtain this dataset we performed a measurement experiment using the PPLive application, developed and popular especially in China. The event monitored was the memorial service of Michael Jackson, on July 7, 2009, during which we passively measured the network traffic of the application and saved the data for offline analysis in two different vantage points: our network lab at the Technical University of Catalonia in Barcelona, Spain and a home broadband network in Cluj-Napoca, Romania. The traces are denoted with ES and RO respectively in the following. This was a very popular live event, to the extent that the channel broadcasting it was advertised by the PPLive client to the users with pop-ups, and attracted a large number of users.

We identified 23,713 and 3,252 unique IP addresses in the RO and ES traces respectively, during the 2:43h and 3:23h timeframe we captured traffic. The RO trace is 40 minutes shorter, because traffic capture started later into the event. We attribute the difference in population to the restrictive firewall of the university, which caused a significant decrease in the number of different IP addresses observed in the trace. Additionally, multimedia traffic in the former is transported predominantly over UDP, while the latter over TCP. This seems to corroborate the assumption of the firewall, as TCP connection tracking is more reliable on stateful firewalls.

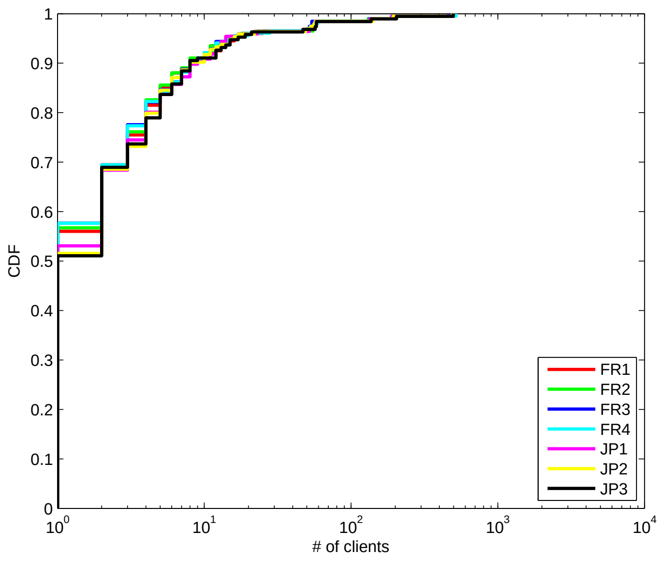
### 7.2.2 AS Distribution Analysis

Recalling from Equation 7.7 CoreCast saves bandwidth compared to existing solutions when these P2P applications use more than 0.16 ($\alpha$) inter-domain links per node on average. In order to better understand this parameter we study the amount of IPs contacted by each monitored node, and how these IPs are clustered among ASes. They are directly related to the amount inter-domain links used by each node. Besides the $\alpha$ parameter, the efficiency of CoreCast heavily depends on how clients viewing the stream are distributed into ASes. The worst case scenario for CoreCast is when each AS contains a single client, the best case being all the clients clustered in the same AS.

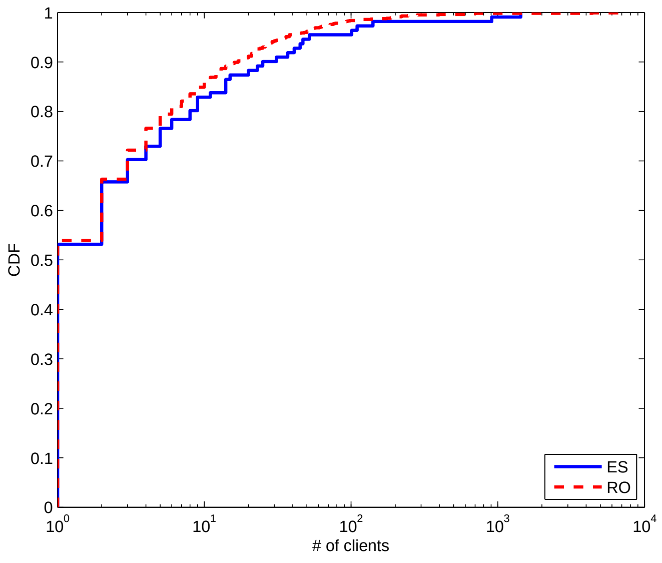
The 7 traces from *Set 1* had very similar characteristics, contacting about 1,850 distinct IP addresses, distributed among approximately 200 ASes. This results in almost 10 clients per AS on average, but the distribution of clients is not uniform. Fig 7.1(a) plots the cumulative distribution of the number of clients in ASes. We can see that for all data sets we have just over 50% of the ASes with a single client. These ASes would neither benefit from Corecast, nor be negatively impacted by its deployment. But there are a few domains containing a large number of clients, the largest population exceeding 500 viewers. Using CoreCast in these domains would result in a 500-fold decrease of expensive inter-domain traffic.

For the RO trace in *Set 2*, the 23,713 clients were distributed in 1291 ASes according to the CDF in Figure 7.1(b). It is worth noting that the first three ASes (all of them from China) captured 12,160 clients, which is more than half the total number of clients. These autonomous systems would have benefited the most from using CoreCast. In the ES trace, clients cluster even more in the top ASes: the top two (which are the same as for RO) contain 72% of all viewers.

According to these results, P2P clients use tens of inter-domain links on average, suggesting that $\alpha \gg 0.16$, a conservative lower bound for CoreCast's efficiency computed in § 7.1. Particularly the ASes that include a very large amount of clients would strongly benefit from the use of CoreCast.



(a) Dataset 1



(b) Dataset 2

**Figure 7.1:** Number of clients per autonomous system

### 7.2.3 Bandwidth Comparison

In this subsection we aim to estimate the total amount of inter-domain traffic saved with respect to P2P live streaming systems, based on actual measurement data. We also provide a comparison to unicast, using today's common parameters.

Typical servers today are equipped with 1 Gbps network cards, with some high end models supporting 10 Gbps interfaces. A common data rate for multimedia streams is 384 kbps. The maximum number of supported clients by CoreCast and unicast are given by Equations 6.1 and 6.2. A server with a 10 Gbps capacity, broadcasting a single channel, considering $P = H_L + P_{CC} = 20 + 1200$, $H = H_L + H_{CC} = 20 + 60 = 80$, will support $397,135$ clients when using CoreCast. Unicast would limit that number to only $26,041$, an order of magnitude smaller. Note that large content providers could horizontally scale to millions of viewers using several servers in a data center with 40 Gbps or 100 Gbps connectivity.

When comparing CoreCast bandwidth consumption to that of P2P applications, we need the average payload size, inter-packet arrival times and length of the broadcast to calculate the total CoreCast traffic. But to get the total P2P traffic, we either have to capture at each participating node, or have the client software on each node reporting statistics to a centralized measurement host. Since the first method is unfeasible due to its scale, and the second is only available to the owners of the P2P application, we need an alternative approach. To this end, we considered the *Set 1* data, captured using the TVAnts client software, and estimated the total traffic of a hypothetical, but plausible P2P system. To build this system, we assume that all peers ($c_{1 \ldots k}$) have a similar traffic profile. By traffic profile we refer to how the amount of bytes downloaded from the peers is distributed among them. This assumption seems reasonable when considering Figure 7.2, because the cumulative distribution functions for the 7 capture points almost completely overlap.

To estimate the total traffic, we determine the video download traffic (in bytes) of the monitored client in the trace from all contacted peers. We then create a two-dimensional traffic matrix with both columns and rows represented by peers $c_i$, and fill the row for $c_1$ with the data determined from the trace. Rows $c_2 \ldots c_k$ are random permutations of the same values. Iterating through each element $c_{ij}$ of the matrix,

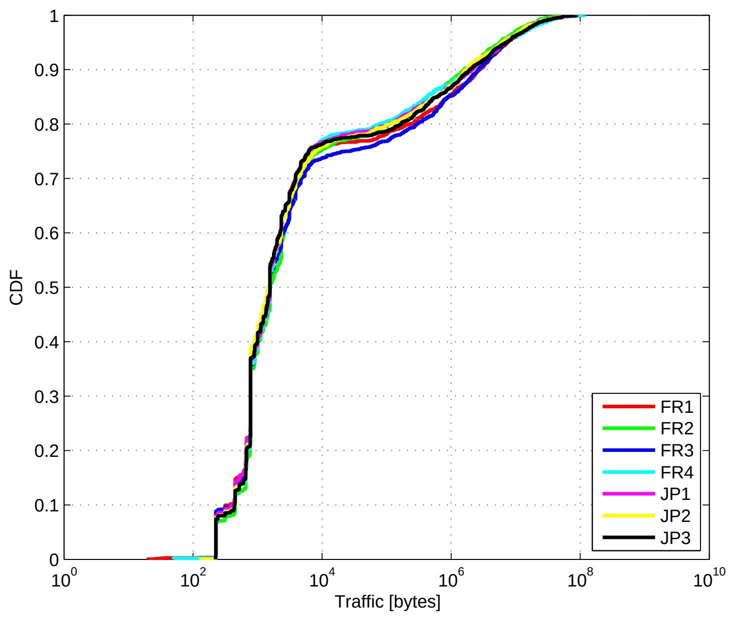


**Figure 7.2:** Cumulative distribution of the amount of traffic exchanged with peers from the 7 monitoring points of *Set 1*: they are almost identical

knowing the IP addresses of clients we determine if the communications was transit or local, and sum the amount of bytes to corresponding counter.

Table 7.1 shows the results for dataset 1. Column one specifies the trace collection point (except last row, representing CoreCast), while columns two and three describe the number of unique IP addresses and ASes per trace respectively. Columns TT and LT show the transit and local traffic, calculated with the algorithm described in the previous paragraph. It is worth noting here that we are not asserting that Table 7.1 presents an estimation of the total traffic produced by the applications running on the monitored networks. Instead, the values can be thought as the traffic generated by a representative P2P application, which is reasonable according to Figure 7.2. Using formulae 7.2 and 7.4 we also calculated the equivalent CoreCast traffic (last row), the results suggesting remarkable savings in terms of transit traffic with respect to P2P systems. On the other hand, local traffic volume increases when CoreCast is used. Since local traffic does not incur the same high costs as transit traffic, this should not be a problem for the destination domain. Moreover, using multicast internally would

| **Trace** | **IPs** | **ASes** | **TT** [GB] | **LT** [GB] |
|---|---|---|---|---|
| FR1 | 1855 | 209 | 1940 | 252 |
| FR2 | 1865 | 204 | 1472 | 195 |
| FR3 | 1769 | 201 | 1761 | 228 |
| FR4 | 1888 | 207 | 1974 | 268 |
| JP1 | 1856 | 201 | 1754 | 226 |
| JP2 | 1863 | 197 | 1645 | 215 |
| JP3 | 1878 | 194 | 1840 | 248 |
| CoreCast | 1853 | 202 | 131 | 895 |

**Table 7.1:** Estimated transit (TT) and local (LT) traffic of CoreCast, and a hypothetical P2P network, based on *Set 1*

bring traffic to the same level as the transit traffic, resulting in an improvement over P2P even for local traffic. In fact, several ISPs are already using intra-domain multicast to distribute video traffic [26]. Note that P2P cannot be optimized by using network layer multicast.

Figure 7.3 shows the cumulative distribution of the previously estimated P2P and CoreCast inter-domain traffic per autonomous system. As our analysis in § 7.2.2 showed, approximately 50% of domains have just one client. We can see in the figure that for these domains the bandwidth requirements of the P2P application and that of CoreCast are similar, the latter being slightly less efficient. Since CoreCast sends a separate payload packet and then a separate header packet, this overhead is to be expected.

The rest of the domains however save considerable amounts of traffic with CoreCast, and 10% of the them, in the case of this hypothetical P2P network, consume over an order of magnitude less of bandwidth (best case scenario is a 300-fold decrease). Considering that 10-fold increases in interconnect technologies are coming at a decreasing rate (3 years from 100Mbps to 1Gbps, 4 years for 1Gbps to 10Gbps and only draft standard 100Gbps in 8 years), this is a very important improvement.

In summary, the reversal of preference for intra-domain traffic in the case of CoreCast is beneficial for ISPs, because inter-domain traffic is more expensive. Additionally, while not all domains would benefit from CoreCast, 10% of domains could have reduced traffic with more than an order of magnitude for our datasets.

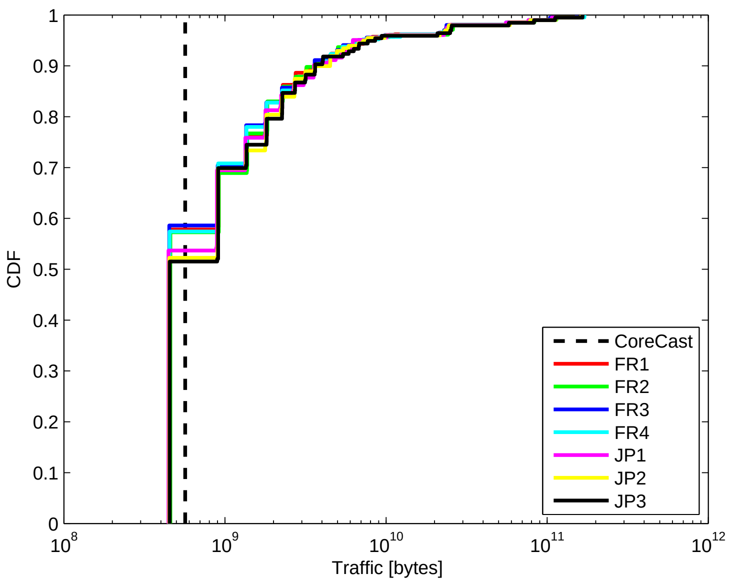


**Figure 7.3:** Cumulative distribution of the video download traffic per AS. For the slightly over 50% of ASes with only one peer CoreCast has a small overhead, but performs better in all other cases

## 7.3 Implementation

Because CoreCast introduces additional operations in the packet forwarding path on supported routers, it is important to quantify the processing overhead caused by these operations, as excessive overhead would discourage CoreCast adoption. In the following we quantify the increase in CPU load caused by CoreCast, compared to unicast forwarding.

### 7.3.1 Testbed

In order to test the processing overhead of the CoreCast protocol, we have implemented it as a Linux 2.6 loadable kernel module. This kernel module creates a Netfilter hook, which receives all IP packets before they are routed and applies Algorithm 1 to them. For the hashing function we first considered MD5, but finally chose SHA1, because the speed is comparable to MD5, but MD5 is now considered broken. The main disadvantage of SHA1 compared to the MD5 is the extra 4 bytes it adds to the CoreCast

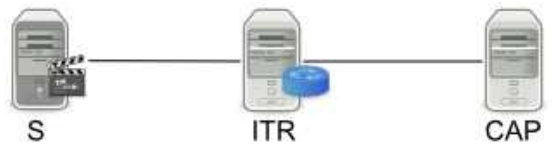


**Figure 7.4:** Testbed. *S* generates CoreCast traffic, *ITR* processes the packets, and *CAP* captures the resulting traffic

header. If this is a concern, the feasibility of removing the last 4 bytes of the SHA1 hash can be studied.

Due to the hashing, hash lookup and EID-to-RLOC lookup, CoreCast incurs an overhead compared to simple unicast packet forwarding. To quantify this overhead, we set up a small testbed with 3 machines (see Figure 7.4): one for generating packets ($S$), one for acting as a CoreCast capable $ITR$ and one to count received packets ($CAP$). All machines were the same hardware configuration: 3 GHz Pentium 4 processor, 1 GB of RAM and 2 on-board Intel Gigabit Network controllers. On the software side, they were running Debian GNU/Linux, with a 2.6.26 version of the Linux kernel. All machines were running only the bare minimum of the services necessary to run the experiments.

The first machine, *S*, was running a CoreCast traffic generator, which we implemented using raw sockets. For our experiments, we used the generator to send a typical 384 Kbps video stream with 1200 byte payloads every 25 ms, each payload packet being followed by header packets corresponding to a client list. To have realistic load on the *ITR*, in terms of distribution of clients among ASes, we used the client list from a trace with 1429 clients. The total duration of the stream was set to 30 seconds. The traffic generator also implements a scaling factor, which we use to gradually increase the load on the *ITR*. The scaling factor is a multiplier, specifying the up-scaling ratio for the total number of clients. The generator sends header packets to the number of clients present in the client list, multiplied by the scaling factor, using the same AS distribution.

We performed tests with scaling factors ranging from 1 (simulating 1429 clients) to 7 (10003 clients), repeating each test 20 times and averaging the results. To better understand the potential overhead incurred by CoreCast we performed the experiment for each scaling factor mixing unicast and CoreCast traffic. With this setup we have heterogeneous traffic. In particular we transmit 0%, 33%, 66% and 100% of CoreCast traffic, while the remainder is standard unicast traffic. The 0% case represents the baseline of unicast forwarding, since no CoreCast traffic is present. All the combinations resulted in performing a total of 560 experiments.

On the *ITR* we used the `sar` command to monitor the CPU usage during each 30 second test. Since LISP is still not implemented for Linux, we used a static EID-to-RLOC mapping cache, which was built based on the AS distribution from the Veetle trace. Finally, we captured all CoreCast packets on *CAP*, to determine packet loss.

### 7.3.2 Experimental results

Using the algorithm presented in the previous subsection, we obtained increasingly larger values for the number of packets sent per second by $S$ when increasing the scaling factor (see Figure 7.5). The increase was linear up to a scaling factor of 5 (equivalent of 7145 clients), at which point the hardware limitations of $S$ caused a slowdown of the increase in packet rate. It is worth to note that in this case $S$ generated a total of 285,080 packets per second, reaching 297,470 at the maximum scaling factor.

Figure 7.6 shows the evolution of the CPU usage on the *ITR*, while progressively increasing the scaling factor. The CPU usage shows a strong correlation with the packet rate and the percentage of CoreCast traffic.

CoreCast incurs a slightly higher CPU usage compared to simple unicast packet forwarding. The increase is only about 1% in absolute terms and has a linear dependence with respect to the amount of forwarded CoreCast packets. Note that when $S$ reaches its sending capacity, saturating the CPU, the *ITR*, which has the exact same hardware configuration, reaches only 3.5% CPU usage when routing 100% CoreCast packets.

To have a deeper understanding of the CPU usage of CoreCast, we selectively disabled the different parts of the *ITR* CoreCast kernel module and repeated the experiments. The parts disabled where the hash lookup, SHA1 hashing, and EID-to-RLOC lookup functions. As expected, the hash lookup and SHA1 hashing did not incur in any measurable overhead compared to the unicast forwarding. On the one hand the hash lookup is performed over a table of limited size, 1 entry for a single video-stream, and typically a server transmits a several streams (up to hundreds). Therefore this

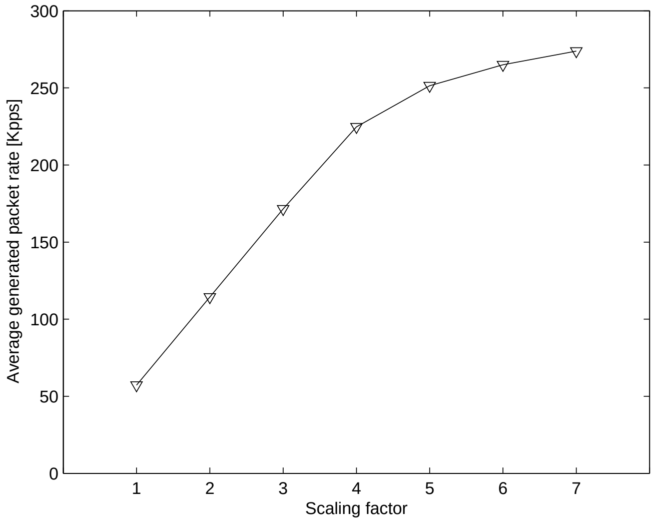


**Figure 7.5:** Average generated packet rate on *S*

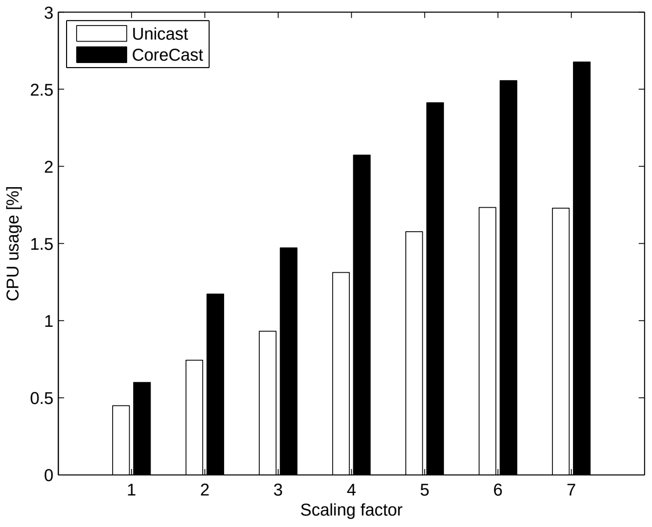


**Figure 7.6:** CPU usage on *ITR*

operation does not impact the CPU. On the other hand, the SHA1 hashing operations do not introduce any measurable overhead either. This is because standard, widely used hash functions are designed in such a way that they can be implemented in hardware, or highly optimized in software. As a result, the Linux kernel includes a highly optimized implementation of SHA1, which explains its excellent performance. Please note that other systems include similar optimizations, and on a production CoreCast capable router this hash could be implemented in hardware, basically eliminating the overhead caused by hashing.

We determined that the slight increase in CPU usage was caused by the EID-to-RLOC lookups. The implemented module includes a naïve linear search algorithm, clearly inefficient for this task. It work by iterating through the static EID-to-RLOC lookup table, until a match is found for the destination EID. For our table with 1429 entries an important overhead is introduced because of this operation. However, in a real router this would not represent an issue since a LISP router must perform such EID-to-RLOC lookups for each packet, independently of CoreCast. Because of this, production LISP routers implement such lookups in hardware using ternary content-addressable memory (TCAM) or in software using optimized radix trees.

Taking into consideration the experiments and the above reasoning we can conclude that the CoreCast protocol does not introduce any measurable overhead to packet processing on a production router.

## 7.4 Conclusions

CoreCast is a reliable network layer live streaming protocol. The main improvements over existing solutions are architectural: it is a simple protocol that provides one-to-many multicast in a future core/edge separated Internet. CoreCast circumvents the high deployment cost of previous network layer approaches because it works on top of LISP. Note that there is considerable consensus in the routing research community that the current routing system must be upgraded, and splitting the current address space is seen as the most scalable solution [67]. CoreCast has been implemented to operate on LISP, however it could operate on other proposals [48, 82] with minor modifications. Therefore, CoreCast's low deployment cost is not limited to LISP, but to any core/edge separation protocol deployed in the Internet.

The CoreCast architecture enables network planning for ISPs, because they can estimate the resources necessary for supporting a given number of streams. This further allows service level agreements between content providers and service providers, ensuring proper delivery of the content for the former and opening a new business opportunity to the latter.

Another contribution of the study is the Linux implementation of CoreCast. Using this implementation we saw a 52% increase in CPU usage when comparing unicast and CoreCast forwarding. The increase was determined to be caused by the EID-to-RLOC lookup function, which is part if the LISP protocol, and is a built-in feature of LISP routers. This operation is easily optimizable in hardware and will likely have negligible overhead in production equipment.

Additionally, our analytical model, combined with measurement data, suggests that compared to reasonable P2P live streaming systems, CoreCast produces significantly less inter-domain traffic. The gains depend on the distribution of clients among autonomous systems, with as much as 300-fold decrease observed in our datasets.

Part III

# Contributions to the LISP Community

# 8

# Deployment of LISP Network Elements

The Locator/Identifier Separation Protocol (LISP) addresses the scaling issues of the global Internet routing system by separating the current addressing scheme into Endpoint IDentifiers (EIDs) and Routing LOCators (RLOCs). The main protocol specification [32] describes how the separation is achieved, which new network elements are introduced, and details the packet formats for the data and control planes.

While the boundary between the core and edge is not strictly defined, one widely accepted definition places it at the border routers of stub autonomous systems, which may carry a partial or complete default-free zone (DFZ) routing table. The initial design of LISP took this location as a baseline for protocol development. However, the applications of LISP go beyond of just decreasing the size of the DFZ routing table, and include improved multihoming and ingress traffic engineering (TE) support for edge networks, and even individual hosts. Throughout this chapter we will use the term LISP site to refer to these networks/hosts behind a LISP Tunnel Router. We formally define it as:

**LISP site** A single host or a set of network elements in an edge network under the administrative control of a single organization, delimited from other networks by LISP Tunnel Router(s).

Since LISP is a protocol which can be used for different purposes, it is important to identify possible deployment scenarios and the additional requirements they may impose on the protocol specification and other protocols. The main specification mentions positioning of tunnel routers, but without an in-depth discussion. This chapter fills that gap, by exploring the most common cases. While the theoretical combinations of device placements are quite numerous, the more practical scenarios are given preference in the following.

Each subsection considers an element type, discussing the impact of deployment scenarios on the protocol specification. For definition of terms, please refer to the appropriate documents (as cited in the respective sections).

## 8.1 Tunnel Routers

LISP is a map-and-encap protocol, with the main goal of improving global routing scalability. To achieve its goal, it introduces several new network elements, each performing specific functions necessary to separate the edge from the core. The device that is the gateway between the edge and the core is called Tunnel Router (xTR), performing one or both of two separate functions:

1. Encapsulating packets originating from an end host to be transported over intermediary (transit) networks towards the other end-point of the communication
2. Decapsulating packets entering from intermediary (transit) networks, originated at a remote end host.

The first function is performed by an Ingress Tunnel Router (ITR), the second by an Egress Tunnel Router (ETR).

Section 8 of the main LISP specification [32] has a short discussion of where Tunnel Routers can be deployed and some of the associated advantages and disadvantages. This section adds more detail to the scenarios presented there, and provides additional scenarios as well.

### 8.1.1 Customer Edge

LISP was designed with deployment at the core-edge boundary in mind, which can be approximated as the set of DFZ routers belonging to non-transit ASes. For the purposes of this chapter, we will consider this boundary to be consisting of the routers connecting LISP sites to their upstreams. As such, this is the most common expected scenario for xTRs, and this chapter considers it the reference location, comparing the other scenarios to this one.

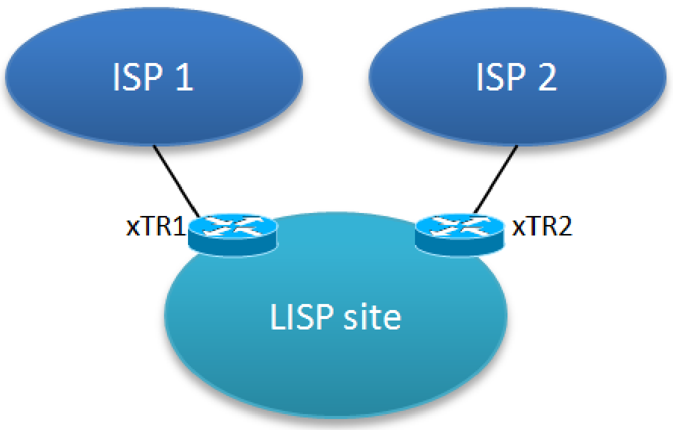


**Figure 8.1:** xTRs at the customer edge

From the LISP site perspective the main advantage of this type of deployment (compared to the one described in the next section) is having direct control over its ingress traffic engineering. This makes it is easy to set up and maintain active/active, active/backup, or more complex TE policies, without involving third parties.

Being under the same administrative control, reachability information of all ETRs is easier to synchronize, because the necessary control traffic can be allowed between the locators of the ETRs. A correct synchronous global view of the reachability status is thus available, and the Loc-Status-Bits can be set correctly in the LISP data header of outgoing packets.

By placing the tunnel router at the edge of the site, existing internal network configuration does not need to be modified. Firewall rules, router configurations and address assignments inside the LISP site remain unchanged. This helps with incremental deployment and allows a quick upgrade path to LISP. For larger sites with many external connections, distributed in geographically diverse PoPs, and complex internal topology, it may however make more sense to both encapsulate and decapsulate as soon as possible, to benefit from the information in the IGP to choose the best path (see Section 8.1.3 for a discussion of this scenario).

Another thing to consider when placing tunnel routers are MTU issues. Since encapsulating packets increases overhead, the MTU of the end- to-end path may decrease, when encapsulated packets need to travel over segments having close to minimum MTU. Some transit networks are known to provide larger MTU than the typical value of 1500 bytes of popular access technologies used at end hosts (e.g., IEEE 802.3 and 802.11). However, placing the LISP router connecting to such a network at the customer edge could possibly bring up MTU issues, depending on the link type to the provider as opposed to the following scenario.

### 8.1.2 Provider Edge

The other location at the core-edge boundary for deploying LISP routers is at the Internet service provider edge. The main incentive for this case is that the customer does not have to upgrade the CE router(s), or change the configuration of any equipment. Encapsulation/decapsulation happens in the provider's network, which may be able to serve several customers with a single device. For large ISPs with many residential/business customers asking for LISP this can lead to important savings, since there is no need to upgrade the software (or hardware, if it's the case) at each client's location. Instead, they can upgrade the software (or hardware) on a few PE routers serving the customers. This scenario is depicted in Figure 8.2

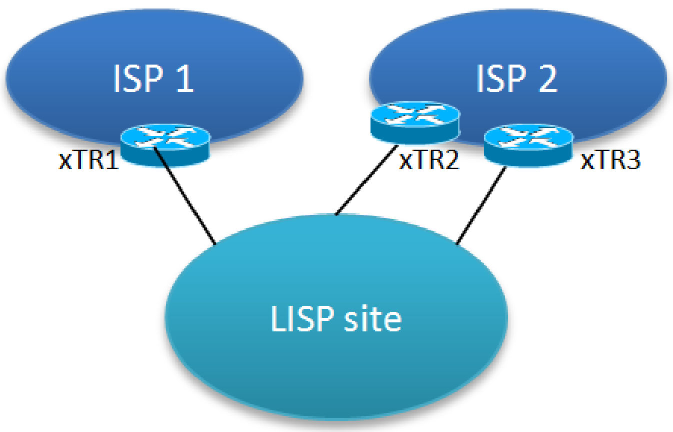


**Figure 8.2:** xTRs at the provider edge

While this approach can make transition easy for customers and may be cheaper for providers, the LISP site looses one of the main benefits of LISP: ingress traffic engineering. Since the provider controls the ETRs, additional complexity would be needed to allow customers to modify their mapping entries.

The problem is aggravated when the LISP site is multihomed. Consider the scenario in Figure 8.2: whenever a change to TE policies is required, the customer contacts both ISP1 and ISP2 to make the necessary changes on the routers (if they provide this possibility). It is however unlikely, that both ISPs will apply changes simultaneously, which may lead to inconsistent state for the mappings of the LISP site (e.g., weights for the same priority don't sum 100). Since the different upstream ISPs are usually competing business entities, the ETRs may even be configured to compete, either to attract all the traffic or to get no traffic. The former will happen if the customer pays per volume, the latter if the connectivity has a fixed price. A solution could be to have the mappings in the Map- Server(s), and have their operator give control over the entries to customer, much like in today's DNS.

Additionally, since xTR1, xTR2, and xTR3 are in different administrative domains, locator reachability information is unlikely to be exchanged among them, making it difficult to set Loc-Status- Bits correctly on encapsulated packets.

Compared to the customer edge scenario, deploying LISP at the provider edge might have the advantage of diminishing potential MTU issues, because the tunnel router is closer to the core, where links typically have higher MTUs than edge network links.

### 8.1.3 Split ITR/ETR

In a simple LISP deployment, xTRs are located at the border of the LISP site (see Section 8.1.1). In this scenario packets are routed inside the domain according to the EID. However, more complex networks may want to route packets according to the destination RLOC. This would enable them to choose the best egress point.

The LISP specification separates the ITR and ETR functionality and considers that both entities can be deployed in separated network equipment. ITRs can be deployed closer to the host (i.e., access routers). This way packets are encapsulated as soon as possible, and packets exit the network through the best egress point in terms of BGP policy. In turn, ETRs can be deployed at the border routers of the network, and packets are decapsulated as soon as possible. Again, once decapsulated packets are routed according to the EID, and can follow the best path according to internal routing policy.

In the following figure we can see an example. The Source (S) transmits packets using its EID and in this particular case packets are encapsulated at ITR1. The encapsulated packets are routed inside the domain according to the destination RLOC, and can egress the network through the best point (i.e., closer to the RLOC's AS). On the other hand, inbound packets are received by ETR1 which decapsulates them. Then packets are routed towards S according to the EID, again following the best path.

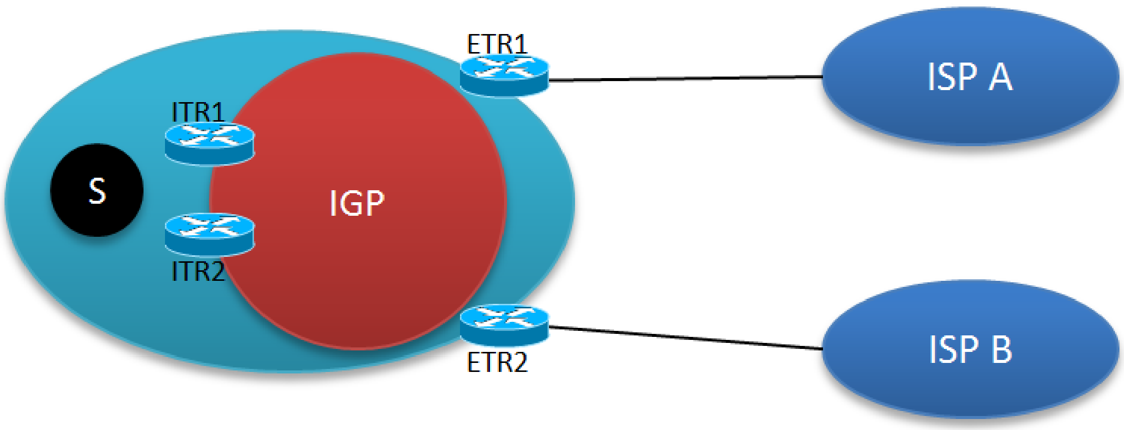


**Figure 8.3:** Split ITR/ETR scenario

This scenario has a set of implications:

- The site must carry at least partial BGP routes in order to choose the best egress point, increasing the complexity of the network. However, this is usually already the case for LISP sites that would benefit from this scenario.

- If the site is multihomed to different ISPs and any of the upstream ISPs is doing uRPF filtering, this scenario may become impractical. ITRs need to determine the exit ETR, for setting the correct source RLOC in the encapsulation header. This adds complexity and reliability concerns.

- In LISP, ITRs set the reachability bits when encapsulating data packets. Hence, ITRs need a mechanism to be aware of the liveness of ETRs.

- ITRs encapsulate packets and in order to achieve efficient communications, the MTU of the site must be large enough to accommodate this extra header.

- In this scenario, each ITR is serving fewer hosts than in the case when it is deployed at the border of the network. It has been shown that cache hit ratio grows logarithmically with the amount of users [49]. Taking this into account, when ITRs are deployed closer to the host the effectiveness of the mapping cache may be lower (i.e., the miss ratio is higher). Another consequence of this is that the site will transmit a higher amount of Map-Requests, increasing the load on the distributed mapping database.

### 8.1.4 Inter-Service Provider Traffic Engineering

With LISP, two LISP sites can route packets among them and control their ingress TE policies. Typically, LISP is seen as applicable to stub networks, however the LISP protocol can also be applied to transit networks recursively.

Consider the scenario depicted in Figure 8.4. Packets originating from the LISP site Stub1, client of ISP_A, with destination Stub4, client of ISP_B, are LISP encapsulated at their entry point into the ISP_A's network. The external IP header now has as the source RLOC an IP from ISP_A's address space (R_A1, R_A2, or R_A3) and destination RLOC from ISP_B's address space (R_B1 or R_B2). One or more ASes separate ISP_A from ISP_B. With a single level of LISP encapsulation, Stub4 has control over its ingress traffic. However, ISP_B only has the current tools (such as BGP prefix deaggregation) to control on which of his own upstream or peering links should packets enter. This is either not feasible (if fine-grained per-customer control is required, the very specific prefixes may not be propagated) or increases DFZ table size.

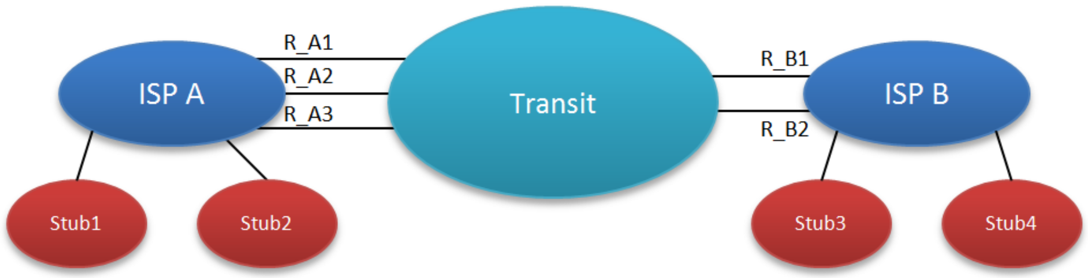


**Figure 8.4:** Inter-service provider TE scenario

A solution for this is to apply LISP recursively. ISP_A and ISP_B may reach a bilateral agreement to deploy their own private mapping system. ISP_A then encapsulates packets destined for the prefixes of ISP_B, which are listed in the shared mapping system. Note that in this case the packet is double-encapsulated. ISP_B's ETR removes the outer, second layer of LISP encapsulation from the incoming packet, and routes it towards the original RLOC, the ETR of Stub4, which does the final decapsulation.

If ISP_A and ISP_B agree to share a private distributed mapping database, both can control their ingress TE without the need of disaggregating prefixes. In this scenario the private database contains RLOC-to-RLOC bindings. The convergence time on the TE policies updates is fast, since ISPs only have to update/query a mapping to/from the database.

This deployment scenario includes two important recommendations. First, it is intended to be deployed only between two ISPs (ISP_A and ISP_B in Figure 8.4). If more than two ISPs use this approach, then the xTRs deployed at the participating ISPs must either query multiple mapping systems, or the ISPs must agree on a common shared mapping system. Second, the scenario is only recommended for ISPs providing connectivity to LISP sites, such that source RLOCs of packets to be reencapsulated belong to said ISP. Otherwise the participating ISPs must register prefixes they do not own in the above mentioned private mapping system. Failure to follow these recommendations may lead to operational and security issues when deploying this scenario.

Besides these recommendations, the main disadvantages of this deployment case are:

- Extra LISP header is needed. This increases the packet size and, for efficient communications, it requires that the MTU between both ISPs can accommodate double-encapsulated packets.
- The ISP ITR must encapsulate packets and therefore must know the RLOC-to-RLOC binding. These bindings are stored in a mapping database and may be cached in the ITR's mapping cache. Cache misses lead to an extra lookup latency.
- The operational overhead of maintaining the shared mapping database.

### 8.1.5 Tunnel Routers Behind NAT

NAT in this section refers to IPv4 network address and port translation provided by the ISP.

#### 8.1.5.1 ITR

Packets encapsulated by an ITR are just UDP packets from a NAT device's point of view, and they are handled like any UDP packet, there are no additional requirements for LISP data packets.

Map-Requests sent by an ITR, which create the state in the NAT table have a different 5-tuple in the IP header than the Map-Reply generated by the authoritative ETR. Since the source address of this packet is different from the destination address of the request packet, no state will be matched in the NAT table and the packet will be dropped. To avoid this, the NAT device has to do the following:

- Send all UDP packets with source port 4342, regardless of the destination port, to the RLOC of the ITR. The most simple way to achieve this is configuring 1:1 NAT mode from the external RLOC of the NAT device to the ITR's RLOC (Called "DMZ" mode in consumer broadband routers).
- Rewrite the ITR-AFI and "Originating ITR RLOC Address" fields in the payload.

This setup supports a single ITR behind the NAT device.

#### 8.1.5.2 ETR

An ETR placed behind NAT is reachable from the outside by the Internet-facing locator of the NAT device. It needs to know this locator (and configure a loopback interface with it), so that it can use it in the Map-Replies. Thus support for dynamic locators for the mapping database is needed in LISP equipment.

Again, only one ETR behind the NAT device is supported.

An implication of the issues described above is that LISP sites with xTRs can not be behind carrier based NATs, since two different sites would collide on the port forwarding.

### 8.1.6 Summary and Feature Matrix

Table 8.1 summarizes the effects of the different deployment scenarios on some LISP properties:

| **Feature** | **CE** | **PE** | **Split** | **Inter-ISP** |
|---|---|---|---|---|
| Control of ingress TE | ✓ | × | ✓ | ✓ |
| No modifications to existing internal networks | ✓ | ✓ | × | × |
| Loc-Status-Bits sync | ✓ | × | ✓ | ✓ |
| MTU/PMTUD issues minimized | × | ✓ | × | ✓ |

**Table 8.1:** Summary and feature matrix

## 8.2 Map-Resolvers and Map-Servers

### 8.2.1 Map-Servers

The Map-Server learns EID-to-RLOC mapping entries from an authoritative source and publishes them in the distributed mapping database. These entries are learned through authenticated Map-Register messages sent by authoritative ETRs. Also, upon reception of a Map-Request, the Map-Server verifies that the destination EID matches an EID-prefix for which it is responsible for, and then re-encapsulates and forwards it to a matching ETR. Map-Server functionality is described in detail in [35].

The Map-Server is provided by a Mapping Service Provider (MSP). A MSP can be any of the following:

- **EID registrar.** Since the IPv4 address space is nearing exhaustion, IPv4 EIDs will come from already allocated Provider Independent (PI) space. The registrars in this case remain the current five Regional Internet Registries (RIRs). In the case of IPv6, the possibility of reserving a /16 block as EID space is currently under consideration [46]. If granted by IANA, the community will have to determine the body responsible for allocations from this block, and the associated policies. For already allocated IPv6 prefixes the principles from IPv4 should be applied.

- **Third parties.** Participating in the LISP mapping system is similar to participating in global routing or DNS: as long as there is at least another already participating entity willing to forward the newcomer's traffic, there is no barrier to entry. Still, just like routing and DNS, LISP mappings have the issue of trust, with efforts underway to make the published information verifiable. When these mechanisms will be deployed in the LISP mapping system, the burden of providing and verifying trust should be kept away from MSPs, which will simply host the secured mappings. This will keep the low barrier of entry to become an MSP for third parties.

In all cases, the MSP configures its Map-Server(s) to publish the prefixes of its clients in the distributed mapping database and start encapsulating and forwarding Map-Requests to the ETRs of the AS. These ETRs register their prefix(es) with the Map-Server(s) through periodic authenticated Map-Register messages. In this context, for some LISP end sites, there is a need for mechanisms to:

- Automatically distribute EID prefix(es) shared keys between the ETRs and the EID-registrar Map-Server.
- Dynamically obtain the address of the Map-Server in the ETR of the AS.

The Map-Server plays a key role in the reachability of the EID-prefixes it is serving. On the one hand it is publishing these prefixes into the distributed mapping database and on the other hand it is encapsulating and forwarding Map-Requests to the authoritative ETRs of these prefixes. ITRs encapsulating towards EIDs under the responsibility of a failed Map-Server will be unable to look up any of their covering prefixes. The only exception are the ITRs that already contain the mappings in their local cache. In this case ITRs can reach ETRs until the entry expires (typically 24 hours). For this reason, redundant Map-Server deployments are desirable. A set of Map-Servers providing high-availability service to the same set of prefixes is called a redundancy group. ETRs are configured to send Map-Register messages to all Map-Servers in the redundancy group. To achieve fail-over (or load-balancing, if desired), current known BGP or DNS practices can be used on the LISP+ALT BGP overlay or in LISP-TREE respectively.

Additionally, if a Map-Server has no reachability for any ETR serving a given EID block, it should not originate that block into the mapping system.

### 8.2.2 Map-Resolvers

A Map-Resolver a is a network infrastructure component which accepts LISP encapsulated Map-Requests, typically from an ITR, and finds the appropriate EID-to-RLOC mapping by either consulting its local cache or by consulting the distributed mapping database. Map-Resolver functionality is described in detail in [35].

Anyone with access to the distributed mapping database can set up a Map-Resolver and provide EID-to-RLOC mapping lookup service. In the case of the LISP+ALT mapping system, the Map-Resolver needs to become part of the ALT overlay so that it can forward packets to the appropriate Map-Servers. For more detail on how the ALT overlay works, see [36]. In the case of LISP-TREE, there is no such requirement, as the root servers are publicly accessible by anyone, without prior configuration (as is required for LISP+ALT access).

For performance reasons, it is recommended that LISP sites use Map- Resolvers that are topologically close to their ITRs. ISPs supporting LISP will provide this service to their customers, possibly restricting access to their user base. LISP sites not in this position can use open access Map-Resolvers, if available. However, regardless of the availability of open access resolvers, the MSP providing the Map-Server(s) for a LISP site should also make available Map-Resolver(s) for the use of that site.

In medium to large-size ASes, ITRs must be configured with the RLOC of a Map-Resolver, operation which can be done manually. However, in Small Office Home Office (SOHO) scenarios a mechanism for autoconfiguration should be provided.

One solution to avoid manual configuration in LISP sites of any size is the use of anycast RLOCs for Map-Resolvers similar to the DNS root server infrastructure. Since LISP uses UDP encapsulation, the use of anycast would not affect reliability. LISP routers are then shipped with a preconfigured list of well know Map-Resolver RLOCs, which can be edited by the network administrator, if needed.

The use of anycast also helps improving mapping lookup performance. Large MSPs can increase the number and geographical diversity of their Map-Resolver infrastructure, using a single anycasted RLOC. Once LISP deployment is advanced enough, very large content providers may also be interested running this kind of setup, to ensure minimal connection setup latency for those connecting to their network from LISP sites.

While Map-Servers and Map-Resolvers implement different functionalities within the LISP mapping system, they can coexist on the same device. For example, MSPs offering both services, can deploy a single Map-Resolver/Map-Server in each PoP where they have a presence.

## 8.3 Proxy Tunnel Routers

### 8.3.1 P-ITR

Proxy Ingress Tunnel Routers (P-ITRs) are part of the non-LISP/LISP transition mechanism, allowing non-LISP sites to reach LISP sites. They announce via BGP certain EID prefixes (aggregated, whenever possible) to attract traffic from non-LISP sites towards EIDs in the covered range. They do the mapping system lookup, and encapsulate received packets towards the appropriate ETR. Note that for the reverse path LISP sites can reach non-LISP sites simply by not encapsulating traffic. See [53] for a detailed description of P-ITR functionality.

The success of new protocols depends greatly on their ability to maintain backwards compatibility and inter-operate with the protocol(s) they intend to enhance or replace, and on the incentives to deploy the necessary new software or equipment. A LISP site needs an interworking mechanism to be reachable from non-LISP sites. A P-ITR can fulfill this role, enabling early adopters to see the benefits of LISP, similar to tunnel brokers helping the transition from IPv4 to IPv6. A site benefits from new LISP functionality (proportionally with existing global LISP deployment) when going LISP, so it has the incentives to deploy the necessary tunnel routers. In order to be reachable from non-LISP sites it has two options: keep announcing its prefix(es) with BGP (see next subsection), or have a P-ITR announce prefix(es) covering them.

If the goal of reducing the DFZ routing table size is to be reached, the second option is preferred. Moreover, the second option allows LISP-based ingress traffic engineering from all sites. However, the placement of P-ITRs greatly influences performance and deployment incentives. The following subsections present the LISP+BGP transition strategy and then possible P-ITR deployment scenarios. They use the loosely defined terms of "early transition phase" and "late transition phase", which refer to time periods when LISP sites are a minority and a majority respectively.

#### 8.3.1.1 LISP+BGP

For sites wishing to go LISP with their PI prefix the least disruptive way is to upgrade their border routers to support LISP, register the prefix into the LISP mapping system, but keep announcing it with BGP as well. This way LISP sites will reach them over LISP, while legacy sites will be unaffected by the change. The main disadvantage of this approach is that no decrease in the DFZ routing table size is achieved. Still, just increasing the number of LISP sites is an important gain, as an increasing LISP/non-LISP site ratio will slowly decrease the need for BGP-based traffic engineering that leads to prefix deaggregation. That, in turn, may lead to a decrease in the DFZ size in the late transition phase.

This scenario is not limited to sites that already have their prefixes announced with BGP. Newly allocated EID blocks could follow this strategy as well during the early LISP deployment phase, if the costs of setting up BGP routing are lower than using P-ITR services, or the expected performance is better. Since this leads to an increase in the DFZ size, one of the following scenarios should be preferred for new allocations.

#### 8.3.1.2 Mapping Service Provider P-ITR Service

In addition to publishing their clients' registered prefixes in the mapping system, MSPs with enough transit capacity can offer them P-ITR service as a separate service. This service is especially useful for new PI allocations, to sites without existing BGP infrastructure, that wish to avoid BGP altogether. The MSP announces the prefix into the DFZ, and the client benefits from ingress traffic engineering without prefix deaggregation. The downside of this scenario is path stretch, which is greater than 1.

Routing all non-LISP ingress traffic through a third party which is not one of its ISPs is only feasible for sites with modest amounts of traffic (like those using the IPv6 tunnel broker services today), especially in the first stage of the transition to LISP, with a significant number of legacy sites. When the LISP/non-LISP site ratio becomes high enough, this approach can prove increasingly attractive.

Compared to LISP+BGP, this approach avoids DFZ bloat caused by prefix deaggregation for traffic engineering purposes, resulting in slower routing table increase in the case of new allocations and potential decrease for existing ones. Moreover, MSPs serving different clients with adjacent aggregable prefixes may lead to additional decrease, but quantifying this decrease is subject to future research study.

#### 8.3.1.3 Tier 1 P-ITR Service

The ideal location for a P-ITR is on the traffic path, as close to non-LISP site as possible, to minimize or completely eliminate path stretch. However, this location is far away from the networks that most benefit from the P-ITR services (i.e., LISP sites, destinations of encapsulated traffic) and have the most incentives to deploy them. But the biggest challenge having P-ITRs close to the traffic source is the large number of devices and their wide geographical diversity required to have a good coverage, in addition to considerable transit capacity. Tier 1 service providers fulfill these requirements and have clear incentives to deploy P-ITRs: to attract more traffic from their customers. Since a large fraction is multihomed to different providers with more than one active link, they compete with the other providers for traffic.

To operate the P-ITR service, the ISP announces an aggregate of all known EID prefixes (a mechanism will be needed to obtain this list) downstream to their customers with BGP. First, the performance concerns of the MSP P-ITR service described in the previous section are now addressed, as P-ITRs are on-path, eliminating path stretch (except when combined with LISP+BGP). Second, thanks to the direction of the announcements, the DFZ routing table size is not affected.

The main downside of this approach is non-global coverage for the announced prefixes, caused by the downstream direction of the announcements. As a result, a LISP site will be only reachable from customers of service providers running P-ITRs, unless one of the previous approaches is used as well. Due to this issue, it is unlikely that existing BGP speakers migrating to LISP will withdraw their announcements to the DFZ, resulting in a combination of this approach with LISP+BGP. At the same time, smaller new LISP sites still depend on MSP for global reachability. The early transition phase thus will keep the status quo in the DFZ routing table size, but offers the benefits of increasingly better ingress traffic engineering to early adopters.

As the number of LISP destinations increases, traffic levels from those non-LISP, large multihomed clients who rely on BGP path length for provider selection (such as national/regional ISPs), start to shift towards the Tier 1 providing P-ITRs. The competition is then incentivised to deploy their own service, thus improving global P-ITR coverage. If all Tier 1 providers have P-ITR service, the LISP+BGP and MSP alternatives are not required for global reachability of LISP sites. Still, LISP+BGP user may still want to keep announcing their prefixes for security reasons (i.e., preventing hijacking). DFZ size evolution in this phase depends on that choice, and the aggregability of all LISP prefixes. As a result, it may decrease or stay at the same level.

For performance reasons, and to simplify P-ITR management, it is desirable to minimize the number of non-aggregable EID prefixes. In IPv6 this can be easily achieved if a large prefix block is reserved as LISP EID space [46]. If the EID space is not fragmented, new LISP sites will not cause increase in the DFZ size, unless they do LISP+BGP.

#### 8.3.1.4 Migration Summary

The following table presents the expected effects of the different transition scenarios during a certain phase on the DFZ routing table size:

| **Phase** | **LISP+BGP** | **MSP** | **Tier-1** |
|---|---|---|---|
| Early transition | no change | slowdown increase | no change |
| Late transition | may decrease | slowdown increase | may decrease |
| LISP Internet | considerable decrease | | |

**Table 8.2:** DFZ routing table size during migration

It is expected that a combination of these scenarios will exist during the migration period, in particular existing sites choosing LISP+BGP, new small sites choosing MSP, and competition between Tier 1 providers bringing optimized service. If all Tier 1 ISPs have P-ITR service in place, the other scenarios can be deprecated, greatly reducing DFZ size.

#### 8.3.1.5 Content Provider Load Balancing

By deploying P-ITRs in strategic locations, traffic engineering could be improved beyond what is currently offered by DNS, by adjusting percentages of traffic flow to certain data centers, depending on their load. This can be achieved by setting the appropriate priorities, weights and loc-status-bits in mappings. And since the P-ITRs are controlled by the content provider, changes can take place instantaneously.

### 8.3.2 P-ETR

In contrast to P-ITRs, P-ETRs are not required for the correct functioning of all LISP sites. There are two cases, where they can be of great help:

- LISP sites with unicast reverse path forwarding (uRPF) restrictions, and

- LISP sites without native IPv6 communicating with LISP nodes with IPv6-only locators.

In the first case, uRPF filtering is applied at their upstream PE router. When forwarding traffic to non-LISP sites, an ITR does not encapsulate packets, leaving the original IP headers intact. As a result, packets will have EIDs in their source address. Since we are discussing the transition period, we can assume that a prefix covering the EIDs belonging to the LISP site is advertised to the global routing tables by a P-ITR, and the PE router has a route towards it. However, the next hop will not be on the interface towards the CE router, so non-encapsulated packets will fail uRPF checks.

To avoid this filtering, the affected ITR encapsulates packets towards the locator of the P-ETR for non-LISP destinations. Now the source address of the packets, as seen by the PE router is the ITR's locator, which will not fail the uRPF check. The P-ETR then decapsulates and forwards the packets.

The second use case is IPv4-to-IPv6 transition. Service providers using older access network hardware, which only supports IPv4 can still offer IPv6 to their clients, by providing a CPE device running LISP, and P-ETR(s) for accessing IPv6-only non-LISP sites and LISP sites, with IPv6-only locators. Packets originating from the client LISP site for these destinations would be encapsulated towards the P-ETR's IPv4 locator. The P-ETR is in a native IPv6 network, decapsulating and forwarding packets. For non-LISP destination, the packet travels natively from the P-ETR. For LISP destinations with IPv6-only locators, the packet will go through a P-ITR, in order to reach its destination.

For more details on P-ETRs see the interworking draft [53].

P-ETRs can be deployed by ISPs wishing to offer value-added services to their customers. As is the case with P-ITRs, P-ETRs too may introduce path stretch. Because of this the ISP needs to consider the tradeoff of using several devices, close to the customers, to minimize it, or few devices, farther away from the customers, minimizing cost instead.

Since the deployment incentives for P-ITRs and P-ETRs are different, it is likely they will be deployed in separate devices, except for the CDN case, which may deploy both in a single device.

In all cases, the existence of a P-ETR involves another step in the configuration of a LISP router. CPE routers, which are typically configured by DHCP, stand to benefit most from P-ETRs. To enable autoconfiguration of the P-ETR locator, a DHCP option would be required.

As a security measure, access to P-ETRs should be limited to legitimate users by enforcing ACLs.

# 9

# Monitoring the LISP Pilot Network with LISPmon

## 9.1 The LISP Pilot Network

One of the “founding beliefs” of the IETF is embodied in this quote by David Clark: “We reject kings, presidents and voting. We believe in rough consensus and running code” [41]. The “running code” part means that nothing demonstrates better the viability of a proposed protocol than a working implementation.

In the case of LISP, there are both closed and open source implementations available. The most complete implementation is for the Cisco NX-OS platform, the company’s new, data-center oriented router operating system. There is a less complete implementation for the more widely deployed Cisco IOS platform, but it is rapidly catching up with the previous one. Both implementations are closed source, with binaries available for Cisco customers.

An open source implementation exists for the FreeBSD 7.x operating system [47]. It is still a work in progress, but testing showed it was interoperable with the Cisco implementations.

LISP is designed to improve inter-domain routing. Because of this, testing the protocol needs a global test network to be built, where deploying the existing implementations will result in useful operational experience. That, in turn, drives changes in the protocol engineering details, while it is still in experimental stage.

The LISP Pilot Network (LPN) was thus started by the protocol designers. A small core network of ALT routers provides the mapping system, and participating sites connect to this core. The prefix 153.16.0.0/16 was allocated by RIPE NCC as the first experimental EID address space for the testbed. LISP sites received prefixes of lengths between /24 and /28 from this pool, administered by David Meyer from the University of Oregon, one of the LISP protocol designers. Allocation of prefixes is based on the region where the site is residing, see Table 9.1.

| **Region** | **EID Prefix** |
|---|---|
| ARIN | 153.16.0.0/19 |
| RIPE NCC | 153.16.32.0/19 |
| APNIC | 153.16.64.0/19 |
| AfriNIC | 153.16.96.0/19 |
| LACNIC | 153.16.128.0/19 |

**Table 9.1:** Allocation of EID prefixes

Figure 9.1 depicts the topology of the LPN on May 14, 2010.

The core is made up of a fully meshed LISP+ALT router topology, with participants from four of the five regions. In addition to being an ALT node, the APNIC router also offers Map-Resolver/Map-Server (MR-MS) functionality. The ARIN and RIPE NCC nodes have a more complex topology, allowing the existence of several MR-MS providers, connecting to the core ALT router.

Stub sites are at the lowest level of this hierarchy. They register their allocated prefixes to the immediately higher level infrastructure component, and their ETRs respond the Map-Requests routed to them by that component.

As it can be seen from the statistics provided on the figure, the LPN spans over 4 continents, 11 countries and 45 sites.

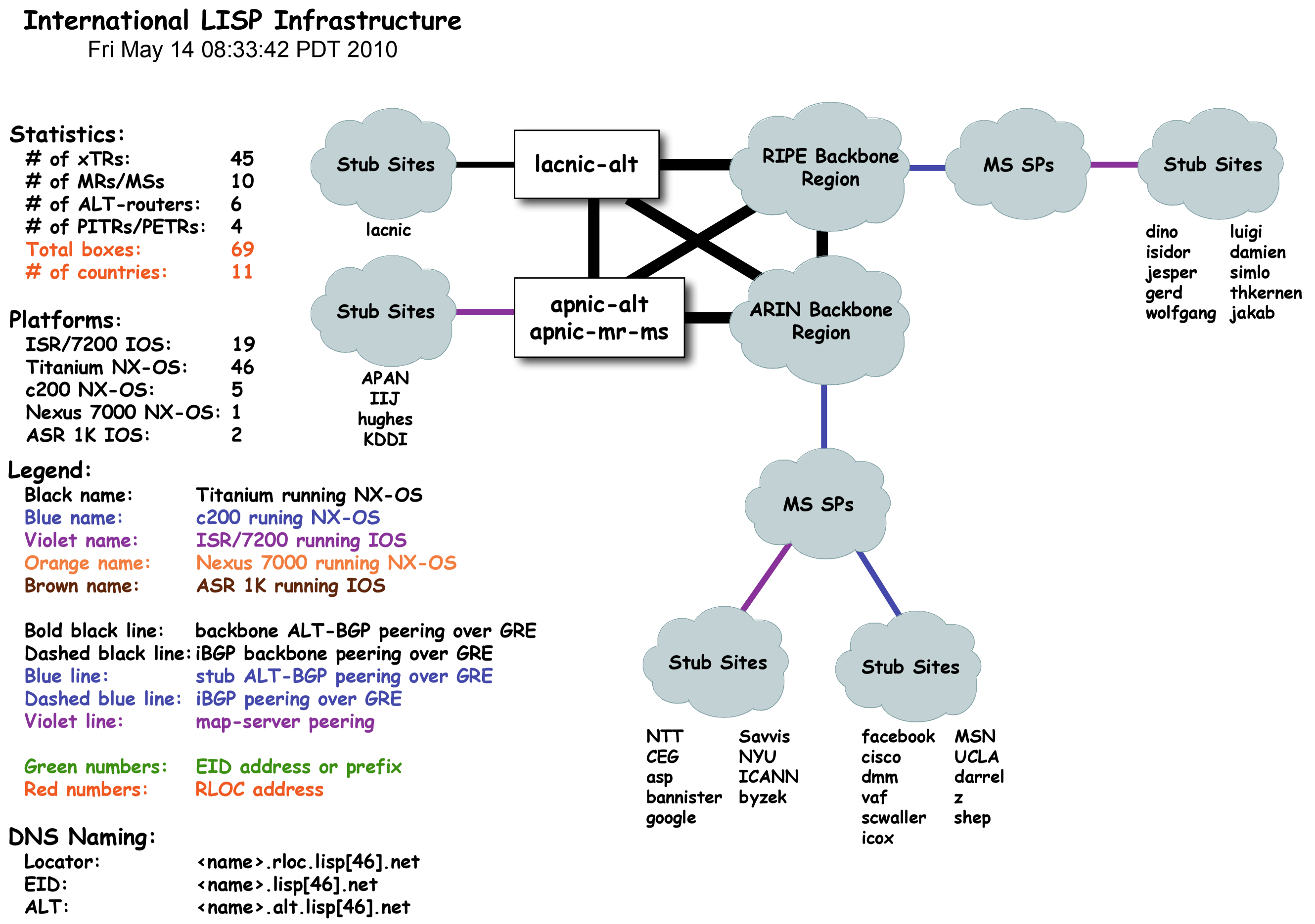

International LISP Infrastructure
Fri May 14 08:33:42 PDT 2010
Statistics:
# of xTRs: 45
# of MRs/MSs 10
# of ALT-routers: 6
# of PITRs/PETRs: 4
Total boxes: 69
# of countries: 11
Platforms:
ISR/7200 IOS: 19
Titanium NX-OS: 46
c200 NX-OS: 5
Nexus 7000 NX-OS: 1
ASR 1K IOS: 2
Legend:
Black name: Titanium running NX-OS
Blue name: c200 runing NX-OS
Violet name: ISR/7200 running IOS
Orange name: Nexus 7000 running NX-OS
Brown name: ASR 1K running IOS
Bold black line: backbone ALT-BGP peering over GRE
Dashed black line: iBGP backbone peering over GRE
Blue line: stub ALT-BGP peering over GRE
Dashed blue line: iBGP peering over GRE
Violet line: map-server peering
Green numbers: EID address or prefix
Red numbers: RLOC address
DNS Naming:
Locator: <name>.rloc.lisp[46].net
EID: <name>.lisp[46].net
ALT: <name>.alt.lisp[46].net
Stub Sites
lacnic
Stub Sites
APAN
IIJ
hughes
KDDI
lacnic-alt
apnic-alt
apnic-mr-ms
RIPE Backbone Region
ARIN Backbone Region
MS SPs
Stub Sites
dino
isidor
jesper
gerd
wolfgang
luigi
damien
simlo
thkernen
jakab
MS SPs
Stub Sites
NTT
CEG
asp
bannister
google
Savvis
NYU
ICANN
byzek
Stub Sites
facebook
cisco
dmm
vaf
scwaller
icox
MSN
UCLA
darrel
z
shep

**Figure 9.1:** LISP testbed (From www.lisp4.net. Used with permission)

## 9.2 LISPmon Implementation

The main goal of LISPmon is to monitor several aspects of the LISP Pilot Network (LPN). In particular:

- discover autonomously all sites connecting to the LPN including those outside the already allocated 153.16.0.0/16 EID prefix;
- get the mapping for discovered sites;
- determine geographical location and ISP of their tunnel routers.

In this context, autonomously means that site detection should work without manual intervention. As long as the LPN is small, this list can easily be maintained manually. However, when allocation of EID resources will become more decentralized, this will likely become challenging. LISPmon is designed from the beginning to scale to these future needs, being able to detect LISP-enabled sites without prior knowledge.

Collected data is then:

- presented in both human- and computer-friendly formats;
- stored for later analysis.

### 9.2.1 Architecture

In order to meet the goals described above, LISPmon is composed of three main building blocks, as depicted in Figure 9.2:

- Data collector backend
- Data store
- Presentation frontend

The data collector backend contains the logic necessary to gather monitored data from the testbed, and interfaces with both the data store and the presentation frontend. The data store receives gathered data from the data collector backend and stores it in a structured format for later retrieval. The machine-readable output module of the presentation frontend receives data directly from the data collector backend, while the human-friendly output module uses the data store to generate content. All architectural blocks are presented in detail in the following.

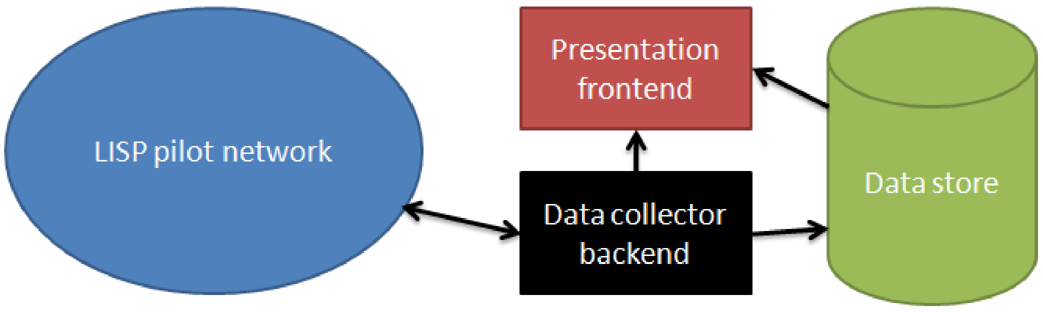


**Figure 9.2:** LISPmon architecture

### 9.2.2 Data Collector Backend

The data collector backend is the block responsible to gather monitored data.

#### 9.2.2.1 Collected Data

The cornerstone of the data gathering are the EID-to-RLOC mappings of registered EID prefixes. Such mappings include the following information:

- the EID prefix for which the mapping is valid,
- the time-to-live value of the mapping, in minutes,
- a boolean value (flag) indicating if the responding entity is an authoritative source,
- a boolean value (flag) indicating if the prefix is for mobile nodes,
- one record, for each locator of the prefix, containing:
  - the IP address (RLOC) of the tunnel router,
  - state of the tunnel router (up or down),
  - priority of the locator, and
  - weight of the locator.

After the data collector obtains the mapping for an EID prefix, it looks up additional data for each of the RLOCs:

- corresponding BGP prefix,

- BGP prefix allocation date,
- allocating Regional Internet Registry (RIR)
- the Autonomous System number and name,
- country code,
- approximate geographical coordinates,
- reverse DNS entry,
- round-trip time (RTT) from the monitoring server

The maintainers of the LPN also offer a publicly available status page of the testbed at `http://www.lisp4.net/status/`. This page reports from the vantage point of each Map-Resolver/Map-Server the following data:

- router name,
- router EID,
- router RLOC,
- Map-Request RTT when sending a query for the router EID to the MRMS in question from the monitoring station, and
- RTT from the monitoring station to the RLOC of the router using ICMP echo request/reply.

Each MRMS page is updated every hour. As an improvement to LISPmon, it this data is scraped from the web pages and is included in the data collection process. However, being an external source of data, which may become inaccessible, it is not an integral part of LISPmon, which can exist on its own without this data.

Section 9.2.3 details how this data is structured and stored in the data store. The rest of this subsection explains what each data element represents and which tools are used to obtain it.

**Mappings** The “glue” between the identifier and locator spaces is the LISP mapping system. For a detailed discussion of how the mapping system works, see Chapter 2.

Typically, it is the ITR that queries the mapping system. However, any host that can reach a Map-Resolver is able to send queries and receive responses. The **L**ISP **I**nternet **G**roper utility [33], which is open source, can be used by any *nix compatible host to query the mapping system. It sends an Encapsulated Map-Request to a Map-Resolver given on the command line or the shell environment variable `LISP_MAP_RESOLVER`, and receives and interprets the corresponding Map-Reply. A sample output of `lig` is as follows:

```
$ lig www.lisp4.net
Send map-request to 195.50.116.18 for www.lisp4.net ...
Received map-reply from 207.98.65.94 with rtt 0.25600 secs

Mapping entry for EID 153.16.10.6:
153.16.10.0/24, via map-reply, record ttl: 1440, auth, not mobile
  Locator                                 State     Priority/Weight
  128.223.156.134                         up        1/50
  207.98.65.94                            up        1/50
  2001:468:d01:9c::80df:9c86              up        2/100
```

The first line in the output of `lig` shows the destination Map-Resolver and the EID to be looked up. In this case, the address of the Map-Resolver was configured using the shell variable and was not specified on the command line. `lig` accepts, and translates, hostnames to IP addresses normally, and considers the result the EID to be looked up, as shown in the fourth line.

The second line shows where the response came from, in this case it is not the Map-Resolver, so we can assume for now it is one of the destination site’s ETR. The query was resolved in 256 milliseconds.

Next is the mapping entry. The query is for one single EID, but the response returns the covering EID prefix, as expected. For the example query we have 153.16.10.0/24 containing 153.16.10.6. This is useful for caching: the response is typically cached in an ITR, and for packets destined for IPs different from 153.16.10.6, but still covered by the prefix no lookups are required. The mapping entry also specifies the maximum validity of the mapping in the cache, in this case 1440 minutes or 1 day (the recommended value by the protocol [32]). Further, it is confirmed that a site ETR responded, because the mapping is marked authoritative. Additionally, the prefix is not associated to mobile nodes.

The last three lines are the locator records associated to the mapping. This site has two IPv4 locators and one IPv6 locator. Once again, it is confirmed that an ETR responded, because we find the replying IP address in the list of locators.

The priority/weight values describe the ingress traffic engineering policies of the LISP site. Lower priority values mean higher priority. Locators with the same priority receive traffic load balanced according to weights. For the above example, the policy is:

- the 2 IPv4 locators have the same highest priority, and must be used, unless no IPv4 connectivity is available at the encapsulating ITR;
- traffic towards the 2 locators must be load balanced, with a 50-50% share each;
- if no IPv4 connectivity is available, fall back to the lower priority IPv6 locator.

**Locator Allocation Data** Once locators for an EID prefix are determined, additional information about them can be discovered.

RLOCs are present in the DFZ global routing table, and are covered by **BGP prefixes**, allocated by **Regional Internet Registries** to **Autonomous Systems** with the **allocation date** recorded. One tool that helps determine these attributes is the Team Cymru IP-to-ASN mapping service [10].

Team Cymru uses several publicly available databases, published by RIRs and LIRs to create a single centralized query interface where information about BGP prefixes can be looked up. The data can be queried over several different protocols: WHOIS, DNS, and HTTP/HTTPS. The first two can be used programmatically, while HTTP/HTTPS is web form designed for human users.

Due to simplicity and flexibility reasons, LISPmon uses the WHOIS interface to the Cymru database. To continue the previous example, the lookup for locator 207.98.65.94 gives the following results:

```
$ whois -h whois.cymru.com " -c -r -u -a -p 207.98.65.94"
AS      | IP               | BGP Prefix          | CC | Registry | Allocated  | AS Name
3701    | 207.98.65.94     | 207.98.64.0/18      | US | arin     | 1996-04-08 | NERONET
- Oregon Joint Graduate Schools of Engineering
```

**Locator Geographical Data** If an IP address can be geolocated, then tunnel routers composing the LPN can be represented on a geographical map. MaxMind is a company specialized on geolocation services, and apart from commercial offerings, they make a free version of their IP-to-coordinates database available as well. This database is updated every month, and can be downloaded as a single file (either binary or CSV format).

The determine the release date:

```
$ geoiplookup -v -f GeoLiteCity.dat 207.98.65.94
GeoIP City Edition, Rev 1: GEO-533LITE 20100601 Build 1 Copyright (c) 2010
MaxMind Inc All Rights Reserved
```

Next, a lookup is performed for the locator:

```
$ geoiplookup -f GeoLiteCity.dat 207.98.65.94
GeoIP City Edition, Rev 1: US, OR, Eugene, 97403, 44.036400, -123.054703, 801, 541
```

The database returns country, state/region, ZIP code, latitude, longitude, and two codes used internally (metroCode and areaCode).

Note that locations returned by the database are approximate, MaxMind lists a "99.5% accuracy on a country level and 79% on a city level for the US within a 25 mile radius" for the free version [7].

**Reverse DNS** Reverse DNS entries are interesting in that they may offer additional information about the router, if the naming scheme follows a certain convention. Some naming conventions are obvious, others can only be decoded by the network administrators. LISPmon makes no attempt to decode the rDNS entry, it is only collected (and presented) for convenience.

There are several tools available to do (reverse) DNS lookups, such as *nslookup*, *host*, and *dig*. *dig* was chosen because its flexibility and many features, but any of them would have worked.

For our example locator, the rDNS entry is:

```
$ dig -x 207.98.65.94 +short
uo-pxtr.rloc.lisp4.net.
```

This particular entry follows the naming scheme presented in Figure 9.1: it is an RLOC in the lisp4 testbed and it is a Proxy-xTR (pxtr) at the University of Oregon (uo). The other locator points to the following rDNS entry:

```
$ dig -x 128.223.156.134 +short
titanium-dmm.lisp.uoregon.edu.
```

This no longer follows the *lisp4.net* naming scheme, but it is also part of the University of Oregon network. Additionally the word *titanium* suggests it is the new codename "Titanium" router from the Cisco Nexus line.

These examples show how rDNS entries can provide interesting information about the routers.

**Round-trip Time** For completeness sake, the round-trip time value from the monitoring station to the locator is also collected, by sending ICMP echo request packets:

```
$ ping -c 3 207.98.65.94
PING 207.98.65.94 (207.98.65.94) 56(84) bytes of data.
64 bytes from 207.98.65.94: icmp_seq=1 ttl=238 time=232 ms
64 bytes from 207.98.65.94: icmp_seq=2 ttl=238 time=232 ms
64 bytes from 207.98.65.94: icmp_seq=3 ttl=238 time=232 ms

--- 207.98.65.94 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 232.209/232.239/232.260/0.556 ms
```

#### 9.2.2.2 Crawling the LISP Mapping System

One of the main features of LISPmon is the auto-discovery of EID prefixes in the global Internet. Although the LPN is a "beta-test" LISP deployment, it is not an isolated (overlay) network, using private addresses. Rather, it works over, and integrates with the public Internet, and may grow to be an important part of of this infrastructure.

Auto-discovery of EID prefixes means scanning all the 4 billion IP addresses to determine if they are EIDs or not. In practice, this does not involve 4 billion Map-Requests, because Map-Replies contain prefixes, which usually cover more than one address (although /32 prefixes are allowed as well).

After sending a Map-Request, three cases can be distinguished:

**Positive Map-Reply** In this case, the prefix covering the address is returned, which is then known to be EID space;

**Negative Map-Reply** In this case, the largest contiguous power of two address block covering the queried address is returned, which is known to be RLOC space;

**No response** In this case, if the Map-Resolver is reachable, the queried address is most likely an EID, but there is no information about the size of the covering prefix.

The above cases form the basis of the LISPmon EID space discovery engine, which works according to Algorithm 4.

Since 0.0.0.0/8 is reserved by IANA, discovery starts at 1.0.0.0. A mapping lookup is preformed:

```
$ lig 1.0.0.0
Send map-request to 195.50.116.18 for 1.0.0.0 ...
```

**Algorithm 4** LISPmon EID space discovery engine

```
/* Initialize variables */
$ip = 1.0.0.0$
$granularity = /28$
while $ip < 255.255.255.255$ do
   $m$ =get_mapping($ip$)
   if $m.type = positive$ OR $m.type = negative$ then
      save_data()
      /* Get first IP after covered prefix*/
      $ip = m.$get_last_ip() $+ 1$
   else
      /* No reply was received */
      $ip = ip + 2^{32-granularity}$
   end if
end while
```

```
Received map-reply from 195.50.116.18 with rtt 0.04600 secs

Mapping entry for EID 1.0.0.0:
0.0.0.0/5, via map-reply, record ttl: 15, not auth, not mobile
  Negative cache entry, action: forward-native
```

A negative reply is received, for the whole 0.0.0.0/5 prefix, covering $2^{27}$ IPs. Because of this, it was possible to determine with a single mapping lookup that approximately 134 million IPs are from the locator space. The next IP to be looked up is the one immediately following this prefix: 8.0.0.0:

```
$ lig 8.0.0.0
Send map-request to 195.50.116.18 for 8.0.0.0 ...
Received map-reply from 195.50.116.18 with rtt 0.04300 secs

Mapping entry for EID 8.0.0.0:
8.0.0.0/7, via map-reply, record ttl: 15, not auth, not mobile
  Negative cache entry, action: forward-native
```

Again, a negative entry, covering about 33 million IPs. The process is repeated until the whole IPv4 address space is covered.

For positive replies, the process is the same, except the data must be extracted:

```
$ lig 153.16.0.0
Send map-request to 195.50.116.18 for 153.16.0.0 ...
Received map-reply from 198.6.255.39 with rtt 0.14800 secs
```

```
Mapping entry for EID 153.16.0.0:
153.16.0.0/24, via map-reply, record ttl: 1440, auth, not mobile
  Locator                                State     Priority/Weight
  198.6.255.39                           up        1/100
```

The third case is when no reply is received:

```
$ lig 153.16.3.0
Send map-request to 195.50.116.18 for 153.16.3.0 ...
Send map-request to 195.50.116.18 for 153.16.3.0 ...
Send map-request to 195.50.116.18 for 153.16.3.0 ...
*** No map-reply received ***
```

There may be three causes for this:

- the Map-Resolver is not reachable;
- the ALT is not functional – however, if for other EIDs replies are received during a crawl, both cases are unlikely;
- the covering prefix is registered as EID space, but the none of the site's ETRs are reachable.

The crawler always assumes the last case. Since no prefix is received, for a complete crawl it would be necessary to advance one IP at a time. That is very slow, however and because of that LISPmon has a configurable granularity parameter, specifying the step size for unresponsive EID space. Currently the smallest EID prefix used on the LPN is /28, consisting of 16 IP addresses, and this value was chosen for LISPmon.

There are two more variables affecting discovery speed, and they are both related to unresponsive EIDs:

- *Timeout* – the time in seconds after which a Map-Request is considered not replied
- *Number of retries* – the number of Map-Requests *lig* sends before giving up

By default *lig* used a timeout of 2 seconds and retries a query 3 times. LISPmon reduced the number of retries to 2, to decrease discovery time, as experiments showed no impact on accuracy. However, both parameters can be easily modified, if the need arises.

LISPmon parameters are summarized in Table 9.2. Using those parameters yields a discovery time for the complete IP address space of about 2 hours. Note that if the EID prefix density increases, the time it takes to scan the LPN also increases, while keeping the configuration parameters constant.

| Parameter | *lig* default | LISPmon value |
|---|---:|---:|
| Granularity | N/A | /28 |
| Timeout | 2 s | 2 s |
| Retries | 3 | 2 |

**Table 9.2:** LISPmon discovery engine parameters

### 9.2.3 Data Store

Once data is acquired, it needs to be stored efficiently and in way that permits its creative use later. Text files (e.g., in CSV format) are easy to work with, compress well, but are not very flexible in terms of easy data mining. Because of this, a SQL based relational database was chosen to support the data store of LISPmon.

Relational database packages considered were the popular, widely available open source projects SQLite, MySQL and PostgreSQL. SQLite is a very lightweight SQL database (as its name suggests), using a single file for a database and requiring no server process for access. MySQL is more complex relational database management system (RDBMS) with many features, storage engines for different needs, widely used for web programming. PostgreSQL is not as popular as MySQL, but it has even more features.

The choice of which package to use was ultimately decided by support for network address data types. Most data in LISPmon has to do with IP addresses and prefixes, and native support for this data was preferred. Although it is possible to work with 32-bit integers for IPv4, support for prefix length needs extra implementation work, and IPv6 is more complex.

PostgreSQL offers native support for IP addresses and prefixes with the `inet` data type. The `inet` type holds an IPv4 or IPv6 host address, and optionally its subnet, all in one field. Additionally PostgreSQL offers several operators and functions to work with these data types, making it easy to implement checks such as the inclusion of a prefix in another one, getting the first or last address in a prefix, or arithmetic operations on addresses.

Because of the reasons described above, the RDBMS of choice was PostgreSQL.

### 9.2.4 Presentation Frontend

As the name suggests, the presentation frontend is tasked to exposed the data gathered and stored by LISPmon to users. Since it is functionally separated by the other modules of LISPmon, it is easy to add, remove or modify frontends, without affecting the rest of the infrastructure. In turn, the other modules can only be modified in a backward compatible way, meaning no functionality can be removed without affecting the other modules.

LISPmon integrates two different ways of presenting the data into the project web site:

- machine parsable plain text files – using a custom CSV format, daily mapping reports and weekly rDNS reports;
- an interactive router placement map – based on the Google Maps API, it shows approximate locations of discovered LISP tunnel routers on geographical map, with marker popups showing additional information.

Additionally, there is an interface to perform live queries on the LISP mapping system, called a "looking glass". It allows users to select one of the available Map-Resolvers, specify a host name or IP address and perform the mapping lookup. A CGI script executes *lig* with the chosen parameters and presents it's output on the web page. See Section 9.3.3 for an example.

### 9.2.5 Putting it All Together

LISPmon is hosted on an old desktop-class machine with an Intel Pentium III 866 MHz processor and 512 MB of RAM, which is shared with projects. It is located at the laboratories of the Advanced Broadband Communication Center at the Technical University of Catalonia, where the project was developed. The machine runs Gentoo Linux, with the *2.6.34-gentoo-r1* kernel.

The database backend is PostgreSQL version 8.4, the latest stable version available.

Job scheduling is done by the *cron* daemon offered by the operating system. Each day at 09:00h a *Bash* script is executed to do the main data gathering, by means of the following *crontab* entry:

```
0 9 * * *       /var/www/localhost/htdocs/lispmon/code/explore_lisp4_mappings.sh
```

The status pages from the LPN are updated about every hour, but in order to avoid missing updates due to clock skew, and to discover changes faster, they are scraped every half hour:

```
*/30 * * * *    /var/www/localhost/htdocs/lispmon/code/dmm/getdmm
```

The weekly rDNS report is generated every Saturday at 13:00h:

```
0 13 * * 6      /var/www/localhost/htdocs/lispmon/code/explore_lisp4_dns.sh
```

The data gathering work is done by scripts written in Perl. Perl was chosen because it's one of the most versatile languages when it comes to text output processing, and the LISPmon platform is based on processing output from a variety of tools. The above Bash scripts launch the Perl code doing the actual work and check for errors. Existing files are only overwritten if the Perl script finishes correctly. In either case, an email is sent to the administrator of LISPmon: if no errors occurred, the changes from the previous run are sent, otherwise the error is reported.

The Perl code implements the algorithms described in Section 9.2.2.

## 9.3 Experimental Results

### 9.3.1 Text Files

#### 9.3.1.1 Mappings

Table 9.3 presents a summary of the mappings discovered on June 9, 2010:

| | |
|---|---:|
| EID prefixes | 26 |
| Covered EID addresses | 4976 |
| Live EID addresses | 38 |
| RLOC addresses | 43 |
| Avg. RLOC addresses per EID prefix | 1.65 |
| Max. RLOC addresses per EID prefix | 4 |

**Table 9.3:** Mapping statistics for June 9, 2010

During the period between January 7, 2010 and June 19, 2010, the maximum number of active prefixes was 32, reached on May 26. The minimum number during the same period was observed on June 5, caused by an incompatible change in the LISP control packet format. A "flag day" was scheduled to upgrade all implementations to the new format. Since not everyone was able to do on that specific day, the number of active prefixes was low, and recovered only after a few days ("flag day“ was during a weekend). The average value for the whole period was 27.2 prefixes. See Figure 9.3 for the evolution of active prefixes.

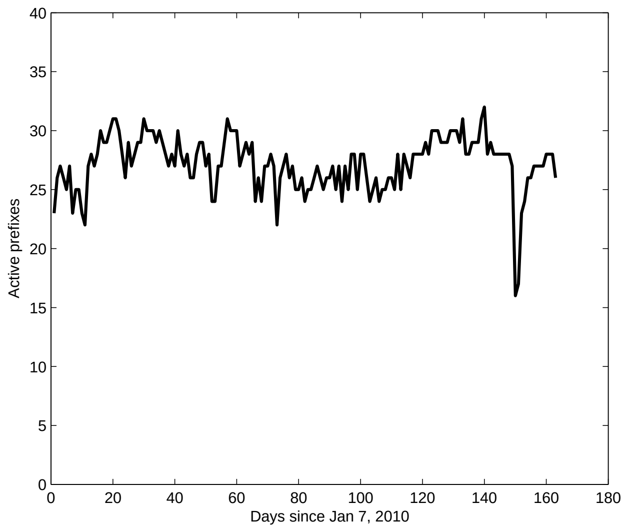


**Figure 9.3:** Evolution of the number of prefixes

#### 9.3.1.2 Reverse DNS

Figure 9.4 shows the weekly evolution for the number of rDNS entries.

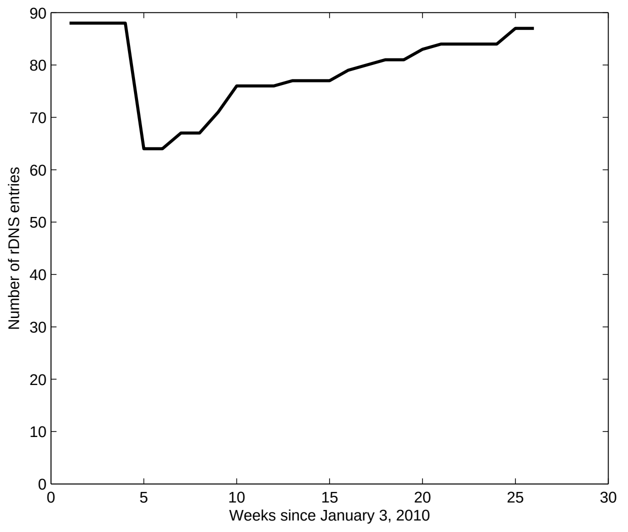


**Figure 9.4:** Evolution of the number of rDNS entries

### 9.3.2 Google Maps Overlay

The Google Maps overlay showing the approximate placement of the LISP pilot network elements is embedded on the main page of the LISPmon project, `http://lispmon.net`.

Figure 9.5 shows the initial view of the map, as retrieved on June 13, 2010.

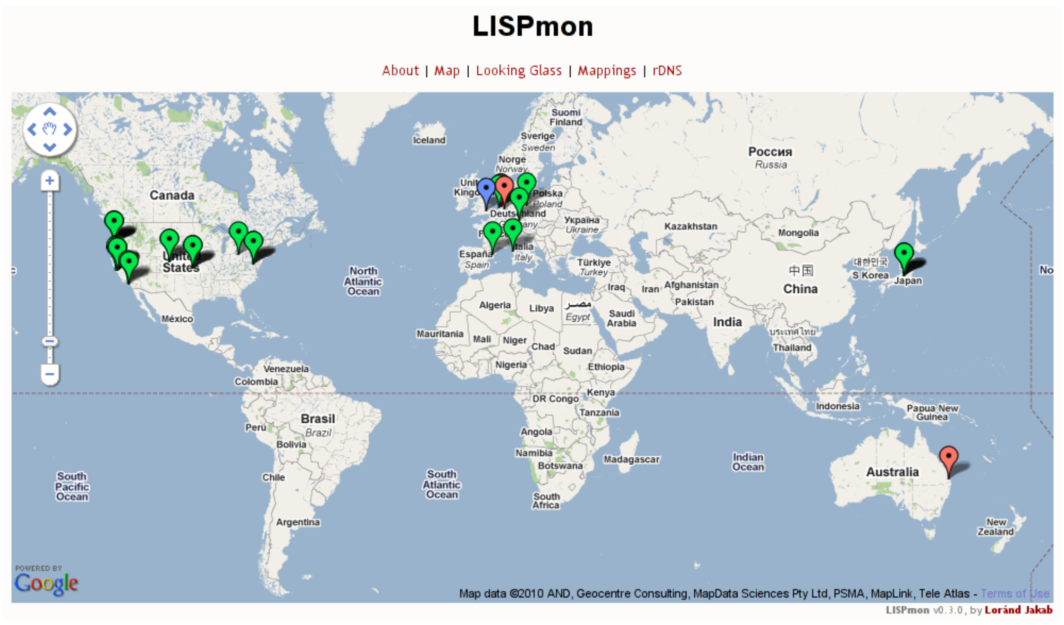
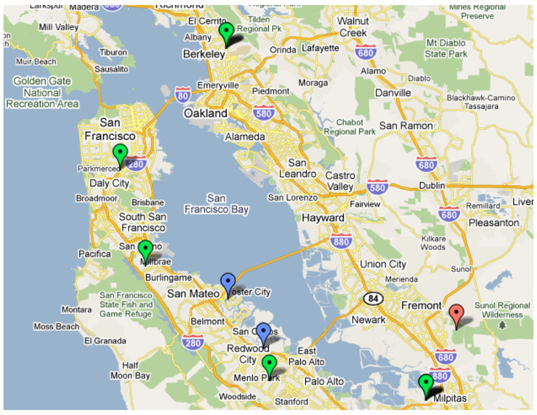

LISPmon
About | Map | Looking Glass | Mappings | rDNS
Canada
United States
México
North Atlantic Ocean
Iceland
Россия Russia
Kazakhstan
Mongolia
China
India
Japan
Australia
Brasil
Map data ©2010 AND, Geocentre Consulting, MapData Sciences Pty Ltd, PSMA, MapLink, Tele Atlas - Terms of Use
LISPmon v0.3.0, by Loránd Jakab

**Figure 9.5:** LISPmon router map

The markers visible on the map show either Map-Resolver/Map-Server elements (blue) or active (green) or inactive (red) tunnel routers. Hovering with the mouse over a marker shows a tooltip with the name of the router. Most routers are concentrated in the United States of America and Europe.

Figure 9.6(a) shows the San Francisco Bay Area (Silicon Valley and surroundings), which has the highest concentration of routers. This is of no surprise, because many technology companies participating in the testbed are based in Silicon Valley. Routers in Europe can be viewed on higher zoom level in Figure 9.6(b).

When clicking on a marker, a popup appears with data about the router. This data is organized into two tabs: the first one shows data which was determined locally on the LISPmon servers (Figure 9.7(a)), while the second shows data scraped off from the LPN status page (Figure 9.7(b)).


(a) San Francisco Bay Area

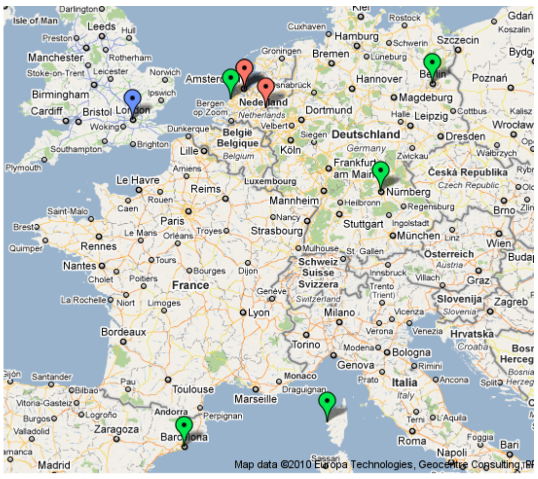
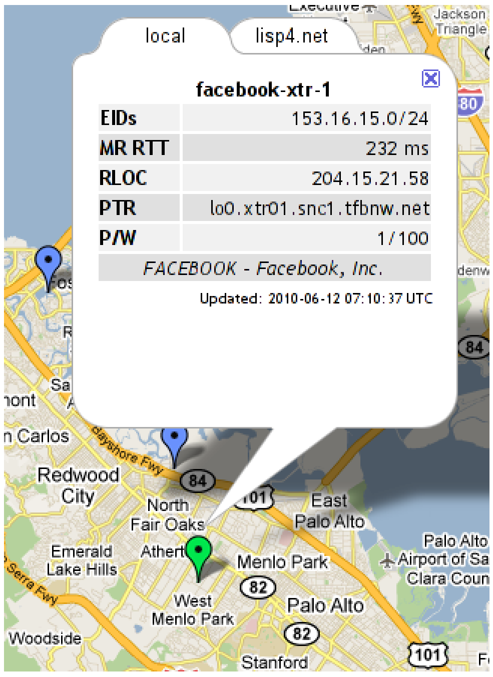

(b) Europe

**Figure 9.6:** Detailed map view



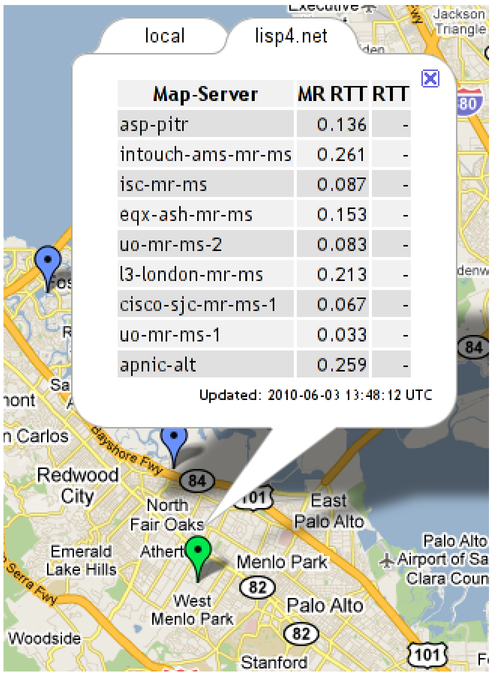
facebook-xtr-1

| | |
|---|---|
| EIDs | 153.16.15.0/24 |
| MR RTT | 232 ms |
| RLOC | 204.15.21.58 |
| PTR | lo0.xtr01.snc1.tfbnw.net |
| P/W | 1/100 |

*FACEBOOK - Facebook, Inc.*

Updated: 2010-06-12 07:10:37 UTC

(a) Local Data



| Map-Server | MR RTT | RTT |
|---|---|---|
| asp-pitr | 0.136 | - |
| intouch-ams-mr-ms | 0.261 | - |
| isc-mr-ms | 0.087 | - |
| eqx-ash-mr-ms | 0.153 | - |
| uo-mr-ms-2 | 0.083 | - |
| l3-london-mr-ms | 0.213 | - |
| cisco-sjc-mr-ms-1 | 0.067 | - |
| uo-mr-ms-1 | 0.033 | - |
| apnic-alt | 0.259 | - |

Updated: 2010-06-03 13:48:12 UTC

(b) LPN Data

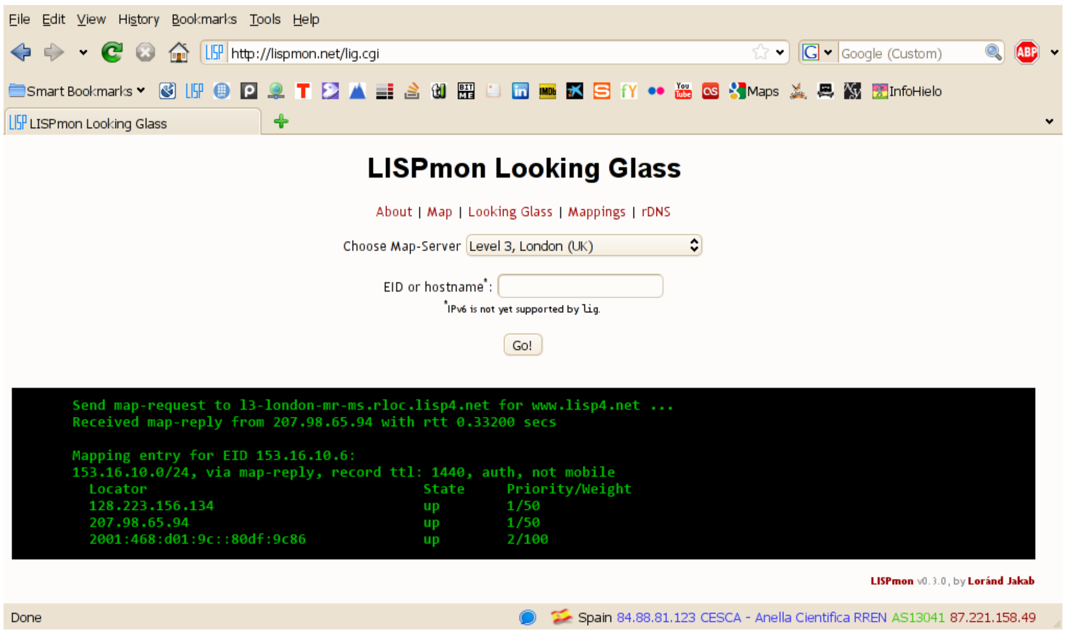
**Figure 9.7:** Example tooltip

### 9.3.3 Looking Glass

LISPmon offers a *looking glass* into the LISP mapping system. A looking glass server in networking context is a remotely accessible user interface giving a view into the network from the vantage point of the server.

Users of the LISPmon looking glass can query the LISP mapping system. This is useful for debugging purposes: e.g., when a network engineer can't get a response from the mapping system on his local network, using the looking glass it can determine if the problem lies in the mapping system or his network.

See Figure 9.8 for an example query output.


File Edit View History Bookmarks Tools Help
http://lispmon.net/lig.cgi
Google (Custom)
Smart Bookmarks
Maps
InfoHielo
LISPmon Looking Glass

LISPmon Looking Glass

About | Map | Looking Glass | Mappings | rDNS

Choose Map-Server: Level 3, London (UK)

EID or hostname*:

*IPv6 is not yet supported by lig.

Go!

```
Send map-request to l3-london-mr-ms.rloc.lisp4.net for www.lisp4.net ...
Received map-reply from 207.98.65.94 with rtt 0.33200 secs

Mapping entry for EID 153.16.10.6:
153.16.10.0/24, via map-reply, record ttl: 1440, auth, not mobile
  Locator                                    State     Priority/Weight
  128.223.156.134                            up        1/50
  207.98.65.94                               up        1/50
  2001:468:d01:9c::80df:9c86                 up        2/100
```

LISPmon v0.3.0, by Loránd Jakab

Done    Spain 84.88.81.123 CESCA - Anella Cientifica RREN AS13041 87.221.158.49

**Figure 9.8:** LISPmon Looking Glass Results

# 10

# LISP Protocol Dissector for Wireshark

Developing a network protocol software stack involves examining packets generated by the implementation, to check for conformance to specification. Protocol dissectors are a useful tool to do the examination, by decoding individual protocol fields, showing their name and value. The most popular open source network protocol analyzer is Wireshark, based on the *libpcap* packet capture library.

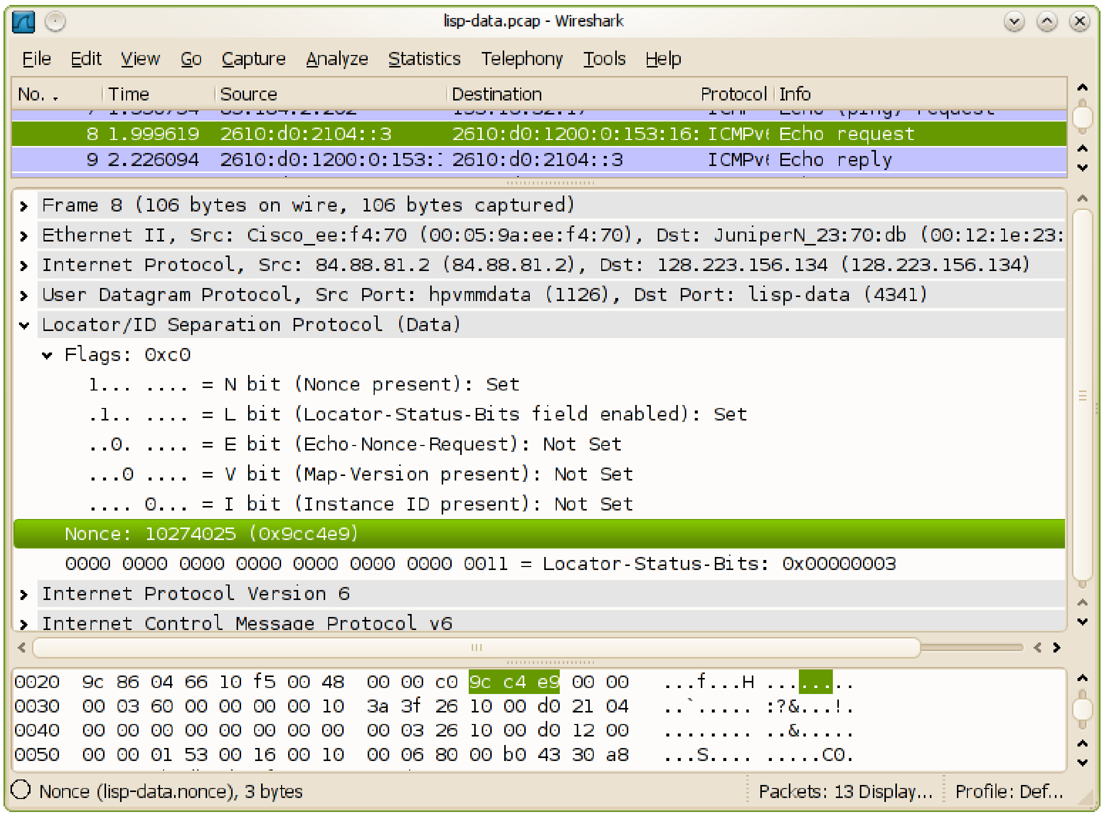
Wireshark provides a framework for protocol decoder plugins written in the C programming language. For each new protocol a plugin must be written using the Wireshark API so that packets conforming to that protocol are correctly detected and decoded. When this work started, no such plugin existed for LISP.

As part of this work, to help with LISP experimentation and development, two plugins were developed for the Wireshark network analyzer: one to decode the LISP data encapsulation header and another to decode LISP control packets. Protocol dissection using the first one is triggered for the payloads of UDP port 4341 datagrams, while the second codepath is activated by UDP port 4342. These ports are the LISP data and LISP control ports respectively.

Figure 10.1 shows an example screenshot of Wireshark decoding a LISP data encapsulation header. Once the header is decoded, control is returned to the IP protocol decoder, to continue dissection of the rest of the packet.

In Figure 10.2 a control packet structure (Map-Reply) is illustrated.

The new dissector also adds the appropriate display filters to Wireshark. The keyword *lisp* can be used to show only LISP control packets, or *lisp-data* to show LISP encapsulated data packets. Hierarchical labels allow direct access to filtering based on


lisp-data.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help

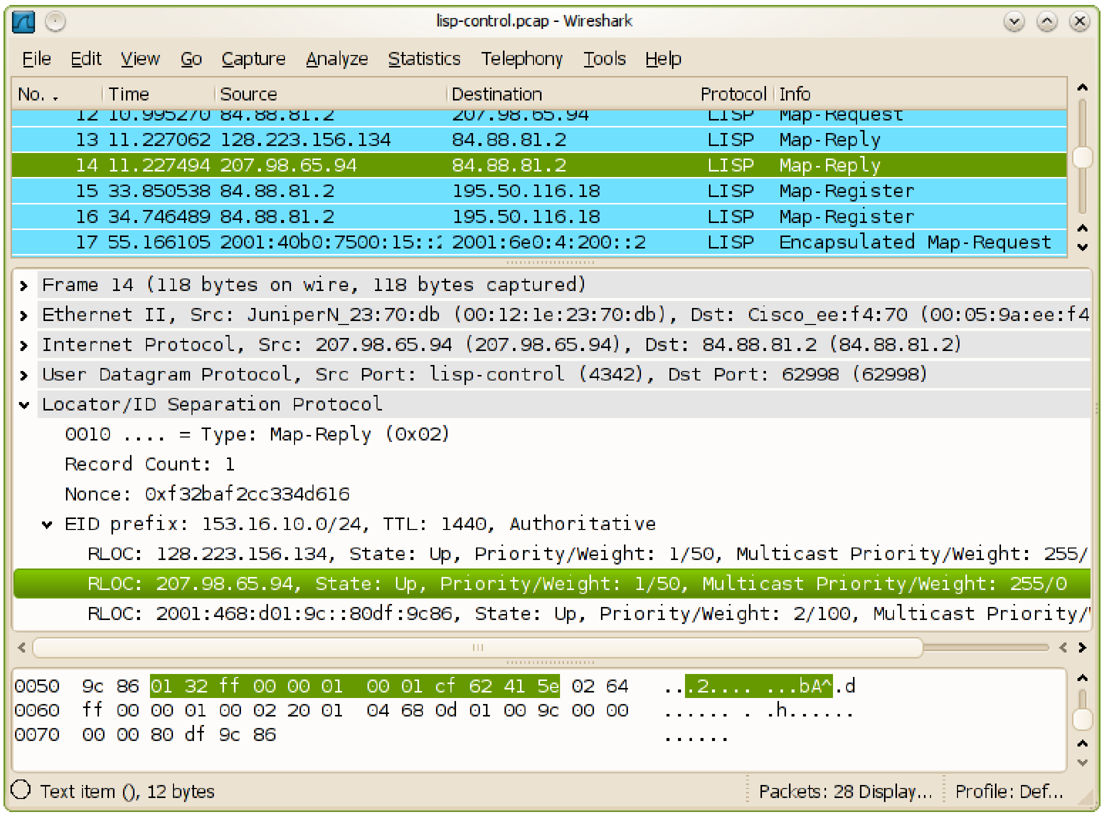
```
No. .  Time      Source                  Destination              Protocol Info
    8 1.999619  2610:d0:2104::3         2610:d0:1200:0:153:16:  ICMPv6  Echo request
    9 2.226094  2610:d0:1200:0:153::    2610:d0:2104::3         ICMPv6  Echo reply

> Frame 8 (106 bytes on wire, 106 bytes captured)
> Ethernet II, Src: Cisco_ee:f4:70 (00:05:9a:ee:f4:70), Dst: JuniperN_23:70:db (00:12:1e:23:
> Internet Protocol, Src: 84.88.81.2 (84.88.81.2), Dst: 128.223.156.134 (128.223.156.134)
> User Datagram Protocol, Src Port: hpvmmdata (1126), Dst Port: lisp-data (4341)
v Locator/ID Separation Protocol (Data)
  v Flags: 0xc0
      1... .... = N bit (Nonce present): Set
      .1.. .... = L bit (Locator-Status-Bits field enabled): Set
      ..0. .... = E bit (Echo-Nonce-Request): Not Set
      ...0 .... = V bit (Map-Version present): Not Set
      .... 0... = I bit (Instance ID present): Not Set
    Nonce: 10274025 (0x9cc4e9)
    0000 0000 0000 0000 0000 0000 0000 0011 = Locator-Status-Bits: 0x00000003
> Internet Protocol Version 6
> Internet Control Message Protocol v6

0020  9c 86 04 66 10 f5 00 48  00 00 c0 9c c4 e9 00 00   ...f...H ........
0030  00 03 60 00 00 00 00 10  3a 3f 26 10 00 d0 21 04   ..`..... :?&...!.
0040  00 00 00 00 00 00 00 00  00 03 26 10 00 d0 12 00   ........ ..&.....
0050  00 00 01 53 00 16 00 10  00 06 80 00 b0 43 30 a8   ...S.... .....C0.

Nonce (lisp-data.nonce), 3 bytes     Packets: 13 Display...   Profile: Def...
```

**Figure 10.1:** Example screen output of decoding a LISP data encapsulation header

specific field values. For example, the expression *"lisp.mapping.ttl == 1440"* would allow displaying control packets containing a mapping with a one day TTL.

Wireshark has a text based version called `tshark` as well, which is also supported by this LISP dissector. An example text output of a decoded data encapsulation header is as follows:

```
[...]
User Datagram Protocol, Src Port: 61571 (61571), Dst Port: lisp-data (4341)
    Source port: 61571 (61571)
    Destination port: lisp-data (4341)
    Length: 100
    Checksum: 0x0000 (none)
        [Good Checksum: False]
        [Bad Checksum: False]
Locator/ID Separation Protocol (Data)
    Flags: 0xc0
```


lisp-control.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help

```
No. .  Time        Source                 Destination      Protocol Info
   12 10.995270 84.88.81.2             207.98.65.94     LISP   Map-Request
   13 11.227062 128.223.156.134        84.88.81.2       LISP   Map-Reply
   14 11.227494 207.98.65.94           84.88.81.2       LISP   Map-Reply
   15 33.850538 84.88.81.2             195.50.116.18    LISP   Map-Register
   16 34.746489 84.88.81.2             195.50.116.18    LISP   Map-Register
   17 55.166105 2001:40b0:7500:15::  2001:6e0:4:200::2 LISP   Encapsulated Map-Request

> Frame 14 (118 bytes on wire, 118 bytes captured)
> Ethernet II, Src: JuniperN_23:70:db (00:12:1e:23:70:db), Dst: Cisco_ee:f4:70 (00:05:9a:ee:f4
> Internet Protocol, Src: 207.98.65.94 (207.98.65.94), Dst: 84.88.81.2 (84.88.81.2)
> User Datagram Protocol, Src Port: lisp-control (4342), Dst Port: 62998 (62998)
v Locator/ID Separation Protocol
    0010 .... = Type: Map-Reply (0x02)
    Record Count: 1
    Nonce: 0xf32baf2cc334d616
  v EID prefix: 153.16.10.0/24, TTL: 1440, Authoritative
      RLOC: 128.223.156.134, State: Up, Priority/Weight: 1/50, Multicast Priority/Weight: 255/
      RLOC: 207.98.65.94, State: Up, Priority/Weight: 1/50, Multicast Priority/Weight: 255/0
      RLOC: 2001:468:d01:9c::80df:9c86, State: Up, Priority/Weight: 2/100, Multicast Priority/

0050  9c 86 01 32 ff 00 00 01  00 01 cf 62 41 5e 02 64   ...2.... ...bA^.d
0060  ff 00 00 01 00 02 20 01  04 68 0d 01 00 9c 00 00   ...... . .h......
0070  00 00 80 df 9c 86                                  ......

Text item (), 12 bytes                    Packets: 28 Display...   Profile: Def...
```

**Figure 10.2:** Example screen output of decoding a LISP control packet

```
        1... .... = N bit (Nonce present): Set
        .1.. .... = L bit (Locator-Status-Bits field enabled): Set
        ..0. .... = E bit (Echo-Nonce-Request): Not set
        ...0 .... = V bit (Map-Version present): Not set
        .... 0... = I bit (Instance ID present): Not set
    Nonce: 14057215 (0xd67eff)
    0000 0000 0000 0000 0000 0000 0000 0011 = Locator-Status-Bits: 0x00000003
Internet Protocol, Src: 153.16.1.146 (153.16.1.146), Dst: 153.16.32.17
(153.16.32.17)
    Version: 4
    Header length: 20 bytes
[...]
```

The data packet decoder plugin was accepted for inclusion by the upstream Wireshark developers on January 21, 2011, and as of April 24, the control packet decoder was committed as well. The code is now available in the Wireshark project's SVN repository, and the next stable branch (1.6.x, currently in RC) will contain the LISP decoder code developed within this thesis. The decoder plugin is still available on the project web page [1], and new features continue to be developed. The ongoing changes in the LISP specification also requires maintenance, and new code will be pushed upstream, once it stabilizes.

Feedback from the LISP developer community, as well as the project web page access and download statistics, suggest wide use of these plugins.

[1] `http://www.cba.upc.edu/lisp-packet-dissector`

# 11

# Conclusions and Future Work

This thesis studied aspects of the scalability of the Internet control and data planes in the context of the locator/identifier split paradigm, particularly that of locator-to-identifier mapping systems and live video streaming.

We presented the problem of the semantic overloading of Internet Protocol addresses with both location and identity information, how that contributed to scalability issues of the inter-domain Internet routing system and the default-free zone routing tables, and how locator/identifier separation is widely believed to be the best approach to solve this problem. However, it requires a system to map identifiers to locators, and this new addition to the Internet control plane may itself come with its own scalability problems. With this basic motivation, the first part of the thesis explored and compared architectural properties and performance of three different distributed mapping databases designed for the Locator/Identifier Separation Protocol (LISP). LISP is one of the protocol specifications putting the locator/identifier split idea in practice, having the support of one the major router manufacturers, the open source community, and a global experimental network.

We built the CoreSim open source large-scale Internet topology simulator and implemented each mapping system on top of it. The simulator used the actual IPv4 prefixes present in the global routing tables to simulate mapping system lookups in an Ingress Tunnel Router (ITR). The lookups were based on real 24 h traffic traces captured at the border routers of two large European university campuses, which were replayed by the simulator.

Extensive simulations revealed important scalability issues with two of the mapping system proposals, the BGP-based LISP+ALT and the Chord-based LISP-DHT. The third one is the DNS-based LISP-TREE mapping system, first proposed by the LISP team led by professor Olivier Bonaventure from Université catholique de Louvain, and further developed within this thesis. LISP-TREE addresses several architectural shortcomings of the other mapping systems, and shows better performance in most cases. We are collaborating with the LISP authors to further improve the mapping system to the point where it can replace the currently used LISP+ALT as the mapping system of choice for LISP.

Migrating the current Internet to LISP requires upgrading a subset of all routers to support the new protocol. These kind of upgrades do not happen too often, and we believe that the likely migration to a LISP-enabled Internet should be taken advantage of to enable better network layer inter-domain multicast. This thesis proposes the CoreCast protocol, enabling efficient inter-domain live video streaming, modifying the same routers that need to be upgraded to support LISP.

The CoreCast architecture enables network planning for ISPs, because they can estimate the resources necessary for supporting a given number of streams. This further allows service level agreements between content providers and service providers, ensuring proper delivery of the content for the former and opening a new business opportunity to the latter.

Additionally, our analytical model, combined with measurement data, suggests that compared to popular P2P live streaming systems, CoreCast produces significantly less inter-domain traffic. The gains depend on the distribution of clients among autonomous systems, with as much as 300-fold decrease observed in our datasets.

Throughout the document, we pointed out some of the limitations of our work, which present opportunities for future research. The following paragraphs summarize these possible avenues for further exploration.

First, the CoreSim simulator is only working in offline mode, using saved traffic trace files. This limits the length of the simulated period, due to the excessive storage requirements for long traces. An online approach would be able to simulate mapping system performance data on much longer timescales, greatly increasing the relevance of the results, and creating an opportunity to observe trends over time. Such a system is already planned to be implemented on top of the CoMo platform [20], which was the fruit of another thesis from the research group.

Simulations in this work did not take into account time-to-live values of mappings due to the trace length limitation: the typical TTL value is one day, the same time interval as our traffic traces. Using an online system the simulator could support TTL, and provide relevant cache behavior statistics.

Another limitation of our simulation approach is related to transport layer protocol behavior: it cannot reproduce TCP retransmissions due to cache misses in the ITR, or UDP flow control implemented at the application layer. These issues are still unexplored and deserve detailed study.

Further, our simulations assumed migration of the current IPv4 Internet to LISP, and our results are based on that. IPv6 is quickly getting adopted since the depletion of the IANA IPv4 address pool, making an analysis of mapping systems for this next generation Internet protocol increasingly relevant. Such analysis should for example determine the netmasks most appropriate for delimiting the hierarchical levels of the LISP+ALT and LISP-TREE mapping systems for IPv6 prefix allocations.

The chapter on deployments scenarios (which is being worked on as an Internet Draft) presents how new LISP network elements are to be deployed, but no evaluation of the presented scenarios is offered. A detailed analysis, backed by real datasets (similar to [45]) would be valuable to the research community. LISP's new ingress traffic engineering and host mobility features are offering additional interesting research topics.

Finally, CoreCast is a live streaming delivery protocol, using the LISP mapping system to look up destination locators. Since the main purpose of the protocol is to deliver *content*, designing a content based mapping system would greatly enhance its functionality.

# References

[1] **Akamai Technologies**. [Online] http://www.akamai.com/. 5

[2] **comcast.net IPv6 Information Center**. Retrieved January 2011. [Online] http://www.comcast6.net/. 3

[3] **EdgeCast Networks**. [Online] http://www.edgecast.com/. 5

[4] **Hurricane Electric IPv4/IPv6 PoP Addresses**. [Online] http://he.net/HurricaneElectricNetworkMap.pdf. 3

[5] **LimeLight Networks**. [Online] http://www.limelightnetworks.com/. 5

[6] **MaxCDN**. [Online] http://www.maxcdn.com/. 5

[7] **MaxMind - GeoLite City — Free Geolocation Database**. [Online] http://www.maxmind.com/app/geolitecity. 29, 109

[8] **PPLive**. [Online] http://www.pplive.com/. 5, 56

[9] **Sopcast**. [Online] http://www.sopcast.com/. 5, 56

[10] **Team Cymru: IP to ASN Mapping**. [Online] http://www.team-cymru.org/Services/ip-to-asn.html. 108

[11] **TVAnts**. [Online] http://www.tvants.com/. 5, 56

[12] **The Verisign Domain Report**. *The Domain Name Industry Brief*, **7**(4):1–5, November 2010. 7

[13] **Wordwide Infrastructure Security Report**. Technical Report Volume VI, Arbor Networks, Inc., 2010. [Online] http://www.arbornetworks.com/report. 3

[14] Juan Jose Adan. **Tunneled Inter-domain Routing (TIDR)**. draft-adan-idr-tidr-01, November 2006. Work in progress. [Online] http://tools.ietf.org/html/draft-adan-idr-tidr-01. 4

[15] Sharad Agarwal and Jacob R. Lorch. **Matchmaking for online games and other latency-sensitive P2P systems**. In *SIGCOMM*, pages 315–326, August 2009. 30

[16] Shahzad Ali, Anket Mathur, and Hui Zhang. **Measurement of Commercial Peer-To-Peer Live Video Streaming**. In *Proceedings of Workshop in Recent Advances in Peer-to-Peer Streaming*, August 2006. 5

[17] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. **A Survey of Peer-to-Peer Content Distribution Technologies**. *ACM Computing Surveys*, **36**(4):335–371, December 2004. 5

[18] Randall Atkinson, Saleem Bhatti, and Stephen Hailes. **ILNP: Mobility, Multi-homing, Localised Addressing and Security Through Naming**. *Telecommunication Systems*, **42**(3–4):273–291, 2009. 4

[19] Hitesh Ballani, Paul Francis, Tuan Cao, and Jia Wang. **Making Routers Last Longer with ViAggre**. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*, pages 453–446. USENIX Association, April 2009. 3

[20] Pere Barlet-Ros, Gianluca Iannaccone, Josep Sanjuàs-Cuxart, Diego Amores-López, and Josep Solé-Pareta. **Load Shedding in Network Monitoring Applications**. In *USENIX Annual Technical Conference*, 2007. 39, 126

[21] BBC. **Multicast TV and Radio**. [Online] `http://www.bbc.co.uk/multicast/`. 56

[22] Jianjian Bian and Sunil P. Khatri. **IP Routing Table Compression Using ESPRESSO-MV**. In *Proceedings of the 11th IEEE International Conference on Networks (IEEE ICON '03)*, pages 167–172, September 2003. 3

[23] Scott Brim, Noel Chiappa, Dino Farinacci, Vince Fuller, Darrel Lewis, and David Meyer. **LISP-CONS: A Content distribution Overlay Network Service for LISP**. draft-meyer-lisp-cons-04, April 2008. Work in progress. 6, 15, 16, 17

[24] Albert Cabellos-Aparicio, Damien Saucez, Olivier Bonaventure, and Jordi Domingo-Pascual. **Validation of a LISP Simulator**. Technical Report UPC-DAC-RR-2009-46, UPC, 2009. [Online] `http://gsi.ac.upc.edu/apps/reports/2009/46/TR.pdf`. 29

[25] Isidro Castineyra, Noel Chiappa, and Martha Steenstrup. **The Nimrod Routing Architecture**. RFC 1992 (Informational), August 1996. 4

[26] Meeyoung Cha, Pablo Rodriguez, Sue Moon, and Jon Crawcroft. **On Next-Generation Telco Managed P2P TV Architectures**. In *Proceedings of the 7th International Workshop on Peer-to-Peer Systems (IPTPS '08)*, February 2008. 56, 63, 73

[27] Florin Coras. *CoreSim: A Simulator for Evaluating LISP Mapping Systems*. Master's thesis, Technical University of Cluj-Napoca, June 2009. 27

[28] Stephen E. Deering and David R. Cheriton. **Multicast Routing in Datagram Internetworks and Extended LANs**. *ACM Transactions on Computer Systems (ToCS)*, **8**(2):85–110, May 1990. 5

[29] Steve Deering. **Host extensions for IP multicasting**. RFC 1112 (Standard), August 1989. Updated by RFC 2236. 5

[30] Cristophe Diot, Brian Neil Levine, Bryan Lyles, Hassan Kassem, and Doung Balensiefen. **Deployment Issues for the IP Multicast Service and Architecture**. *IEEE Network*, **14**(1):78–88, January 2000. 5, 56, 65

[31] Ahmed Elmokashfi, Amund Kvalbein, and Constantine Dovrolis. **BGP Churn Evolution: a Perspective from the Core**. In *Proceedings of the 29th Conference on Computer Communications (IEEE INFOCOM '10)*, pages 1–9, March 2010. 3

[32] Dino Farinacci, Vince Fuller, David Meyer, and Darrel Lewis. **Locator/ID Separation Protocol (LISP)**. draft-ietf-lisp-10, March 2011. Work in progress. 4, 21, 22, 83, 84, 107

[33] Dino Farinacci and David Meyer. **LISP Internet Groper (LIG)**. draft-ietf-lisp-lig-00, April 2010. Work in progress. [Online] `http://tools.ietf.org/html/draft-ietf-lisp-lig-00`. 107

[34] Patrick Frejborg. **Hierarchical IPv4 Framework**. draft-frejborg-hipv4-12, February 2011. Work in progress. [Online] `http://tools.ietf.org/html/draft-frejborg-hipv4-12`. 4

[35] Vince Fuller and Dino Farinacci. **LISP Map Server**. draft-ietf-lisp-ms-07, March 2011. Work in progress. 23, 92, 94

[36] Vince Fuller, Dino Farinacci, David Meyer, and Darrel Lewis. **LISP Alternative Topology (LISP+ALT)**. draft-ietf-lisp-alt-06, March 2011. Work in progress. 6, 15, 16, 33, 94

[37] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross. **Insights into PPLive: a Measurement Study of a Large-Scale P2P IPTV System**. In *Proceedings of the IPTV Workshop, International WWW Conference*, 2006. 56

[38] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross. **A Measurement Study of a Large-Scale P2P IPTV System**. *IEEE Transactions on Multimedia*, **9**(8):1672–1687, December 2007. 5, 67

[39] Xiaojun Hei, Yong Liu, and Keith W. Ross. **Inferring Network-Wide Quality in P2P Live Streaming Systems**. *IEEE Journal on Selected Areas in Communications*, **25**(9):1640–1654, December 2007. 5

[40] Robert M. Hinden. **New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG**. RFC 1955 (Informational), June 1996. 4

[41] Paul Hoffman. **The Tao of IETF: A Novice's Guide to the Internet Engineering Task Force**. draft-hoffman-tao4677bis-08, May 2010. Work in progress. [Online] `http://tools.ietf.org/html/draft-hoffman-tao4677bis-08`. 101

[42] Geoff Huston. **Growth of the BGP Table - 1994 to Present**. [Online] `http://bgp.potaroo.net/`. 2, 7

[43] Geoff Huston. **Addressing 2010**. *Internet Society ISP Column*, January 2011. [Online] `http://isoc.org/wp/ispcolumn/?p=303`. 3

[44] Luigi Iannone and Olivier Bonaventure. **On the Cost of Caching Locator/ID Mappings**. In *Proceedings of the 3rd International Conference on emerging Networking EXperiments and Technologies (CoNEXT'07)*, pages 1–12. ACM, December 2007. 40

[45] Luigi Iannone and Tapio Levä. **Modeling the economics of Loc/ID Split for the Future Internet**. In Georgios Tselentis, Alex Galis, Anastasius Gavras, Srdjan Krco, Volkmar Lotz, Elena Simperl, Burkhard Stiller, and Theodore Zahariadis, editors, *Towards the Future Internet - Emerging Trends from European Research*, pages 11–20. IOS Press, 2010. 127

[46] Luigi Iannone, Darrel Lewis, David Meyer, and Vince Fuller. **LISP EID Block**. draft-meyer-lisp-eid-block-02, March 2011. Work in progress. 92, 98

[47] Luigi Iannone, Damien Saucez, and Olivier Bonaventure. **OpenLISP Implementation Report**. draft-iannone-openlisp-implementation-01, February 2008. Work in progress. [Online] `http://tools.ietf.org/html/draft-iannone-openlisp-implementation-01`. 6, 27, 101

[48] Dan Jen, Michael Meisel, He Yan, Dan Massey, Lan Wang, Beichuan Zhang, and Lixia Zhang. **Towards a New Internet Routing Architecture: Arguments for Separating Edges from Transit Core**. In *7th ACM Workshop on Hot Topics in Networks (HotNets '08)*, October 2008. 78

[49] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. **DNS Performance and the Effectiveness of Caching**. *IEEE/ACM Transactions on Networking (ToN)*, **10**(5):528–540, October 2002. 89

[50] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. **Delayed Internet Routing Convergence**. *ACM SIGCOMM Computer Communication Review*, **30**(4):175–187, October 2000. 4

[51] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, Farnam Jahanian, and Manish Karir. **ATLAS Internet Observatory 2009 Annual Report**. Technical report, Arbor Networks, the University of Michigan and Merit Network, 2009. 41

[52] Eliot Lear. **NERD: A Not-so-novel EID to RLOC Database**. draft-lear-lisp-nerd-08, March 2010. Work in progress. 6, 15, 16, 17

[53] Darrel Lewis, David Meyer, Dino Farinacci, and Vince Fuller. **Interworking LISP with IPv4 and IPv6**. draft-ietf-interworking-01, August 2010. Work in progress. 95, 99

[54] Darrel Lewis and Margaret Wasserman. **LISP Deployment Scenarios**, November 2009. [Online] `http://www.ietf.org/proceedings/76/slides/lisp-3/lisp-3.htm`. 9

[55] Bo Li, Susu Xie, Gabriel Y. Keung, Jiangchuan Liu, Ion Stoica, Hui Zhang, and Xinyan Zhang. **An Empirical Study of the Coolstreaming+ System**. *IEEE Journal on Selected Areas in Communications*, **25**(9):1627–1639, December 2007. 5

[56] Bo Li, Susu Xie, Yang Qu, Gabriel Y. Keung, Chuang Lin, Jiangchuan Liu, and Xinyan Zhang. **Inside the New Coolstreaming: Principles, Measurements and Performance Implications**. In *Proceedings of the 27th Conference on Computer Communications (IEEE INFOCOM '08)*, pages 1031–1039, April 2008. 5

[57] Tony Li. **Design Goals for Scalable Internet Routing**. RFC 6227 (Informational), May 2011. 4

[58] Tony Li. **Recommendation for a Routing Architecture**. RFC 6115 (Informational), February 2011. 4

[59] Huan Liu. **Reducing Routing Table Size Using Ternary-CAM**. In *Proceedings of Hot Interconnects 9*, pages 69–73, August 2001. 3

[60] Jiangchuan Liu, Bo Li, and Ja-Qin Zhang. **Adaptive Video Multicast over the Internet**. *IEEE Multimedia*, **10**(1):22–33, January 2003. 5, 56

[61] Harsha V. Madhyastha, Thomas Anderson, Arvind Krishnamurthy, Neil Spring, and Arun Venkataramani. **A Structural Approach to Latency Prediction**. In *Proceedings of the Internet Measurement Conference (IMC'06)*, October 2006. 29

[62] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. **iPlane: An Information Plane for Distributed Services**. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI'06)*. USENIX Association, November 2006. 28, 29

[63] Daniel Massey, Lan Wang, Beichuan Zhang, and Lixia Zhang. **A Scalable Routing System Design for Future Internet**. In *SIGCOMM 2007 Workshop "IPv6 and the Future of the Internet"*, August 2007. 4

[64] Laurent Mathy and Luigi Iannone. **LISP-DHT: Towards a DHT to Map Identifiers onto Locators**. In *Re-Architecting the Internet (ReArch'08)*, December 2008. 6, 15, 16, 17

[65] Michael Menth, Matthias Hartmann, and Dominik Klein. **Global Locator, Local Locator, and Identifier Split (GLI-Split)**. Technical Report 470, University of Würzburg, April 2010. [Online] `http://www3.informatik.uni-wuerzburg.de/TR/tr470.pdf`. 4

[66] David Meyer. **The Locator Identifier Separation Protocol (LISP)**. *The Internet Protocol Journal*, **11**(1):23–36, 2008.

[67] David Meyer, Lixia Zhang, and Kevin Fall. **Report from the IAB Workshop on Routing and Addressing**. RFC 4984 (Informational), September 2007. 1, 2, 4, 29, 78

[68] Shinsuke Miwa, Mio Suzuki, Hiroaki Hazeyama, Satoshi Uda, Toshiyuki Miyachi, Youki Kadobayashi, and Yoichi Shinoda. **Experiences in Emulating 10K AS Topology With Massive VM Multiplexing**. In *Proceedings of the ACM SIGCOMM 2009 VISA Workshop*, August 2009. 6

[69] Paul V. Mockapetris and Kevin J. Dunlap. **Development of the Domain Name System**. *SIGCOMM Computer Communication Review*, **25**(1):112–122, 1995. 22

[70] Mike O'Dell. **GSE - An Alternate Addressing Architecture for IPv6**. draft-ietf-ipngwg-gseaddr-00, February 1997. Work in progress. [Online] `http://tools.ietf.org/html/draft-ietf-ipngwg-gseaddr-00`. 4

[71] Bruno Quoitin, Luigi Iannone, Cédric de Launois, and Olivier Bonaventure. **Evaluating the Benefits of the Locator/Identifier Separation**. In *Proceedings of the 2nd International Workshop on Mobility in the Evolving Internet Architecture (MobiArch '07)*, August 2007. 4

[72] Jennifer Rexford and Constantine Dovrolis. **Future Internet Architecture: Clean-Slate Versus Evolutionary Research**. *Communications of the ACM*, **53**(9):36–40, September 2010. 2

[73] Jan Seedorf, Martin Stiemerling, Marco Mellia, Renato Lo Cigno, and Csaba Kiraly. **Design Considerations for a Peer-to-Peer Streaming Protocol**. draft-seedorf-ppsp-design-considerations-01, October 2010. Work in progress. [Online] `http://tools.ietf.org/html/draft-seedorf-ppsp-design-considerations-01`. 56

[74] Alexandro Sentinelli, Gustavo Marfia, Mario Gerla, Leonard Kleinrock, and Saurabh Tewari. **Will IPTV Ride the Peer-to-Peer Stream?** *IEEE Communications Magazine*, **45**(6):86–92, June 2007. 5

[75] Thomas Silverston. *Peer-to-Peer Video Live Streaming: Measurement Experiments and Traffic Analysis*. PhD thesis, Université Pierre et Marie Curie, September 2009. 68

[76] Thomas Silverston and Olivier Fourmaux. **Measuring P2P IPTV Systems**. In *Proceedings of NOSSDAV '07*, June 2007. 5

[77] Thomas Silverston, Olivier Fourmaux, Alessio Botta, Alberto Dainotti, Antonio Pescapé, Giorgio Ventre, and Kavé Salamatian. **Traffic Analysis of Peer-to-Peer IPTV Communities**. *Elsevier Computer Networks*, **53**(4):470–484, March 2009. 5

[78] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. **Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications**. *IEEE/ACM Transactions on Networking*, **11**(1):17–32, February 2003. 17

[79] Fred Templin. **The Internet Routing Overlay Network (IRON)**. RFC 6179 (Experimental), March 2011. 4

[80] Javier Ubillos, Mingwei Xu, Zhongxing Ming, and Christian Vogt. **Name-Based Sockets Architecture**. draft-ubillos-name-based-sockets-03, September 2010. Work in progress. [Online] `http://tools.ietf.org/html/draft-ubillos-name-based-sockets-03`. 4

[81] VB.com. **Domains Counter - Domain Timeline since 1985 by VB.com**, 2009. [Online] `http://www.vb.com/domain-timeline.htm`. 49

[82] Christian Vogt. **Six/One Router: A Scalable and Backwards Compatible Solution for Provider-Independent Addressing**. In *Proceedings of the 3rd International Workshop on Mobility in the Evolving Internet Architecture (MobiArch '08)*, August 2008. 78

[83] Yangyang Wang, Jun Bi, and Jianping Wu. **NOL: Name Overlay Service for Improving Internet Routing Scalability**. In *Proceedings of the 2nd International Conference on Advances in Future Internet (AFIN'10)*, pages 17–21, July 2010. 4

[84] Duane Wessels and Geoffrey Sisson. **Root Zone Augmentation and Impact Analysis**. Technical report, DNS-OARC, 2009. [Online] `https://www.dns-oarc.net/files/rzaia/rzaia_report.pdf`. 36

[85] Robin Whittle. **Ivip (Internet Vastly Improved Plumbing) Architecture**. draft-whittle-ivip-arch-04, March 2010. Work in progress. [Online] `http://tools.ietf.org/html/draft-whittle-ivip-arch-04`. 4

[86] Damon Wischik. **Short messages**. In *Royal Society workshop on networks: modelling and control*, September 2007. 40

[87] Chuan Wu, Baochun Li, and Shuqiao Zhao. **Diagnosing Network-wide P2P Live Streaming Inefficiencies**. In *Proceedings of IEEE INFOCOM '09 Mini-conference*, April 2009. 5, 6, 57

[88] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Liu, and Avi Silberschatz. **P4P: Provider Portal for Applications**. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, August 2008. 65

[89] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. **P4P: Provider Portal for Applications**. In *SIGCOMM*, August 2008. 41

[90] Xiaohu Xu. **Routing Architecture for the Next Generation Internet (RANGI)**. draft-xu-rangi-04, August 2010. Work in progress. [Online] `http://tools.ietf.org/html/draft-xu-rangi-04`. 4

[91] Lixia Zhang. **An Overview of Multihoming and Open Issues in GSE**. *IETF Journal*, **2**(2), September 2006. [Online] `http://isoc.org/wp/ietfjournal/?p=98`. 4

[92] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. **CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming**. In *Proceedings of the 24th Conference on Computer Communications (IEEE INFOCOM '05)*, March 2005. 5, 56

# Appendix A

# Complete List of Publications

## A.1 Related Publications

- T. Silverston, L. Jakab, A. Cabellos-Aparicio, O. Fourmaux, K. Salamatian, K. Cho, “Large-scale measurement experiments of P2P-TV systems. Insights on fairness and locality”, to appear in *Elsevier Signal Processing: Image Communication*, 2011

- L. Jakab, A. Cabellos-Aparicio, F. Coras, J. Domingo-Pascual, D. Lewis, “LISP Network Element Deployment Considerations”, draft-ietf-lisp-deployment (Work in progress)

- L. Jakab, A. Cabellos-Aparicio, T. Silverston, M. Solé, F. Coras, and J. Domingo-Pascual, “CoreCast: How Core/Edge Separation Can Help Improving Inter-Domain Live Streaming”, *Elsevier Computer Networks*, vol. 54, no. 18, pp. 3388–3401, December 2010

- L. Jakab, A. Cabellos-Aparicio, F. Coras, D. Saucez, and O. Bonaventure, “LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System”, *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1332–1343, October 2010

- L. Jakab, A. Cabellos-Aparicio, and J. Domingo-Pascual, “CoreCast: Efficient Live Streaming in the Core-Edge Separated Internet”, in: ACM SIGCOMM 2009 poster session, Barcelona, Spain, August 2009

## A.2 Talks

- “LISP Network Element Deployment Considerations”, 79th IETF Meeting, LISP Working Group, Beijing, China, November 2010
- “Evaluation of LISP+ALT performance”, 75th IETF Meeting, LISP Working Group, Stockholm, Sweden, July 2009

## A.3 Other Publications

- R. Cuevas, Á. Cuevas, A. Cabellos-Aparicio, L. Jakab, and C. Guerrero, “A Collaborative P2P Scheme for NAT Traversal Server Discovery Based on Topological Information”, *Elsevier Computer Networks*, vol. 54, no. 12, pp. 2071–2085, August 2010
- L. Jakab, J. Domingo-Pascual, “A Selective Survey of DDoS Related Research”, Technical Report, UPC-DAC-RR-CBA-2007-3
- R. Serral-Gracià, L. Jakab, and J. Domingo-Pascual, “Measurement Based Call Quality Reporting”, in: Proceedings of the 2nd IEEE LCN Workshop on Network Measurements, Dublin, Ireland, October 2007
- R. Serral-Gracià, L. Jakab, J. Domingo-Pascual, “Out of Order Packets Analysis on a Real Network Environment,” in: Proceedings of the 2nd Conference on Next Generation Internet Design and Engineering (EuroNGI), Valencia, Spain, April 2006
- L. Jakab, R. Serral-Gracià, J. Domingo-Pascual, “A Study of Packet Losses in the EuQoS Network,” in: Proceedings of the 4th International Workshop on Internet Performance, Simulation, Monitoring and Measurement (IPS-MoMe), Salzburg, Austria, February 2006, pp. 138–146
- A. Cabellos-Aparicio, J. Núñez-Martínez, H. Julian-Bertomeu, L. Jakab, R. Serral-Gracià, J. Domingo-Pascual, “Evaluation of the Fast Handover Implementation for Mobile IPv6 in a Real Testbed,” in: T. Magedanz et al. (Ed.), Operations and Management in IP-Based Networks, Lecture Notes in Computer Science, Volume 3751, pp. 181–190, Springer, 2005
- A. Cabellos-Aparicio, R. Serral-Gracià, L. Jakab, J. Domingo-Pascual, “Measurement Based Analysis of the Handover in a WLAN MIPv6 Scenario,” in: C.

Dovrolis (Ed.), Passive and Active Network Measurement, Lecture Notes in Computer Science, vol. 3431, pp. 203–214, Springer, 2005

- L. Jakab, A. Cabellos-Aparicio, R. Serral-Gracià, J. Domingo-Pascual, "Software Tool for Time Duration Measurements of Handovers in IPv6 Wireless Networks", Technical Report, UPC-DAC-2004-25