# Master thesis : LISP Privacy: An addressless approach to client-server communication

**Auteur :** Thielens, Elric
**Promoteur(s) :** Donnet, Benoît; 19442
**Faculté :** Faculté des Sciences appliquées
**Diplôme :** Master : ingénieur civil en informatique, à finalité spécialisée en "computer systems security"
**Année académique :** 2022-2023
**URI/URL :** http://hdl.handle.net/2268.2/16761


University of Liège

School of Engineering and Computer Science

# LISP Privacy: An addressless approach to client-server communication

Master's thesis completed in order to obtain the degree of Master of Science in

Computer Engineering

by

Elric Thielens

**Author**
Elric Thielens

**Supervisor**
Benoit Donnet

Academic Year 2022-2023

# Abstract

With the increasing amount of data being exchanged over the Internet, privacy has become a critical concern for many actors relying on its services. This includes both individuals and organizations, who may be concerned about the confidentiality of their data as well as their own identity confidentiality. This work builds upon a model designed for server anonymization in IP networks, and presents a series of models built on top of the LISP protocol that aim at mitigating threats to privacy, such as network scanning and other forms of surveillance, which can have serious consequences for both clients and servers. It does so by reducing devices' identifiability as much as possible.

The Locator/Identifier Separation Protocol (LISP) has been developed to address the issue of the increasing size of routing tables in routers of the default-free zone (DFZ). The IAB highlighted the overloading of IP address semantics as the main cause. LISP separates the identifier and locator properties of an IP address into two separate address spaces in order to address this issue.

This work explores the potential for using LISP to provide anonymization to end devices in a communication. By implementing and comparing various models in the ns-3 simulation environment, we demonstrated the feasibility of using LISP for this purpose. The models provided in this work proved to be much better in terms of delay compared to the original solution and they can be combined to provide complete privacy to both clients and servers while also being easier to deploy and maintain.

This works explores solution built on top of the LISP protocol which have the advantage to be easy to deploy on top of an existing architecture. As LISP is still in development, it would be interesting to study the advantages of built-in solutions.

# Acknowledgements

I would like to address my gratitude to Mr. Donnet and Mr. Ianonne, that have given me the opportunity to explore this very interesting subject. Their support and monitoring allowed me to stay on tracks and meet my objectives.

I would also like to thank "La main" and my friends for hearing my complaints and providing a somewhat constant support. A special thanks to Antoine and Gilles that read (part of) this work without having any background in Computer Sciences and still provided insightful advices for its organization.

# Contents

# Abbreviations

| | |
|---|---|
| **LISP** | **L**ocator/**I**dentifier **S**eparation **P**rotocol |
| **EID** | **E**ndpoint **ID** |
| **RLOC** | **R**outing **L**ocator |
| **ETR** | **E**gress **T**unnel **R**outer |
| **ITR** | **I**ngress **T**unnel **R**outer |
| **DDT** | **D**elegated **D**atabase **T**ree |
| **ALT** | **A**lternative **L**ogical **T**opology |
| **UDP** | **U**ser **D**atagram **P**rotocol |
| **TCP** | **T**ransmission **C**ontrol **P**rotocol |
| **IP** | **I**nternet **P**rotocol |
| **DNS** | **D**omain **N**ame **S**ystem |
| **RTT** | **R**ound **T**ime **T**rip |
| **DOS** | **D**enial **O**f **S**ervice |
| **NAT** | **N**etwork **A**ddress **T**ranslation |
| **GDPR** | **G**eneral **D**ata **P**rotection **R**egulation |
| **SSL** | **S**ecure **S**ocket **L**ayer |
| **TLS** | **T**ransport **L**ayer **S**ecurity |
| **MR** | **M**ap **R**esolver |
| **MS** | **M**ap **S**erver |

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## Context

Nowadays, most communication is digital and digital privacy has become critically important for both clients and servers. As a result, more and more research is being conducted to provide privacy at the different layers of the Open Systems Interconnection (OSI) model. In this paper, we will study how to provide privacy to communications using the LISP network layer protocol.

LISP, the Locator/Identifier Separation Protocol [1], was developed to address the issue of the increasing size of routing tables of routers in the default-free zone (DFZ). This scalability issue of the Internet routing and addressing system has been discussed in the Internet Architecture Board (IAB) workshop of 2006 [2], which identified the overloading of IP addresses semantics as the main cause of these issues. This results from conflictual interests between the core and the edge networks. The core of the Internet benefits from using an IP address as a *Locator*, allocated with respect to the topology in which they will be used, so that they can be aggregated into a single entry in the routing tables. This keeps the routers' table size as small as possible for efficient routing and reduce router costs. However, at the edge of the Internet, customer ISPs and large enterprises make use of different mechanisms such as multihoming, traffic engineering, and provider-independent addresses to reduce costs and improve the performance and maintainability of their infrastructure. These mechanisms rely on using IP addresses as *identifiers*, allocating them without considering the global topology, which leads to prefix de-aggregation and an increase in the number of entries in the routing tables.

A single address cannot efficiently fulfill this dual role and to address this issue, LISP separates the identifier and locator properties of an IP address into two separate address space. Endpoint Identifier (EID) addresses are used at the edge of the Internet and are locally routable addresses used to identify hosts in a network. Routing Locator (RLOC) addresses are globally routable addresses used in the core to identify network attachment points. To connect these two independent spaces, LISP introduces two components: LISP-enabled routers at the border of LISP sites and a Mapping System [3]. This mapping system provides the mappings between the two address spaces, similar to the Domain Name System (DNS) service. The

LISP protocol then uses IP-in-IP encapsulation to route packets between any border router.

This leads to a high compatibility with the current IP architecture as only these two components need to be added to support LISP, while the rest of the Internet remain unmodified. Due to the static nature of RLOCs and their high aggregability, this solution addresses the scalability issue in the Internet core and reduces the number of updates required to reach and maintain stable routes. EIDs, on the other hand, can be allocated according to an organization's policies without impacting the rest of the Internet. Multihoming, Traffic Engineering and renumbering are also well-supported by LISP.

# Problem

Solutions such as LISP-SEC [4] exist to make LISP more secure, but none have focused on the issue of digital privacy. In this work, digital privacy is divided into two properties: data confidentiality and identity confidentiality. While many solutions exist to implement data confidentiality at the network layer, very few addresses the challenge of anonymization.

Anonymization can provide significant benefits to both clients and servers. For clients, it brings anonymity to data transmission and minimizes the risk of identification or correlation between a client and a service. For servers, it decreases exposure to attacks and increases the reliability and availability of a service. One of the main threat to servers is network scanning [5, 6], which is often used to identify potential targets and their vulnerabilities. Even with the large address space of IPv6 [7], servers remain vulnerable as more and more studies are conducted to create new ways to scan IPv6 addresses [8, 9, 10, 11].

The goal of this work is to provide a first step towards the decrease of identifiability, at the network layer, of devices involved in a communication using LISP.

# Methodology and results

This thesis will demonstrate, starting from a methodology applied to a classic IP network [12], how we can establish both identity and location privacy to reduce both the identifiability of the actors involved in a communication and the risk of scanning for servers. By using LISP-specific architectural elements, a series of solutions will be presented and implemented using the network simulator environment ns-3 [13] and the existing LISP implementation updated by E. Marechal [14]. The solutions will also be compared based on the different metrics gathered from the implementation.

By comparing our models, we will show that these new privacy techniques using LISP are better than the original model implemented on an IP architecture. They are easier to deploy and maintain and result in less delays for clients when proper caching mechanisms are in place. They can also be used to provide privacy to clients.

# Roadmap

This work will be divided as follows. Chapter 1 will provide an overview of network privacy, defining a terminology of digital privacy and its threats, as well as a taxonomy of the solutions that can be developed. It will give an overview of privacy for both IPv4 and IPv6 and will end by presenting the model that inspired this work: Addressless privacy by Hao et al. [12]. Chapter 2 will describe the LISP protocol, its motivations, and its specific mechanisms. Chapter 3 will present the models developed in this work, following the taxonomy defined in Chapter 1. Chapters 5 and 6 will introduce the network simulator ns-3 and describe how it was used to gather data from these models. They will end by presenting a comparison of these models that will be made based on different metrics.

# Chapter 2

# Network Privacy

## 2.1 Privacy

Privacy is a broad term that has a lot of different definitions and interpretations, depending on the context in which it is being used. In this work, we will consider the legal definition of privacy, which refers to *the ability of an individual or organization to control who has access to their personal information and under what circumstances that information can be used.* In the context of Computer Sciences, data privacy encompasses a wide range of practices, tools, and regulations (such as the well-known GDPR [15] legislation) that have been put in place to ensure the proper collection, storage, and usage of digital data.

Digital privacy has become increasingly important nowadays, as more and more information is being shared online. Gathering data is now of crucial importance for a lot of different actors, for purposes ranging from targeted advertising, research, to others far more malevolent. As such, data privacy is a concern at every layer of the OSI model, with solutions at each of these layers that have been developed to protect the addresses and data involved from being listened to or tampered with. It's often through the use of overlapping mechanisms at each layer that digital privacy can be achieved to some extent.

This chapter will first establish a taxonomy of network address privacy, then it will give an overview of privacy in the context of an IP network with a focus on one particular model.

### 2.1.1 Goals and Terminology

Many threats involved are discussed in RFC 7258 [5] and RFC 6973 [6], with the most notable threats to digital privacy being: surveillance of an individual's network traffic, disclosure and mis-attribution of information. In this work, we will assume that the privacy of the data involved in the communication is assured, which is often done by using encryption during the communication and proper storage on the server side.

This work will focus on network-layer addresses and how to provide privacy guarantees to them. Why do we need to protect IP addresses? Let's consider a simple

| Term | Definition |
|---|---|
| *Anonymity* | The state of being anonymous. |
| *Anonymous* | A state of an individual in which an observer or attacker cannot identify the individual within a set of other individuals (the anonymity set). |
| *Anonymity Set* | A set of individuals that have the same attributes, making them indistinguishable from each other from the perspective of a particular attacker or observer. |
| *Identifiability* | The extent to which an individual is identifiable. |
| *Identifiable* | A property in which an individual's identity is capable of being known to an observer or attacker. |
| *Identification* | The linking of information to a particular individual to infer an individual's identity or to allow the inference of an individual's identity in some context. |
| *Correlation* | The combination of various pieces of information related to an individual or that obtain that characteristic when combined. |

Table 2.1: Terminology of Privacy components

example involving a single client contacting a server. In this scenario, the most pressing privacy threats for a client are *Identification* and *Correlation*, for example, linking an individual (or their device) to the use of a specific service. For a server, the threat would be to have its IP address collected, by either *eavesdropping* or *Network Scanning*, which is usually the first step in most attacks. Network scanning is usually used to gather IP addresses of potential targets, that can then be used when needed.

The terminology that will be used in this work is given in Table 2.1 and comes from RFC 6973 [6].

Our goal, using this terminology, is to create an Anonymity set by hiding the IP addresses involved in a client-server communication, making the attribution of the connection and the identification of the devices involved far more complicated. For a client, this has several benefits, such as avoiding its traffic to be linked to its device. But it is also very beneficial for a server, hiding the identity of a server will make the work of potential malicious actors less straightforward, as they would only be able to target a service and no longer target a specific server providing the service.

Figure 2.1 shows in orange the part of the Digital Privacy spectrum that is tackled in this work.

### 2.1.2 Threat to Digital Privacy: Network Scanning

Network scanning is the process of identifying and mapping the devices and services present in a network. It is typically performed by network administrators, security professionals, or malicious actors to identify vulnerabilities and/or troubleshoot network issues.

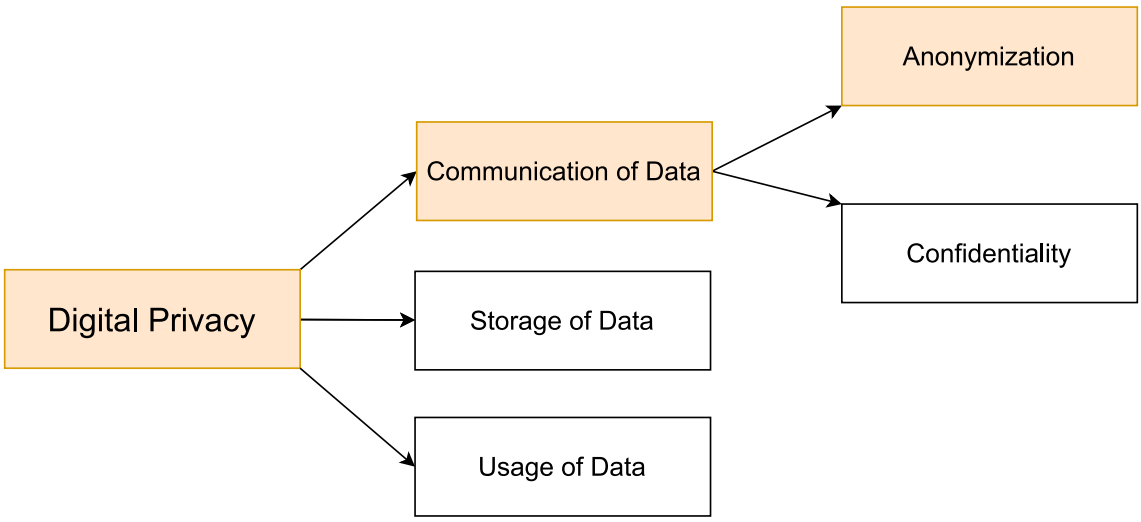


Figure 2.1: Taxonomy of Digital privacy

There are many automated tools have been developed to perform network scanning, including well-known such as Nmap [16] and Zmap [17]. Although they both perform network scanning, they have been specialized in different direction. Nmap is able to perform deep and thorough scanning of small to medium size networks, it is able to detect active hosts, open ports and even perform OS fingerprinting. It also has a scripting framework that allows it to perform vulnerabilities testing on listening ports. Zmap, on the other hand, has specialized for large but shallow network scanning. It is capable of scanning the entire IPv4 address space in less than one hour (1300 times faster than Nmap).

With the emergence of such efficient tools, it is critical for servers administrators to find and use mechanisms to reduce the identifiability of their servers to reduce their exposure to malicious actors.

### 2.1.3 Taxonomy of solutions

Privacy is still an important research topic, and new solutions have been summarized [18] in two categories by the Privacy Enhancements and Assessments Research Group (PEARG) at the Internet Engineering Task Force (IETF):

- *Over-the-top* approaches: Use existing architecture to implement new mechanisms, makes the deployment easier since fewer changes are to be made to the actual topology.
- *Built-in* approaches: Make strong changes in existing protocols or develop new ones. Very hard deployment, requires heavy changes.

In the context of a communication between a client and a server providing a service, we will divide the notion of "address privacy" into 3 properties, as depicted in Figure 2.2. Since an IP address is both the locator and the identifier of a device in an IP network, we can define, for a server:

- *Identity* privacy: hiding the identity of the server that provides the service.
- *Location* privacy: hiding the position of the server on the Internet, or, at least,

where it is located in the service provider topology.

- *Complete* privacy combining the two, ensuring both identity and location privacy.

Of course the same logic can be applied for a client.

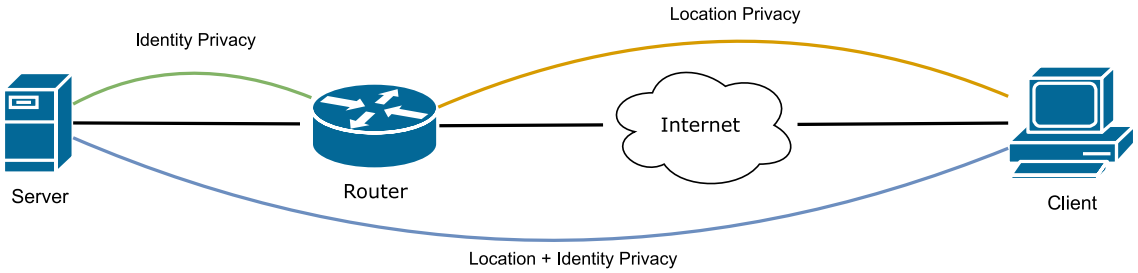


Figure 2.2: Taxonomy of privacy components

## 2.2 Privacy in IP networks

The position of the network layer about privacy is ambiguous, as actors generally prefer to use higher-level protocols, such as Transport Layer Security (TLS) [19], to ensure data confidentiality and usually have to rely on complex mechanisms to guarantee anonymization of network traffic.

There already are some relatively widely used protocols and architectures that try to mitigate threats to privacy at the network layer:

- Tunneling and VPNS using protocols like IPsec [20], but, even in the best case, the router addresses involved in the tunnel are still in clear. Moreover, IPsec is usually used between networks under the same administrative authority.
- Indirect routing using alternate topologies like Tor networks, that introduces heavy delays to the communication.

In the next section, a quick overview of network scanning and privacy in IPv4 and IPv6 will be presented.

### 2.2.1 IPv4

The IPv4 protocol uses a 32 bits address space, which makes it very vulnerable to address scanning by tools such as Zmap, as mentioned earlier.

Nevertheless, the shortage of IPv4 has an interesting side effect in terms of privacy: Network Address Translator (NAT) [21]. NAT allows devices without a public IP address to browse the Internet by sharing a single, public IP address. This helps to mask the true IP addresses of devices within a network and thus can help to improve the privacy of IPv4 communications by making it more difficult for attackers to identify and track individual devices of clients.

For servers, it is more complicated to protect against network scanning as NATs are not well suited in this scenario. However, good firewalls rules can help mitigate these scans or their efficiency. It is interesting to note that for situation where

multiple servers provide the same services, a load balancer is able to provide the same privacy properties as a NAT provides for clients by distributing incoming requests to a service to different servers, thus reducing the identifiability of these servers.

### 2.2.2 IPv6

The IPv6 protocol [7] has been developed to solve the address shortage of IPv4. As such, the main difference is that IPv6 uses an address space of 128-bit, which is far bigger than its predecessor. These 128-bit addresses are composed of 2 parts: a 64-bit network prefix and a 64-bit identifier interface (IID). Two protocols exists for address configuration of hosts: Stateless Address Auto-Configuration [22] (SLAAC) and Dynamic Host Configuration Protocol version 6 (DHCPv6) [23]. DHCPv6 is usually preferred when control is needed over address allocation like in large enterprise networks for example.

IPv6 has been designed with security in mind, learning from the mistakes made earlier to implement *built-in* security mechanisms such as IPsec (with its AH and ESP protocols) for confidentiality. In terms of addresses privacy, this huge address space makes brute-force network scanning far more complicated as it would require years to scan the entire IPv6 address space.

To bypass this time restriction, new IPv6 scanning techniques beyond brute-forcing have been developed, notably:

- Collecting IPv6 addresses from different sources, such as DNS [8] records or BGP advertised prefixes [9].
- Prediction of active/allocated IPv6 addresses using machine learning algorithms [10, 11].

These methods can be used to generate lists of IPv6 addresses that can then be targeted for deeper scanning. However, there is a concern among those involved in the development of IPv6 about the use of NAT [24]. NATing breaks the end-to-end principle, which is why it is not recommended for use with IPv6. As a result, depending on the direction that the practical implementation of IPv6 will take in the future, clients' privacy may be at risk in IPv6.

To address those new privacy concerns, both SLAAC and DHCPv6 have mechanisms in place to try to mitigate these threats. RFC 8981 [25] talks about SLAAC "Temporary addresses extensions" but these temporary addresses last at least several hours, with a network prefix that remains constant. Nevertheless, these mechanisms provide some form of mitigation for a very low cost. A very thorough list of background work on the improvement of address privacy, is given in the work done by Hao et al. [12].

In the next section, an overview of a new model providing "addressless" communication between clients and servers will be presented.

## 2.3 Addressless IP model

The core of this thesis is articulated around the model defined in the scientific paper *"Addressless: A new internet server model to prevent network scanning"*[12]. To compare this model with the results that we will obtain with our models in LISP, it is crucial to define and analyze the original solution.

The goal of this paper is to provide a new model that takes advantage of the vast IPv6 address space, the DHCPv6 prefix delegation mechanism [23] to allocate a prefix of addresses to a device, and a new intermediate device called *Entrance Module* to provide complete privacy to the server.

### 2.3.1 Model description

For this model to work, we have to set up several components:

- The *Entrance Module* is a basic server that has a public IP address that should be configured in the DNS system. Its only role is to generate an IP address and redirect incoming clients.
- The *main service module* is the server that provide the service. It is configured with a prefix of IP addresses thanks to DHCP-PD and uses all the addresses under the prefix to communicate with clients.

The key function is that the server will only accept incoming connections from clients that have gone through the entrance module first. This entrance module will redirect clients by giving them a specific IP address that they must use to contact the server.

The model involves multiple steps, which are depicted in Figure 2.3

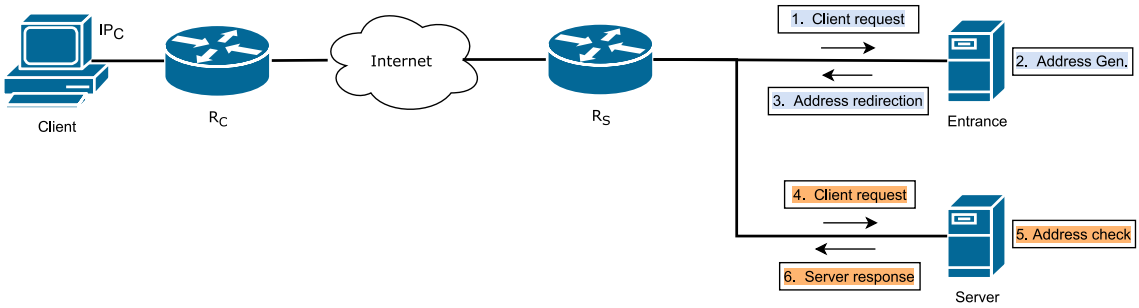


Figure 2.3: Addresslesss IP model topology

1. The client will start by sending a request to the service IP address obtained through the DNS system.
2. After receiving the request, the entrance module uses
   (a) The prefix of the main service module
   (b) The source address of the packet

   to generate an IPv6 address through the encryption function.

3. It will return this generated address to the client.
4. Finally, the client initiates a connection with the main service module using the received address.
5. Before establishing the connection, the main service module verifies the destination address used. If it checks out, the sever carries on with the connection, otherwise, the packet is dropped.
6. Communication takes place as usual with no modifications.

A simplified view of the model is given in Figure 2.4. In this figure, we can define the additional *connection delay* induced by this architecture as the time elapsed between the first request sent by the *Client* and the reception of the response sent by the *Server* to this *Client*.

This work only considers IPv6, as its address space size and the number of unused addresses allow allocating small prefixes to devices.

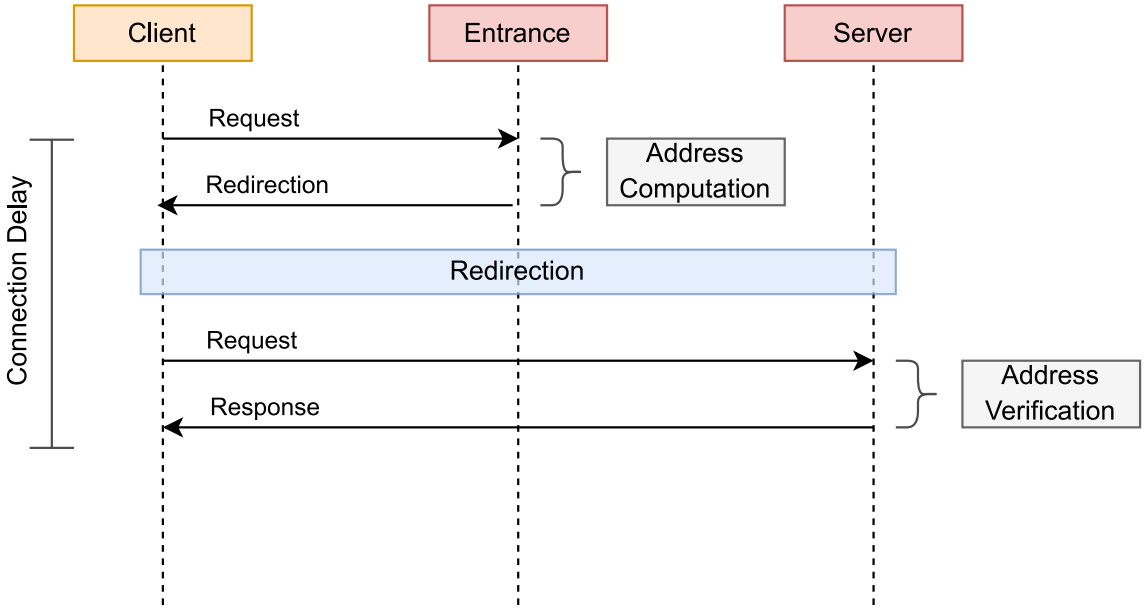


Figure 2.4: Addresslesss IP model packet flow

### 2.3.2 Generation and verification of IP addresses

In this work, a lot of mentions are made to the *generation* and *verification* of the IP addresses used to reach the server. We will follow the nomenclature used in the work of Hao et al. and define a simple version of the generation process:

$$DA_{1:N} = P_S \tag{2.1}$$

$$DA_{N+1:128} = E(H(IP_C), T_S) \tag{2.2}$$

where we define:

1. $DA$: The destination address that will be used by the client to reach the server.
2. $P_S$: The prefix of IP addresses that has been allocated to the server.

3. $E()$: The encryption process used to generate a valid suffix.

4. $IP_C$: The IP address of the client.

5. $N$: The number of bits allocated to the address prefix.

6. $T_S$: The system time when generating the address.

It is important to note that the study of the encryption process itself is beyond the scope of this work, but is studied in multiple other research papers [26, 27, 28, 29, 30]. For these simulations, $E()$ will be used as a simple hash function but the architecture is compatible with any encryption/verification functions that follows these definitions.

The verification process can be defined as

$$(H(IP_C), T_S) = E^{-1}(DA_{N+1:128}) \tag{2.3}$$

The two conditions to accept an incoming connection are:

$$H(IP_S) \stackrel{?}{=} H(IP_C) \tag{2.4}$$

$$T_M - T_S \stackrel{?}{<} T_{tresh} \tag{2.5}$$

where we define:

1. $IP_S$: The address that has been used to contact the server.

2. $T_M$: The system time when verifying the address.

3. $T_{tresh}$: The period of validity of an address.

4. $E^{-1}()$: The decryption process.

When this verification failed, the server simply ends the communication.

The complete description of the generation and verification process can be found in Annex A.1, the use of stateless salt using timestamps are added to fence off replay attacks.

### 2.3.3 Analysis

There are considerable benefits to this architecture, as detailed in great length by its authors. The key advantage is that the server is only accessible through the entrance module, making it far more difficult to scan.

- Attempting to collect active addresses of the server through the DNS records would only give the entrance module's address.
- Using machine learning or pattern recognition algorithms would not yield any valid addresses, as the generation algorithm used to generated valid addresses do not show any pattern, according to the authors.
- Eavesdropping. Using spoofed packets or replay attacks only work during the timeframe of validity of an IP address. However, it would not work if any kind of application or transport layer encryption is used.

- Brute-force, because of the IPv6 address space size, has a negligible chance of yielding any results. And even if the valid address of a given timeframe is found, it would only be valid for a small period of time.

This model also provides some light mitigation of well-known denial of service (DoS) attacks such TCP Syn-Flood and UDP Flooding. This model can mitigate these attacks when erroneous pairs of addresses are used since resource allocation on the server side only occurs after the destination address check. Still, this model is vulnerable within the address validity period when valid pairs of addresses are found. It is also vulnerable to DoS attacks that do not directly send packets to the server, such as bandwidth exhaustion

Finally, another benefit is that the solution allows for load balancing and/or user authentication to be performed at the entrance module, if needed. No modifications are required for the client.

This solution does come with some trade-offs. One drawback is that it introduces an additional round-trip time (RTT) and possibly a connection handshake, for TCP connections, before the actual connection can be established. Additionally, the redirection occurs at the application layer, which means that modifications would be required for any application that wants to use this model. The server also needs to be modified to perform the address checking, as performing this check at the application layer, after the connection is first established, would leave the server vulnerable to DoS attacks. Another implementation cost is the need for an additional device, the entrance module, to be configured and accessible. While this solution can be vulnerable to fast replay attacks and session hijacks within the period during which a source address is associated to a destination address, both of these vulnerabilities can be mitigated by using encryption at the transport or application layer.

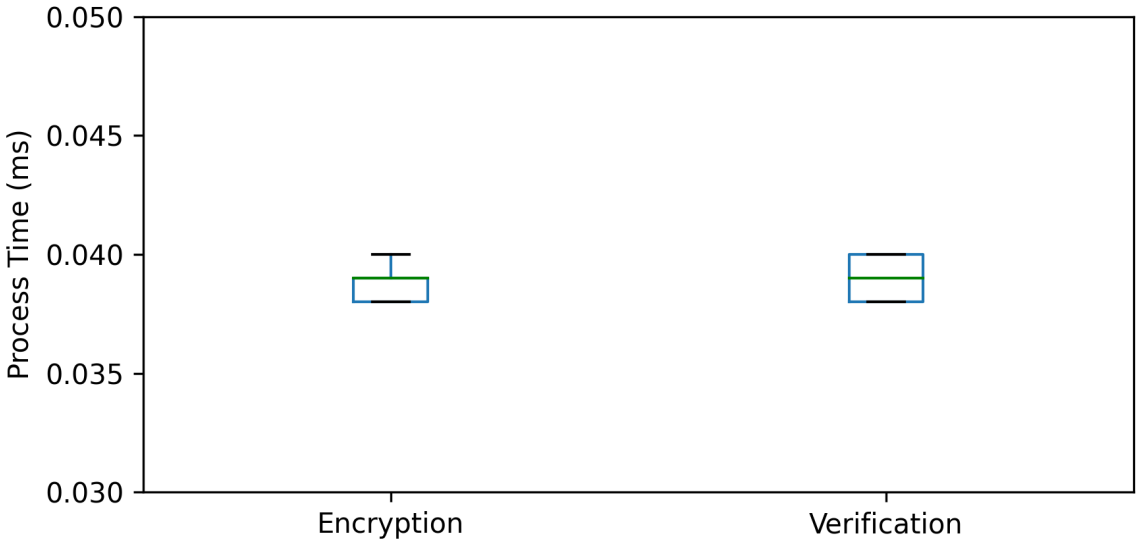


Figure 2.5: Encryption and verification delay
[12]

In their work Hao et al. measured the delay introduced by both encryption and redirection of a client using a real-life implementation of their work. They used multiple Raspberry Pi, running on RaspberryOS based on Debian, in the same local network and configured them to take on the job of the entrance module, the server

and the client as shown in Fig. A.2. They used DHCP-PD to allocate prefixes to the server.

The experiment was done using multiple combination of OSes and web browsers, a raspberry client went through both the entrance module and server, and the delays were measured using Wireshark.

The experiment is repeated 10 times, and the results of their measurements are divided between the delay due to encryption and decryption, shown in Figure 2.5 and the delay due to the change of connection performed by the client, shown in Figure 2.6. These results show that, except for the additional RTT due to the connection change, the delay induced at both the server and client side is negligible. We will use these empiric measures for our timing models in the ns-3 implementation.

An experiment was also conducted to measure the efficiency of the model against scanning attempts. In this scenario, a Raspberry Pi is used as an attacker, that will different method of scanning to try to find an active address of the server.

- A 100-hour brute force that yielded no results.
- Generating addresses using machine learning algorithm such as 6gen also yielded no results.

They concluded that their model was efficient as preventing the main server to be scanned by existing IPv6 scanning approaches.

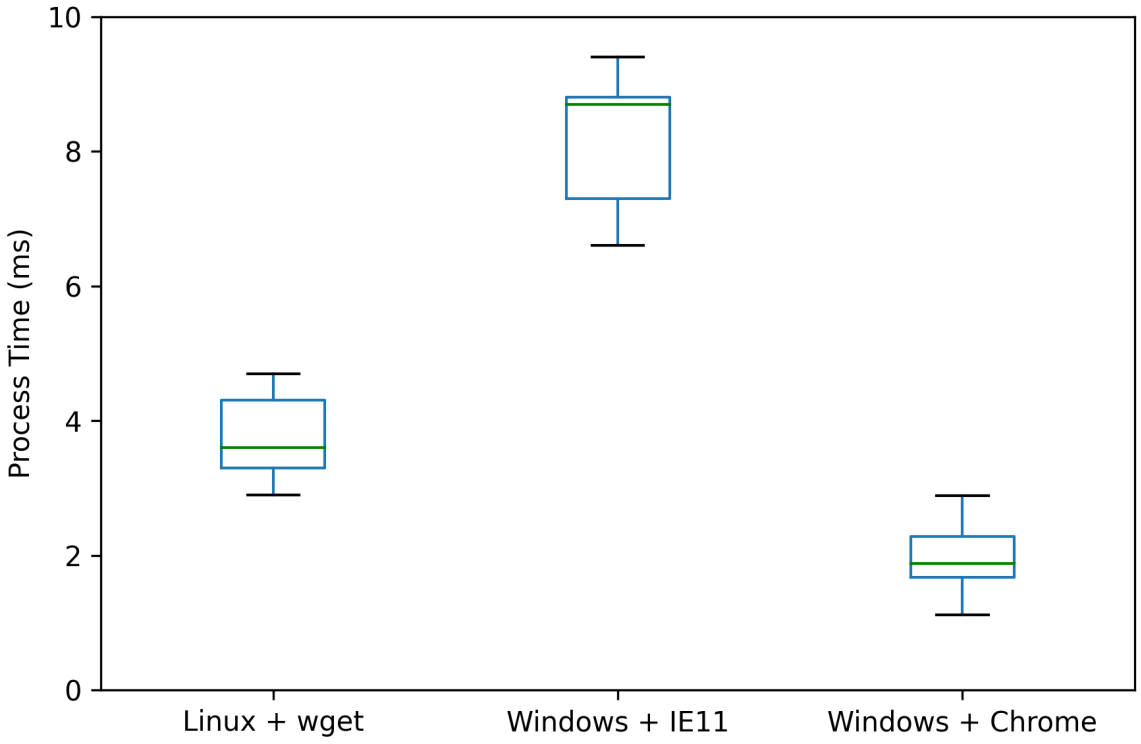


Figure 2.6: Delay due to the change of connection, for different Operating Systems and web browsers

[12]

# Chapter 3

# Locator/Identifier Separation Protocol

The Locator/Identifier separation protocol [1] is a network layer protocol that has been designed to address some shortcomings of the current use of IP addresses in the Internet's addressing and routing system. This chapter will cover the background and objectives of LISP, as well as its functional principles and mechanisms.

## 3.1 Motivation

In 2006, the IAB expressed during its annual workshop on routing and addressing several concerns about the current Internet network architecture. The main conclusion of the workshop was that the current growth of both the Routing Information Base (RIB) and Forwarding Information Base (FIB) of routers would lead to significant scalability issues for these routers in the near future. The following subsections will provide an overview of the issues and their causes, as presented in the workshop summary [2].

### 3.1.1 Scalability of the routing systems

During the meeting, the main concern discussed was the growing size of the DFZ (Default-free zone) routing table. The DFZ is composed of every router and Autonomous Systems (AS) on the Internet that do not use a default route to route packets, the routers in the DFZ have a complete Internet routing table which allows routing packets to any reachable destination. It is also known as the global routing table. The evolution of the size of this table is given in Figure 3.1 for IPv4 and Figure 3.2 for IPv6.

As the RIB size grows in the DFZ, there are several consequences that can impact the performance of the Internet. One of these is the cost of recomputing the FIB. When the RIB grows, it takes more time and resources to recompute the FIB, which can slow down the routing process.

This increased processing time directly impacts routing protocols that uses the RIB and FIB such as the Border Gateway Protocol [32] (BGP). BGP is used to

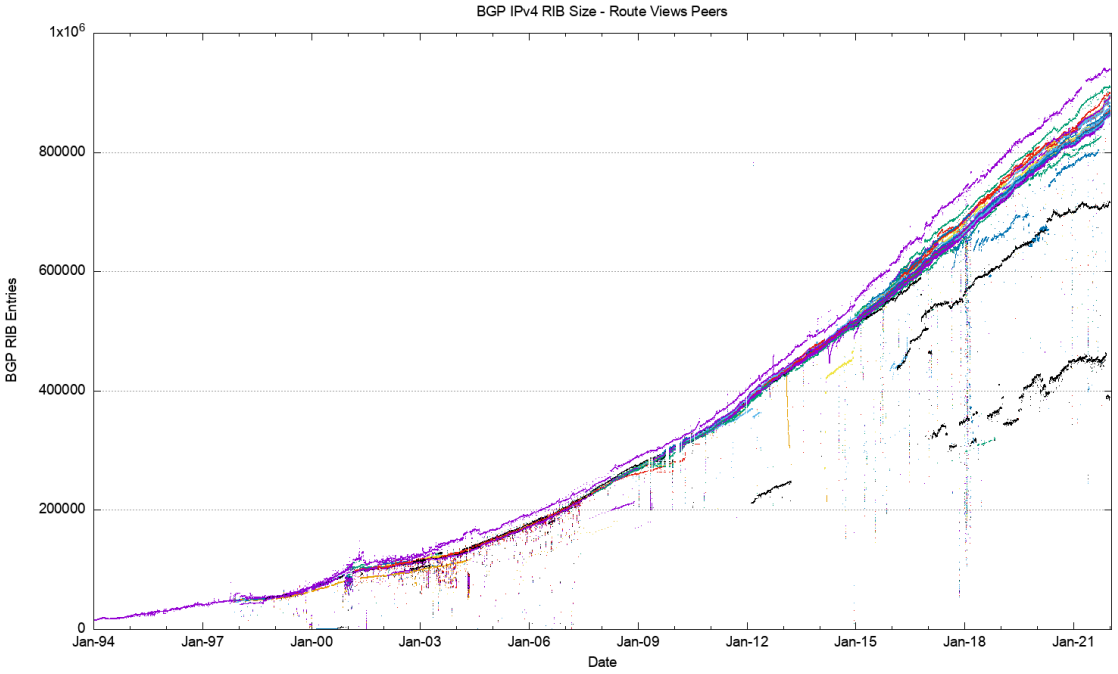


Figure 3.1: IPv4 routing table size evolution over the years [31]

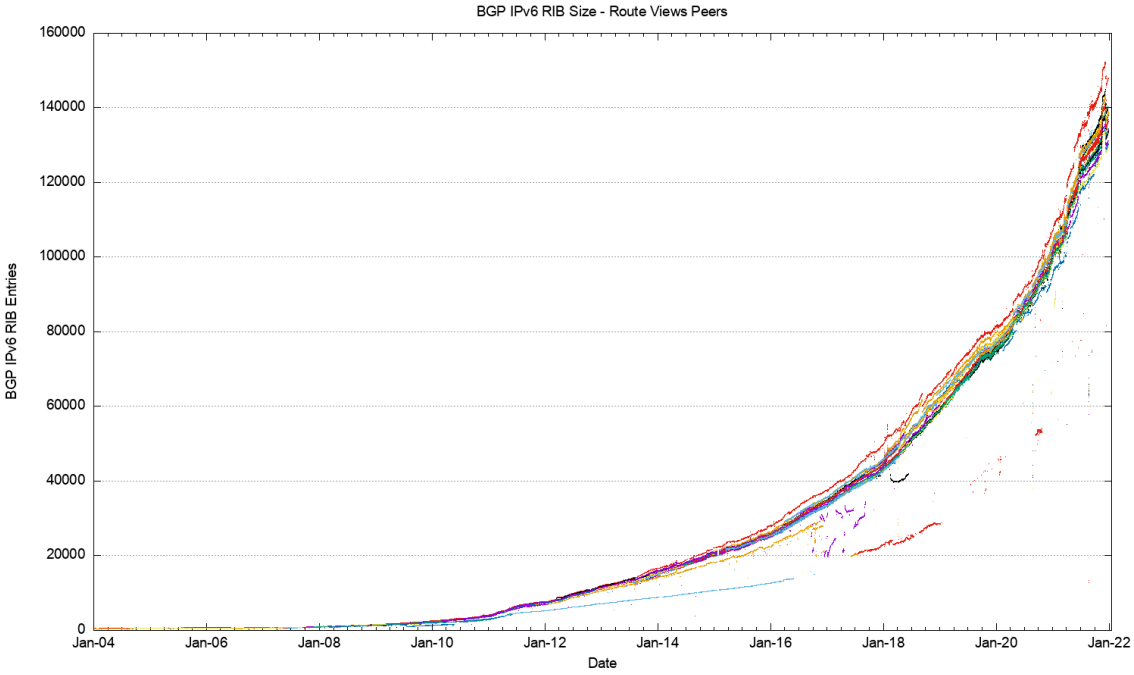


Figure 3.2: IPv6 routing table size evolution over the years [31]

exchange routing information between networks, its messages often lead to the processing of the RIB to compute the FIB. Because of the increase in RIB size, it will take longer for BGP to compute routes which in turn lead to delays in routing traffic. However, this is not the main issue for BGP. Large RIB size, as we will explain in the next sections, is mainly caused by the de-aggregation of routes. This increase in the number of routes can also lead to an increased number of BGP UPDATE messages being injected into the DFZ, which can contribute to UPDATE churn. BGP churn refers to the number of prefixes that are changed, added, or withdrawn over time. More churn a negative impact on routing converge and increase the delays needed to establish stable routes.

Finally, the growing RIB size has implications for the hardware needed to route traffic in the core of the Internet. As the RIB grows, more processing power and efficient storage are required to process and maintain the longer prefixes and to update the FIB. This increases the cost and power consumption of the hardware, as well as the heat dissipation.

The main technique to reduce the size of the RIB and FIB is prefix aggregation. Instead of having one entry per IP address in a network, the number of entries is reduced by representing a group of devices or networks with a single route for the common prefix that these addresses share. Representing a network by the prefix that has been allocated to it seems to be the rational approach, but problem is, for multiple reasons, a tendency towards prefix de-aggregation has been observed, which has lead to this increase of RIB size.

All the factors that will be explained in the following subsections can contribute to prefix de-aggregation and/or the injection of unaggregatable prefixes into the DFZ RIB, resulting in an uncontrolled RIB growth.

### Allocation of PI addresses

A RIR (Regional Internet Registry) is an organism that is responsible, among other things, for allocating addresses in a certain region of the world. It typically allocates two types of addresses: Provider-Independent (PI) and Provider-Aggregatable (PA) addresses.

PA addresses are IP addresses allocated by blocks to Internet service providers (ISP) by the RIR, and are further assigned by ISPs to customers. They are typically used by smaller organizations or networks that do not need to maintain a persistent IP address and are willing to use the address space of their ISP. They are said to be *aggregatable*, as all the IP addresses of an ISP block will usually be summarized in a single aggregated entry in the DFZ RIB.

PI addresses, on the other hand, are addresses that are not assigned by an ISP and are not specific to a particular provider. These addresses are not part of an ISP's allocated block and as such, are not aggregatable with it. This leads to additional entries in the RIB in order for these addresses to be reachable. These addresses are typically used by larger companies that needs stability for their addressing policies.

If an organization with a block of PA addresses wants to change ISPs, it must obtain a new block of IP addresses from the new provider. The owner of PI addresses,

however, can use it with any ISP and can change providers without changing its IP addresses. This is a huge benefit for an organization, as these addresses are often used statically, in firewalls, system's configuration or even for routing rules. It could become costly for an organization to change everything each time it has to change ISP, using PI addresses brings this needed flexibility.

### Multihoming

Multihoming refers to the practice of connecting an AS to multiple ISPs. This can be done for various reasons, such as increasing available bandwidth, increasing reliability, and providing connection redundancy.

When a site uses multihoming with provider-independent (PI) address space, its prefixes will be present in the routing tables of each of its ISPs and won't be aggregatable, as explained in the previous subsection. Even if the site uses provider-aggregatable (PA) address space, it leads to prefix de-aggregation. Each prefix allocated by one ISP can only be aggregated by that ISP and not by the others. In the case where the site has a "primary" ISP that it normally uses for routing unless there is a failure, the additional routing table entries for the other ISPs should only appear in the event of a failure to the primary ISP. However, the primary ISP will often have to de-aggregate the block allocated to a multi-homed site to avoid having the site's traffic directed to a secondary ISP if that ISP chooses to start advertising the prefix. As these prefixes won't be aggregatable by this ISP, traffic will start flowing through it because of the longest-prefix matching rule.

In both scenario, multihoming leads to prefix de-aggregation and thus growing RIB size.

### Traffic engineering

Traffic engineering refers to the practice of designing, implementing, and managing routing policies in a way that allows for the control and manipulation of routes taken by traffic in order to optimize traffic management.

It is often used for several reasons:

- Load balancing: achieve better utilization of network resources by spreading the traffic load.
- Policy routing: enforce some sorts of arbitrary rules or preferences by avoiding or enforcing path usage.
- Reduce cost: balance traffic to either respect peering agreements, or to use the cheapest paths.

Traffic Engineering is usually achieved by adjusting some parameters of BGP advertisement. Problem is, if the prefix to which Traffic Engineering must be applied is part of a bigger block of PA addresses, it must be de-aggregated in order to be advertised separately with the proper parameters. As a result, TE is one of the causes of the RIB growth in the DFZ.

#### IPv6 adoption

It is believed that the current growth of DFZ RIB size is limited by the exhaustion of the IPv4 address space. While IPv4 is still the main version currently used on the Internet, the transition to IPv6 and its vast address space is slowly taking place. As a result of this shift and the increasing number of digital devices being produced each year that require IP addresses, it is believed that an explosion in RIB growth could be due to the adoption of IPv6.

### 3.1.2 Overloading of IP address semantics

All these causes have a common root problem: the overloading of IP address semantics. The origin of this overloading lies in the two different conflicting business models of the organizations that constitute the Internet architecture. At the core, transit ISPs and the DFZ see IP addresses as locators that points to locations in the Internet topology. But as we move closer to the edge, we see the emergence of a new use for IP addresses: identifiers for end devices. At the edge, customer ISPs and organizations use IP addresses as identifiers for their customers.

Problem is, the core of the Internet benefits from keeping the RIB as small as possible for efficient routing and reduced router costs. To achieve this, addresses should be allocated with respect to the topology in which they will be used, so that they can be aggregated. But customer ISPs and large enterprises make use of different mechanisms to reduce their costs or make their infrastructure easier to maintain and more performant. As we saw in the previous sections, these mechanisms mainly lead to prefix de-aggregation, due to addresses being allocated as static identifiers for devices.

A single address cannot efficiently fulfill this dual role that the different actors of the Internet have for it, and as such, its separation has been a key mechanism for new network layer protocols [33], such as LISP, that have been developed to solve the scalability issue that the Internet is facing. This protocol will be presented in the next section.

## 3.2 LISP Introduction

The LISP protocol, short for Locator/ID Separation Protocol, is a network protocol that was developed to address the scalability challenges that have been described. The traditional Internet Protocol (IP) uses a single identifier, known as an IP address, to both identify the host and specify its location on the network. LISP addresses this issue by separating the host identification and location information into two distinct address spaces:

1. The Endpoint Identifier (EID), used to identify end-devices, is assigned independently of their topological location. It is used for intra-domain routing.
2. The Routing Locator (RLOC) are assigned topologically to network attachment points. They are typically routed inter-domain.

This marks a clear separation between the edge and the core of the Internet, as they both use different address spaces for different functions. They are now

| Term | Definition |
|---|---|
| *Endpoint Identifier (EID)* | Addresses used to uniquely identify nodes irrespective of their topological location. Routed intra-domain. |
| *Routing Locator (RLOC)* | Addresses assigned topologically to network attachment points. Routed inter-domain. |
| *Ingress Tunnel Router (ITR)* | A LISP-capable router that encapsulates packets from a LISP site towards the core network. |
| *Egress Tunnel Router (ETR)* | A LISP-capable router that decapsulates packets from the core of the network towards a LISP site. |
| *xTR* | A router that implements both ITR and ETR functionalities. |

Table 3.1: Terminology of LISP elements
[1]

logically separated. The link between the two is made thanks to LISP-enabled routers, which interconnect them and use a new database, the Mapping System, that supports the storage and retrieval of mappings between EIDs and RLOCs. These routers are located at the border of a LISP site and exchange packets by encapsulating them. With these routers and the Mapping System, no modification of the current Internet Architecture is necessary to adopt LISP. EIDs and RLOCs are identical to IP addresses and as such, both the core network and the edge can remain unmodified: the core network can keep routing packets based on their RLOCs, and end-devices will identify each other using EIDs. A table with definitions that will be used throughout this work can be found in Table 3.1. The term xTR will be used for LISP-enabled routers for the rest of this work.

LISP has several benefits, including the ability to support mobility and multi-homing, improve security and privacy, and enable more efficient use of IP addresses. It is an experimental protocol and is not widely deployed, but it has been implemented in a number of research prototypes [34] and is being studied by the IETF.

In the rest of this chapter, we will explore a detailed description of how it works and how it has been implemented in modern networks.

### 3.2.1 Data Plane

The Data Plane of LISP is described in RFC 9300 [1]. The main addition to the classic IP Data Plane is the process of encapsulating and decapsulating packets done by the xTRs, border routers of the LISP site.

To illustrate, we will consider a typical scenario in which a host inside the LISP

site sends a packet to some server. The destination EID used is often found using the DNS system, like a regular IP address. When an xTR receives this packet from inside the site, it will first check if it knows an RLOC associated with the destination EID. If it does not, it will contact the Mapping System to obtain it. Once the mapping is known, the xTR encapsulates the packet with an IP header, a UDP header, and a LISP header. This process tunnels the traffic between the two xTRs through the core network. When the packet is received at the border of the destination's site, the decapsulation is done by the site's destination xTR. The process is then repeated in the opposite direction, as shown in Figure 3.3. The RLOC space is highlighted in red while the EID space is in green.

The format of an encapsulated packet is given in Figure 3.4. The need to encapsulate with multiple headers instead of simply adding and outer IP header, comes from the filtering that can be done by some middle-boxes. An IP-in-IP encapsulation could be filtered by these devices when they don't recognize a classic packet structure, composed of an IP header encapsulating a TCP/UDP header. We can see that the packet now has multiple different headers:

1. An outer IP header with the xTRs' RLOCs as source and destination. This header will be used to route the packet between these two routers.
2. A UDP header, to avoid discarding by middle-boxes.
3. A LISP header containing LISP specific information.
4. An inner IP header which contains the source and destination EIDs.

These additional headers increase the overall size of the original packet, but that is the price to pay for compatibility with the existing architecture and reduced deployment costs.

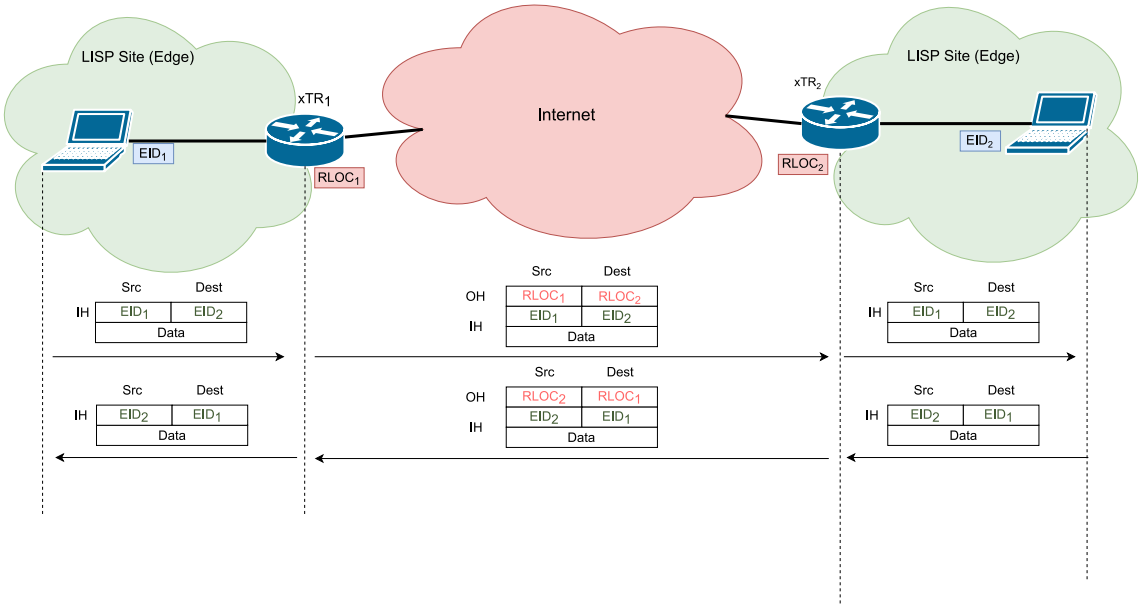


Figure 3.3: LISP Data Plane operations

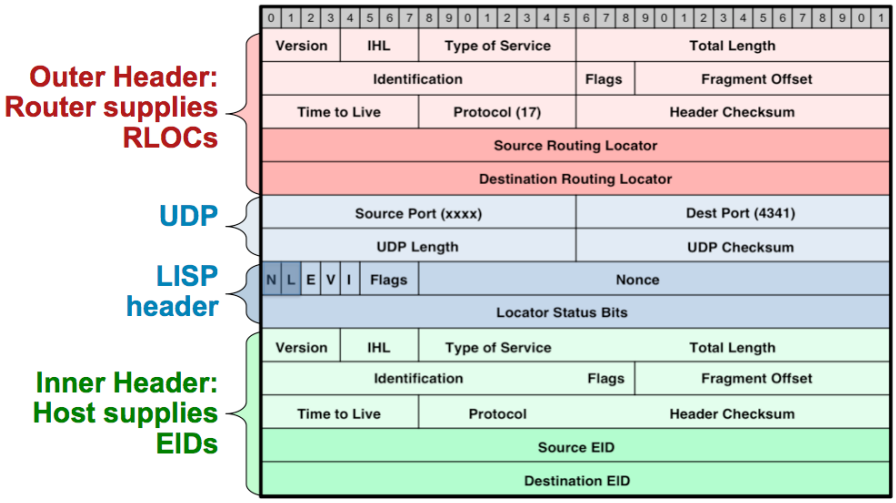


Figure 3.4: LISP header format
[35]

### 3.2.2 Control Plane

The key mechanisms of LISP lie in its Control Plane, which has an essential role in its architecture as it is response for registering, storing, and retrieving EID-to-RLOC mappings. These mappings are used to associate a list of RLOCs with an EID-Prefix. The details of this Control Plane have been defined in RFC 9301 [3].

#### Registering and retrieving: the interface

The LISP Control Plane has been developed with modularity in mind by decoupling the database used to store the mappings from the interface used by the xTRs to interact with it. This allowed multiple different database schemes to be developed in parallel, each with their own advantages and drawbacks. To do this, two new devices are used in LISP topologies:

- The Map-Server: Learns mappings from xTRs and publishes them in the Mapping System.
- The Map-Resolver: Used as a proxy by xTRs requesting mappings.

Four Control Plane messages have also been added for the communication between xTRs and the interface. Two of these messages, the *Map-Request* and the *Map-Reply*, are used to obtain mappings from the Mapping System. The *Map-Register* and *Map-Notify* are authenticated messages used by an xTR to register the EIDs for which it is authoritative to the Mapping System through its Map-Server.

To summarize how all these elements interact together, we will go through the process depicted in Figure 3.5.

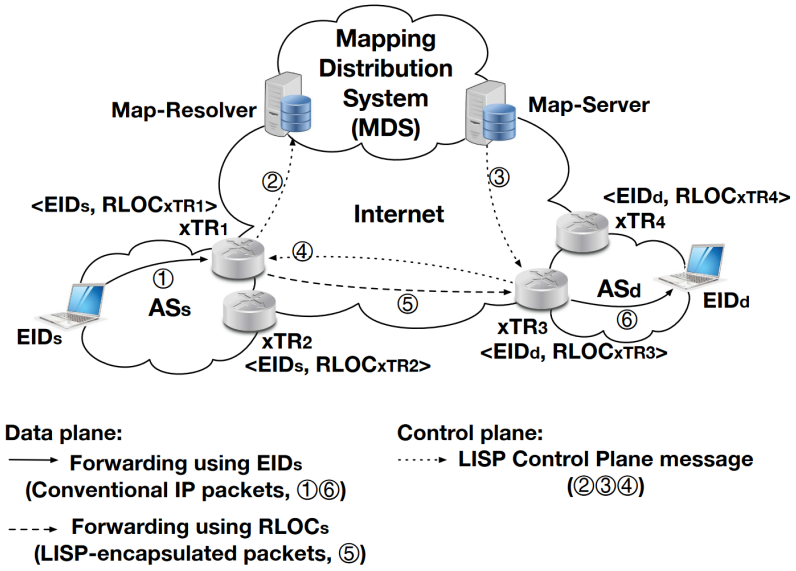


Figure 3.5: Delay perceived with no caching
[36]

1. First, a host in the LISP site will send a packet with a destination outside the LISP site.
2. When receiving this message, the xTR has to encapsulate it. If it does not have an entry with a RLOC associated to the destination EID, it sends a *Map-Request* to its Map-Resolver.
3. Through the mapping system, the Map-Resolver forwards the request until it reaches a Map-Server that knows the authoritative xTR for this EID and can forward the request.
4. The authoritative xTR directly send the *Map-Reply* to the requesting xTR, without going through the Mapping System again.
5. The message can now be encapsulated and sent towards its destination.

The same process must be repeated in the inverse direction if needed.

It is interesting to note that Map-Servers can be configured to answer Map-Requests instead of forwarding them to the xTR, to alleviate the workload of the xTR.

### Storage

Mappings can be stored at various places in the LISP architecture. First, each xTR has a *Mapping Database* containing the mappings for which it is authoritative and is thus derived from the initial configuration. This database is queried by the xTR to generate a Map-Reply when it receives a Map-Request.

The LISP mapping system introduces a new delay to a packet exchange: the delay generated by waiting for a response from the mapping system to a mapping request. To reduce this delay for successive requests, an xTR also has a *Mapping Cache* in which it stores recently received mappings. These mappings remain in the cache until their Time-To-Live expires.

### LISP mapping databases

Multiple specifications for mapping databases have been developed, the most well-known being *LISP Alternative Topology* (LISP+ALT) [37] and *LISP Delegated Database Tree* (LISP-DDT) [38]. In this work, a simplified version of LISP-DDT has been used for the simulations.

The main goal of LISP-DDT is to achieve a scalable mapping system. To do this, a hierarchical approach has been chosen, very similar to the architecture observed in the DNS system. To achieve this hierarchy, the EID space is divided between each node of the system (DDT-node). Starting from the root DDT node, which is responsible for the entire EID space, each child node is responsible for a portion of its parent prefix. Each parent node keeps track of its children, and the prefix it is responsible for. Going down the hierarchy means reaching smaller prefixes, as shown in Figure 3.6. At the bottom of the hierarchy we find the Map-Servers which can redirect requests to the authoritative xTRs.

Going through this architecture for each mapping that needs to be resolved introduces a non-negligible delay to data transmission. That is why using caching at both the xTR and the Map-Resolver is important to reduce this delay as much as possible. A Map-Resolver can be used by multiple different xTRs, similar to a DNS proxy of an ISP. Because of that, caching xTRs' RLOCs and the EID prefix for which they are authoritative, after going through the Mapping system, heavily reduces delay for scenarios where clients are behind different xTRs.

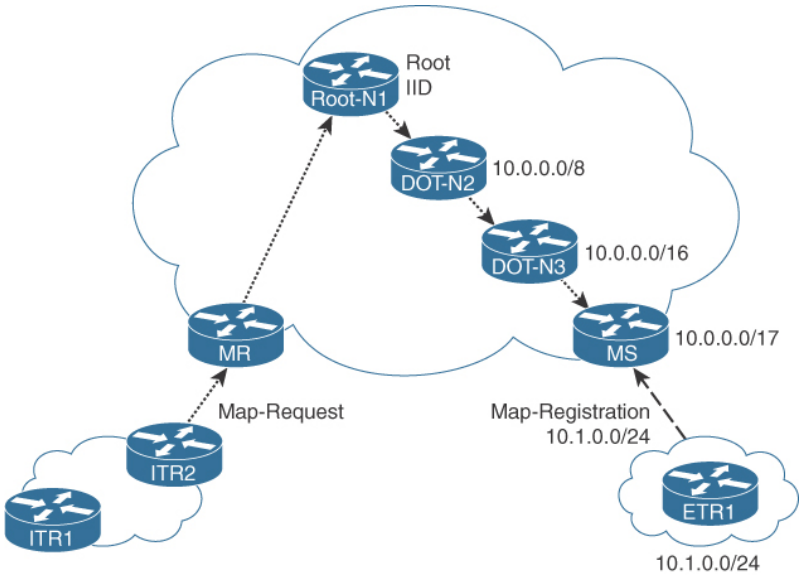


Figure 3.6: LISP-DDT hierarchy example
[35]

## 3.3 Benefits

With LISP, the scalability problem is solved. RLOCs are static addresses that denote attachment points of networks and are expected not to change much. They are assigned with respect to the topology, which leads to a high aggregability and a

low BGP churn. The EIDs, on the other hand, have a more dynamic nature. They can change due to a change of ISPs, configuration, or because of mobility events with end devices moving from network to network. With these events, only the Mapping System entries change to reflect these events, and the core network remains unaffected. Mobility and renumbering is thus well-supported by LISP.

Another benefit is that mapping entries can contain Traffic Engineering policies to achieve both multihoming and load balancing with minimal effort. Weights and priorities are used by the authoritative xTR to implement these policies. Priorities are used to indicate which RLOC should be used by the remote site (inter-domain TE) and weights allow load balancing for RLOCs with the same priority. These two values are dynamic and policies can be used to change them depending on the requesting remote site.

# Chapter 4

# LISP Network Privacy

## 4.1 Securing LISP

RFC 7835 [39] has identified a list of possible threats to the LISP protocol and multiple extensions are in development to provide security guarantees to both the data plane and the control plane of LISP.

To secure the control plane, LISP-SEC [4] is an extension that adds several security mechanisms to the EID-to-RLOC mapping communications to provide message authentication and integrity. For the data plane, RFC 8061 [40] describes how encryption keys could be exchanged using the current LISP architecture and how these keys can be used by xTRs, in the same fashion as IPsec, to encrypt and decrypt the inner IP header, its associated payload, the UDP header and the LISP header of a LISP packet. The outer header remains in clear for proper routing.

But these additions are not enough to provide privacy guarantees for LISP communications. Communication data is protected during transmission, but the devices involved in the communication are still identifiable at both ends.

Now let's review a series of models built upon the example provided by Hao et al. [12]. These models will be implemented in ns-3 alongside the original model and compared in the next chapter based on different metrics.

## 4.2 LISP Topology

The topology used for the LISP models and simulations is nearly identical to the one use for the IP addressless topology (defined in Figure 2.3) with LISP specific elements.

The Map-Resolver (MR) and Map-Server (MS) will act as the LISP interface to access the underlying mapping database system as specified in RFC 9301 [3], allowing our solutions to work with any mapping database system. Nevertheless, In the actual ns-3 implementation, LISP-DDT is the only LISP mapping system implemented, as such, it will be used for the simulations. Let's now proceed iteratively and find how we can benefit from privacy in a LISP network.

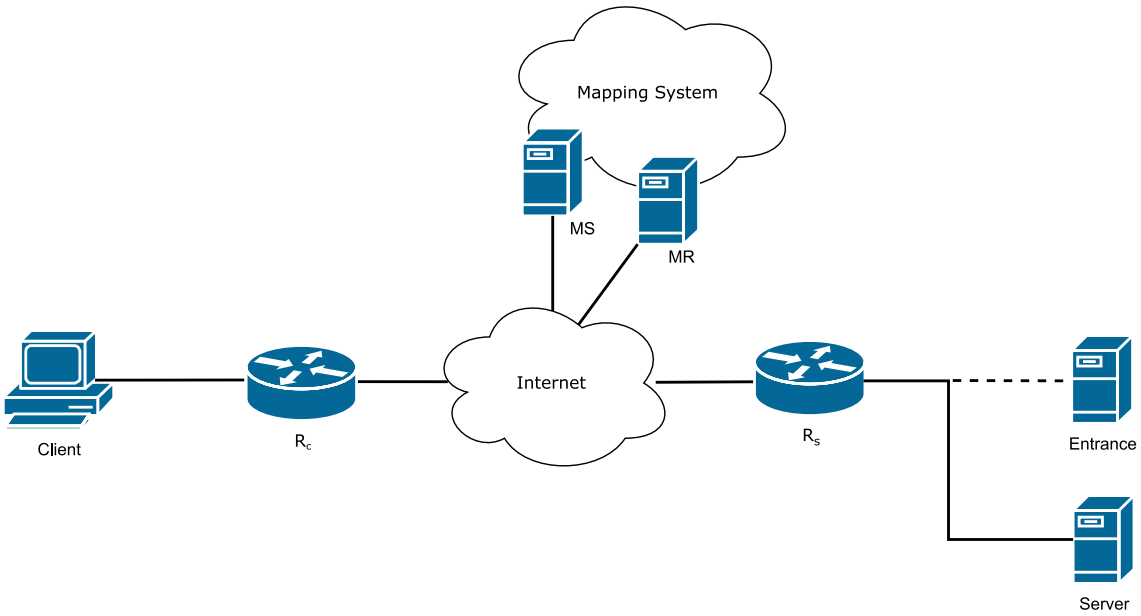


Figure 4.1: Default LISP architecture

| Name | Redirection mechanism | Server | Entrance module |
|---|---|---|---|
| *Addressless IP* | Active at app-layer | Modified | Additional device |
| *Addressless LISP* | Active at app-layer | Unmodified | Additional device |
| *xTR Redirection* | Active at app-layer | Unmodified | Server's xTR |
| *Map-Reply redirection* | Active at app-layer | Unmodified | Server's xTR or Map-Server |
| *Passive redirection* | Passive at network-layer | Unmodified | Not needed |

Table 4.1: Summarized comparison of models providing Identity privacy

## 4.3 Identity privacy

To achieve identity privacy, identifiability of the devices involved in communication must be reduced as much as possible. A serie of models, summarized in Table 4.1, will be presented in the next subsection.

### 4.3.1 Addressless LISP

An obvious (but naive) first solution that could be used with LISP would be a direct implementation of the IP model, with no modifications besides the LISP specific elements:

The steps are nearly identical as the IP version:

1. The client will start by sending a request to the service IP address obtained

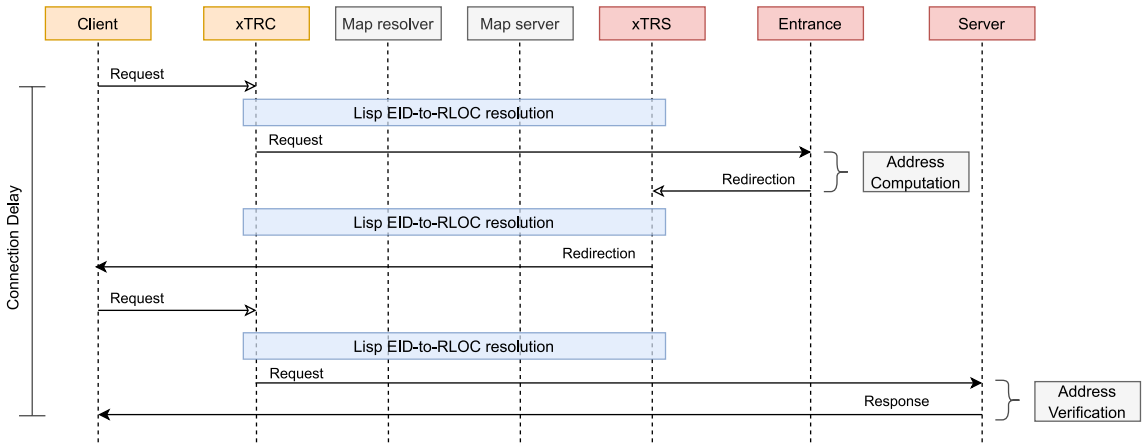


Figure 4.2: Packet flow: LISP addressless approach

through the DNS system.

2. The client's xTR performs an EID-to-RLOC mapping lookup to obtain the entrance module's xTR RLOC.
3. After receiving the request, the entrance module uses
   (a) The prefix of the main service module.
   (b) The source address of the packet

   to generate an IPv6 address through the encryption function.
4. It will return this generated address to the client.
5. The entrance module's xTR performs an EID-to-RLOC mapping lookup to obtain the client's xTR RLOC.
6. Finally, the client initiates a connection with the main service module using the received address.
7. Depending on the prefix advertised, the client's xTR may require to perform an additional mapping lookup.
8. Before establishing the connection, the main service module verifies the destination address used, if it checks out the sever carries on with the connection, otherwise, the packet is dropped.
9. Communication takes place as usual with no modifications.

This solution has the same advantages and drawbacks as the IP version, the only additional drawback being the delay due to the LISP environment, in this case we have up to three EID-to-RLOC mapping resolution that needs to be done before the connection can take place. That is of course the worst case scenario, where the entrance module and the server are in different subnets, if both the entrance module and the main module were located in the same subnets, the server xTR could choose to advertise the subnet prefix in its response to the first mapping request, and only two mapping resolution would have been necessary.

### 4.3.2 xTR Redirection

In this solution, we get rid of the entrance module, taking advantage of our xTR to perform both the address generation and verification. The server's xTR will take on the job of the entrance module and will have a public IP address associated with the server's service. Of course, this address must be configured as an AAAA records in the DNS system.

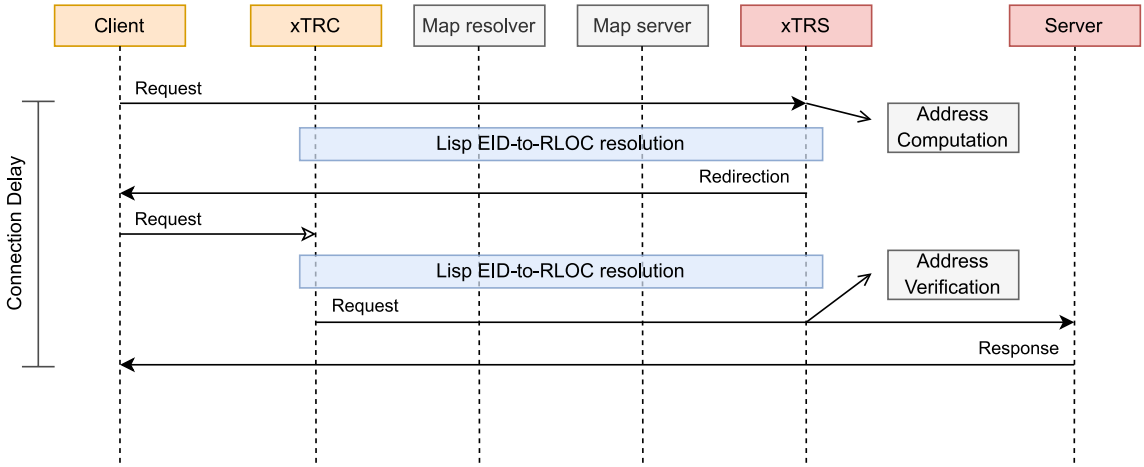


Figure 4.3: Packet flow: Redirection by the xTR

The steps taken are described in Figure 4.3, the main differences with the previous model being the role taken by our xTR. It is interesting to note that the verification process can be done in either a stateless or stateful approach by our xTR. The stateful approach would be to keep track of each line (source address, generated address, time to live) that has been generated by this xTR and check each packet. A stateless approach would be to only check the first packet in a given flow.

This model simplifies the architecture, as we do not need an additional physical device anymore. As a consequence, the mapping required to reach the entrance module is not needed anymore, reducing the connection delay. The second interesting consequence is that our server is now unmodified since the xTR took on the job of verifying the destination EID.

This new model still holds the properties of the previous one. Our xTR can support multiple services as an entrance module by simply having multiple different IP addresses or ports associated with each service. The position of the xTR is not a constraint either, any xTR in the topology can perform the job.

### 4.3.3 Map-Reply redirection

To reduce delay even further, the address redirection message can be send when a Map-Request for the server EID is received.

Using the same setup as the previous model, the communication steps are:

1. The client sends the request to the address obtained thanks to the DNS system.
2. The client xTR will send a LISP Map-Request in order to get the destination RLOC associated to this EID.

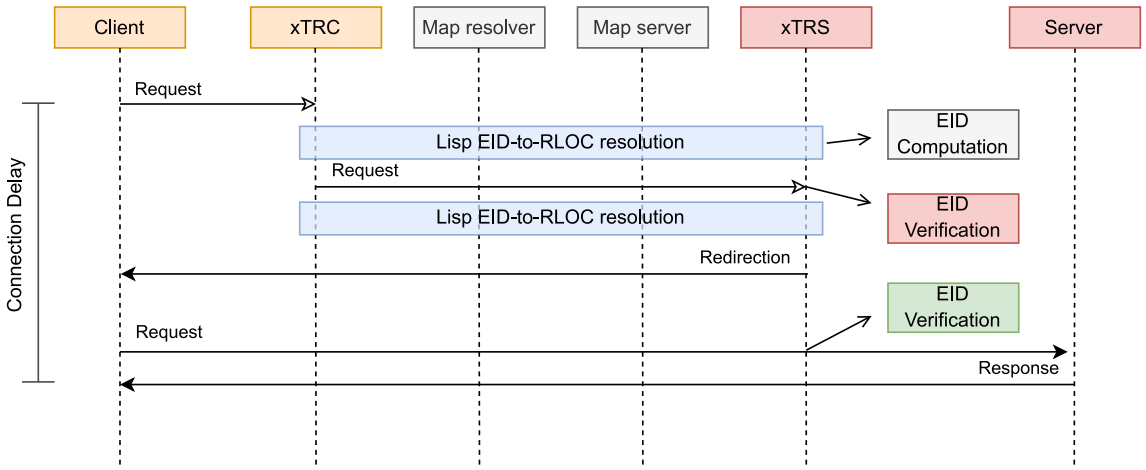


Figure 4.4: Packet flow: redirection with Mapping reply

3. After receiving the request, the server xTR uses the prefix allocated to the server and the EID of the client to generate an IPv6 address through the encryption module. It will send:
   (a) its RLOC in a LISP *Map-Reply* message.
   (b) the computed server EID in a *address redirection* packet addressed to the Client.
4. Finally, the client initiates connections with the server using this address as the destination address.
5. The **server's xTR** verifies the destination address used.
6. Communication takes place as usual.

The server's xTR sends the address redirection alongside the Map-reply in order to minimize the delay caused by this additional pair of messages. If multiple clients are behind the same xTR, RLOC caching must be taken into account. The first client's request will trigger a mapping resolution, while subsequent requests made by the other clients will not. Therefore, our server's xTR must be able to fall back to the classic xTR redirection explained in the previous section for these clients.

### 4.3.4 Passive redirection

So far only solutions using active redirection have been explored, instead, a passive method could be used to silently redirect incoming traffic. The server's xTR will act like a traditional NAT would do, replacing addresses of incoming and outgoing packets but using the encryption module discussed before.

The setup is similar to previous models, with the exception that, in this scenario, the public IP address associated to the service may or may not be allocated to a device in the topology. This IP address is only used to reach the service.

The steps, depicted in Figure 4.5 are as follows:

1. The client sends the request to the address obtained thanks to the DNS system.

2. The server's xTR, when decapsulating the request, will check that:
   (a) This packet is not addressed to any of the IP addresses allocated to the server(s), otherwise the packet is dropped.
   (b) If this packet is addressed to the service, it is addressed to the port(s) used by the service, otherwise the packet is dropped.
3. It will then replace the destination address of the packet by the one generated using the encryption module.
4. The server will receive the modified packet and answer to the client.
5. The server's xTR, when encapsulating the packet, will replace the source IP address of the server by the service IP address.
6. Communication takes place as usual with this additional translating system in place.

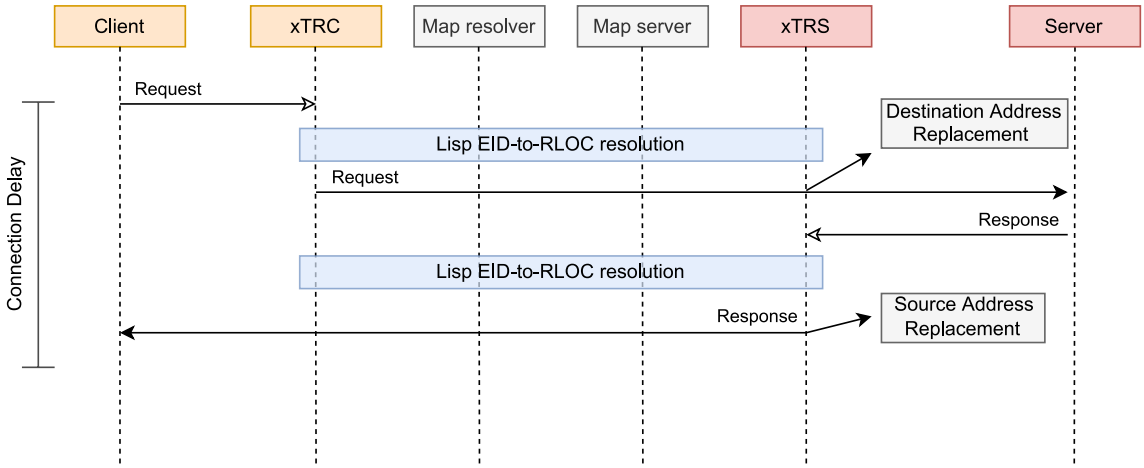


Figure 4.5: Packet flow: Passive redirection

This solution is now completely located at the network layer. This is a huge change comparing to the original model for IP networks, where all mechanisms were located at the application layer. The first obvious advantage is that the model can be used for any application layer protocol whereas previous solutions required the application to include some form of redirection mechanism. Now both the server and the client are left unmodified. The other key advantage is clients are no longer talking to a server but to a service, with no information about which server is answering the requests. The servers are not directly reachable through the IP addresses that have been allocated to them, but only through the unique IP associated to the service, making them less vulnerable to brute-force scanning of the address space. Any client has to use the service IP in order to reach the server. This model can also easily be adapted to provide privacy for clients, as it is described in the following sections.

Still, compared to active redirection mechanisms, we could think that if the service IP address is collected using one of the methods mentioned earlier, it would leave the servers that provide the service vulnerable to scanning because they are now directly reachable, without any application layer specific redirection mechanism. However,

since a sequence of flows initiated by a malicious actor won't be mapped to the same server and since our router limit the traffic going towards the servers to the ports the services use, in the fashion of a firewall, this model severely reduce the scanning ability of an attacker.

The impact that the address replacement will have on delay and performance in case of a high amount of traffic will be evaluated thanks to the ns-3 implementation.

## 4.4 Location privacy

The same notions that we used to hide the identifier of a server can be used to hide its locator.

In this scenario, The server's xTR will be allocated a prefix of RLOC addresses and will use a specific RLOC for each new connection to the server. It performs checks to ensure that the appropriate RLOC is used for flows between a client and the server. Any packets that do not have the appropriate destination RLOC will be discarded.

The communication steps are defined as follows:

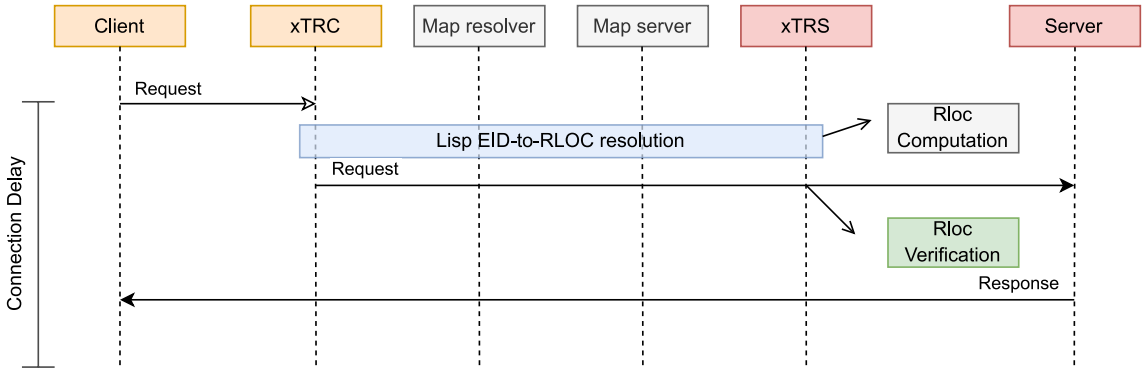


Figure 4.6: Packet flow: RLOC hiding model

1. The client sends the request to the address obtained through the DNS system.
2. The client xTR will send a LISP Map-Request in order to get the destination RLOC.
3. After receiving the request, the server xTR uses the prefix allocated to it and the source EID to calculate an IPv6 address through the encryption module.
4. Then returns this address to the client xTR in a LISP Map-Reply message.
5. Finally, the client xTR encapsulate the client request in a LISP Packet.
6. The server xTR, when receiving LISP packets, verifies the destination RLOC.
7. Communication takes place as usual.

This model is simple and do not require any modification on both ends of the connection, as everything is done in the control plane. In the scenario where proxy

map reply is requested, then the Map-Server will perform the generation of RLOC addresses. This model, like the previous one, can be applied on the client side with no modification. It only adds very few additions to the complexity, one encryption per map request and one encryption per incoming flows to verify the destination RLOC being used.

Unlike the models used to provide identifier privacy, the period of time during which an RLOC is associated with a source address is slightly larger, as it has to accommodate the possible RLOC caching that can be done on either the Map-Resolver or client's xTR side. Another possibility would be to use LISP Publish/Subscribe [41]. This extension of LISP control plane is used to allow xTR to notify other xTRs of changes in mapping. An xTR could use this mechanism to notify maintain the customized RLOC up to date during a communication.

## 4.5 Complete privacy

So far we have only studied identity and location privacy separatly but any solutions described so far can be combined to achieve both location and identity privacy.

In this scenario we will use both the RLOC redirection and the passive EID redirection.

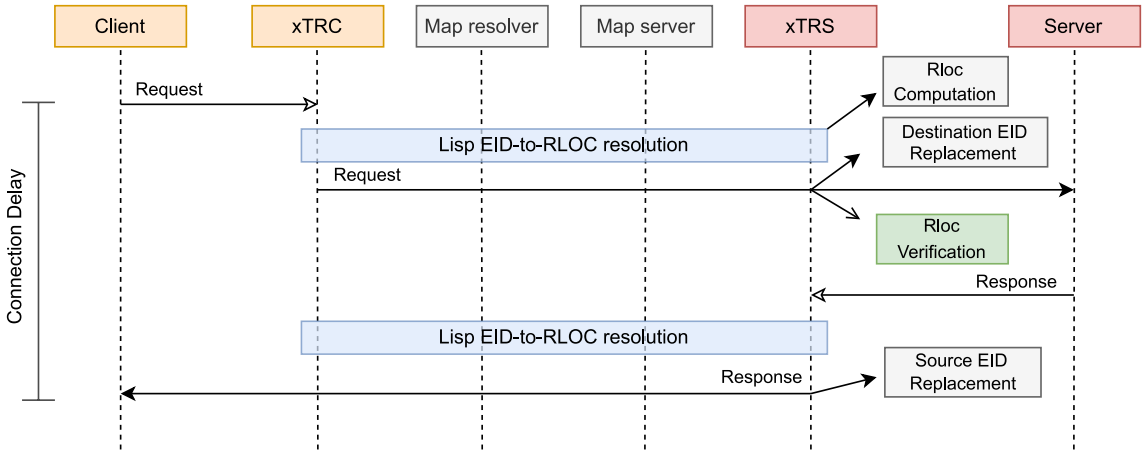


Figure 4.7: Packet flow: Association of Identity and Location privacy

As a result, we now have a model that provides the same privacy properties as the one presented by Hao et al., but with several key advantages. It is easier to deploy and maintain because it does not require any modification to the client or server and does not require any additional devices. It can be used for any application-layer protocol since it performs the redirection at the network-layer, and also achieves better delays, as we will see in the next chapter, because it does not require an additional connection to be established.

We can also easily apply our server privacy model to the client side. In this case, the xTR passive redirection functions like a NAT, replacing the source address of outgoing packets with a generated IP address that depends on the prefix allocated to the client and the destination address of the packet. This allows us to obtain both identifier and locator privacy for the client.

# Chapter 5

# Implementation in NS-3

In Chapter 2, new privacy models for LISP were developed based on the "Addressless" model proposed by S. Hao et al. for IP networks [12]. These models were discussed on a theoretical level. In this chapter, these models will be simulated in the ns-3 network simulator using the existing LISP implementation. This will allow us to compare the performance of the different privacy models in the next chapter using metrics gathered from the ns-3 simulation.

Before diving into the implementation of the privacy models, this chapter will provide an overview of ns-3 and describe the existing LISP implementation, followed, by a description of the modifications made to implement the privacy models.

## 5.1 Ns-3: Network Simulator

Ns-3 is a discrete-event network simulator, targeted primarily for research and educational use. It provides a simulation environment for modeling network protocols and technologies, as well as for evaluating their performance.

Compared to other network simulators, ns-3 is open-source and has no graphical user interface. It is entirely implemented in C++, for performance and structure, and provides a Python interface for scripting and control. Users can utilize the existing codebase of implemented protocols or design their own protocol for any network layer. The basic architecture of ns-3 is shown in Figure 5.1, which was imported from the ns-3 annual meeting of 2017 [42]. In this figure, we can define a ns-3 *Node* as "a shell of a computer, to which applications, protocol stacks, and NetDevices are added", according to the same document. As for the *NetDevices*, they represent the network interface card of computers.

In ns-3, a simulation typically consists of creating a network topology, defining the behavior of each component of the network (e.g., nodes, links, and protocols), running the simulation, and gathering data through their built-in tracing system.

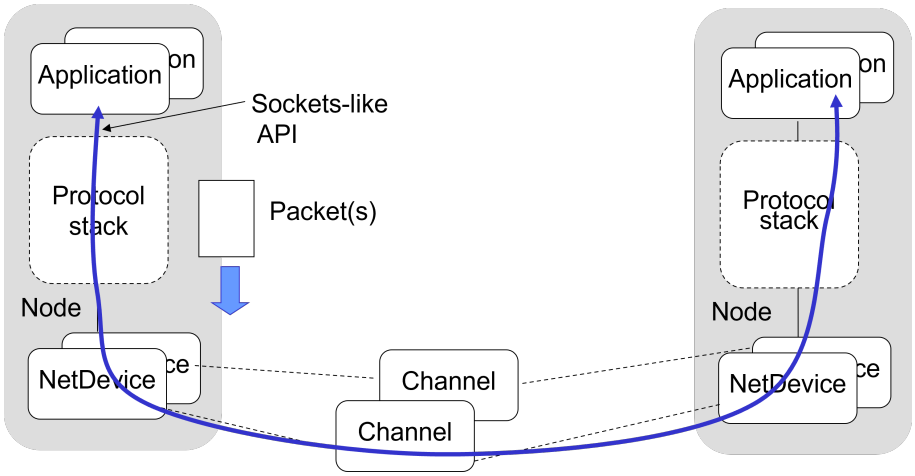


Figure 5.1: Ns-3 basic architecture
[42]

### 5.1.1 Time and ns-3

As previously mentioned, time and events are simulated through the use of a discrete-event simulator. This means that ns-3 doesn't have any knowledge of real time, the simulator simply executes a series of discrete events, rather than simulating a continuous flow of time, and it has critical implications for our simulations.

A simulation typically consists of an initial list of events, each event has a specific execution time associated to it and the simulator executes them in chronological order based on this time stamp. An event can lead to the scheduling of new events, as depicted in Fig. 5.2. This allows the simulator to model the behavior of the network over time. Ns-3 provides various functions to help manage and schedule events, such as the ability to cancel or reschedule events, and to schedule events to occur at regular intervals. All these functions are accessible through the *ns3::Simulator* class.

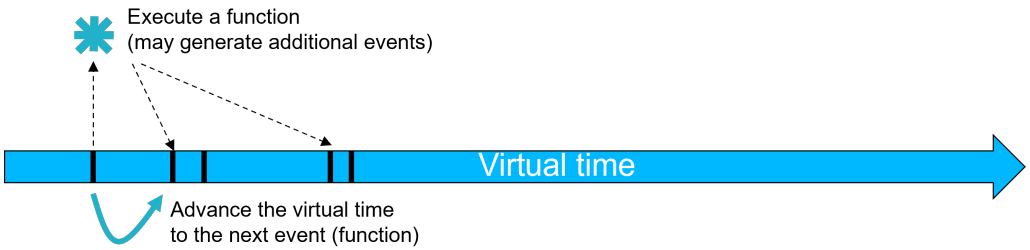


Figure 5.2: Ns-3 events queue
[42]

An important consequence of this simulation method is that events are considered instantaneous. The virtual clock is paused during the execution of an event and the simulator jumps from one event to the next until there are no more scheduled events in the queue. For example, an event lasting 3 seconds in the simulation may have taken hours of real processing time.

This has two side effects. First, the limits of an application's performance must be explicitly provided, otherwise, the application will run until exhaustion of the

resources of the hosting computer. Secondly, To implement things such as delay, explicit timing models must be provided. The models will delay the start of the next event by increasing its timestamps to simulate these delays.

For example, in ns-3, a packet transmission event may be scheduled at a specific time, and when that time arrives, the packet is transmitted according to the rules of the protocol being simulated. Then, event(s) to process the received packet at the receiver(s) end may be scheduled at a later time. The time between the transmission event and the reception event will thus represent the link delay and will typically be handled by the link implementation.

### 5.1.2 Code architecture

The code that composes ns-3 is organized in modules composed of code, documentation, examples and helper classes. These helper classes are used to simplify the process of creating and configuring the objects defined in the main code of the module without having to worry about the low-level details of the code.

Ns-3 code itself is implemented using an object-oriented architecture, the different components of the simulator are implemented as classes, with each object having its own set of attributes and behaviors. This allows the simulator to be modular and flexible and lead to a high reusability of the existing components.

## 5.2 LISP in ns-3

This LISP implementation in ns-3 has been developed and updated over the years by successive authors. The initial version has been created by L. Agbodjan [43] and included the basic LISP functionalities, namely the Data Plane, the Control Plane, and the Map-Server/Map-Resolver interface. Its work was updated to the latest ns-3 version by Y. Li [36] and extended to add support for LISP-MN [44].

E. Marechal [14] et T. Piron [45] also added some LISP extensions to the project, LISP+NAT [46] and LISP Publish-Subscribe [47], respectively. This work uses the latest LISP implementation available at its start, the version of E. Marechal.

The LISP implementation relies heavily on the existing IP implementation, and adds functions calls in this existing code to implement LISP specific mechanisms. The following subsections will give an overview of the LISP implementation and then highlight the modifications that have been done to it.

### 5.2.1 Code Structure

The implementation choice that has been made is to separate the Data Plane and Control Plane in two different ns-3 applications, that communicate through a UDP socket called the LISP mapping socket, as depicted in Figure 5.3. The color code defined in this figure will be used in the following graphs.

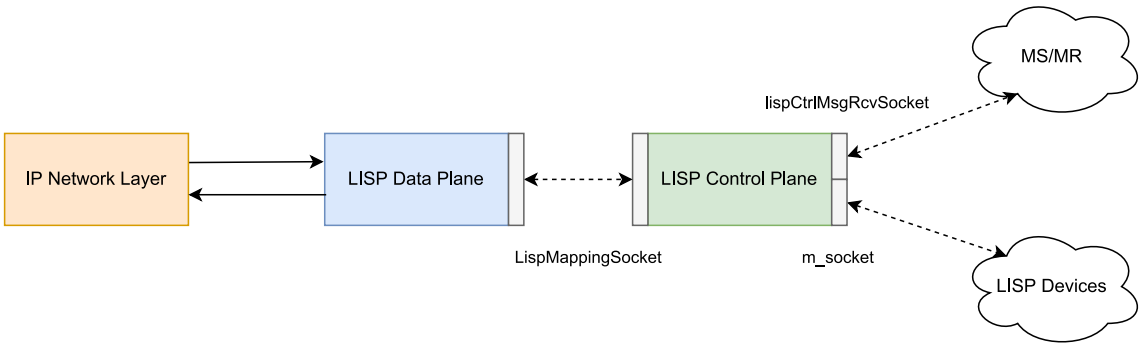


Figure 5.3: LISP code architecture in ns-3
– →: packet exchanged through the named sockets
→: function calls

### 5.2.2 LISP Data plane

The LISP Data plane in ns-3 is responsible for the encapsulation and decapsulation of packets and is mainly located in two classes: *LispOverIP* and its child classes are responsible for the handling of encapsulation and decapsulation packets and the *MapTable* class is used to describe the interface needed for the data structure containing the EID-to-RLOC mapping. Currently, only a simple implementation of this data structure as a list exist and is described in *SimpleMapTable*, it is used for both the mapping cache and mapping database. The packet forwarding is done by the IP code.

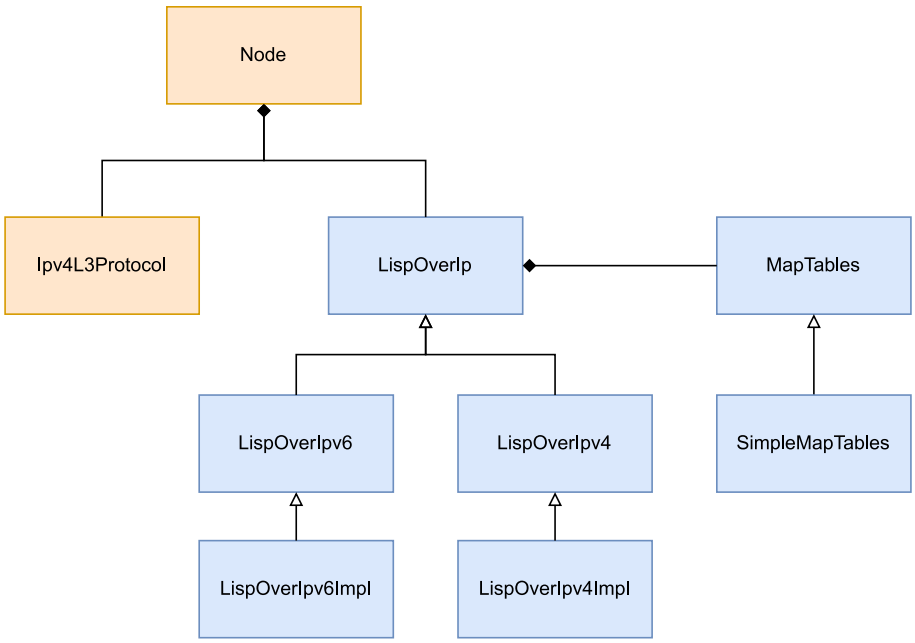


Figure 5.4: Data Plane architecture in UML
[14]

The encapsulation process is depicted in Figure.5.5. Code has been added to the existing *Ipv4L3Protocol* class to check, before sending a packet, if it needs to be encapsulated or not. This check has three possible outcomes:

1. The packet doesn't require an encapsulation and can be forwarded to the lower

layer code.

2. The packet requires an encapsulation and the associated RLOC is found, either in the LISP cache or database. The packet is encapsulated in *LispOutput* and then sent.

3. The packet requires an encapsulation but the associated RLOC has not been found. Packet is dropped and a message is sent to the Control Plane to send a Mapping request.

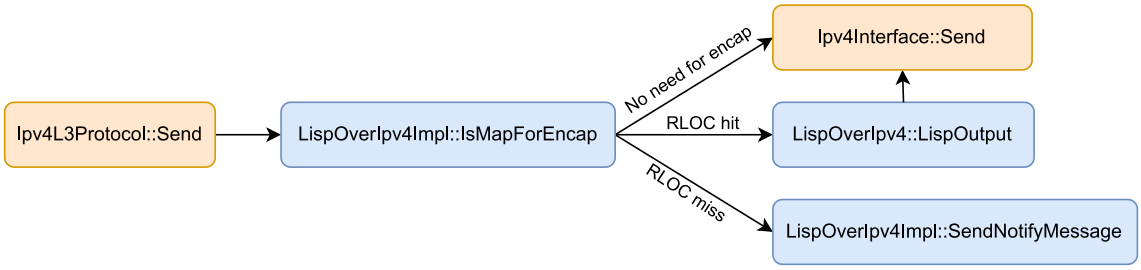


Figure 5.5: Encapsulation process

In the same fashion, calls to *LispInput* have been added to the *receive* function of *Ipv4L3Protocol* to decapsulate the packet if needed. It is interesting to note that the code will loop and decapsulate the packet multiple times if required, which can be the case with the mobility extension for LISP [44].

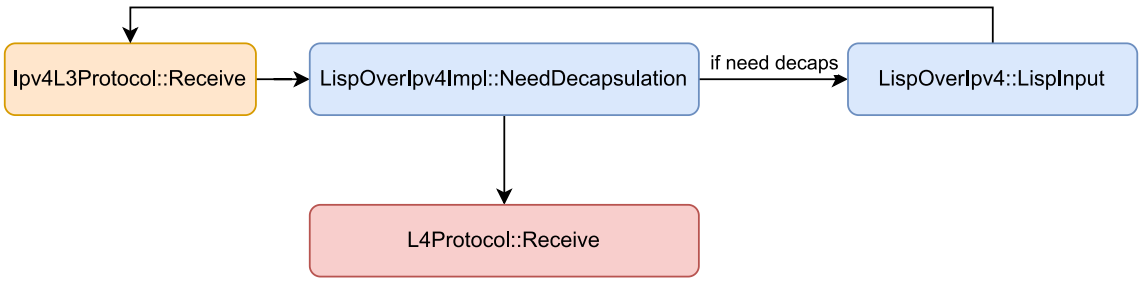


Figure 5.6: Decapsulation process

### 5.2.3 LISP Control Plane

The LISP control plane in ns-3 is made of three components:

1. The *LispEtrItrApplication* is responsible for the Control Plane operations.

2. The *MapServer* and *MapResolver* describes the interface used for the mapping system components.

3. The *LispControlMsg* and its children classes define all the different Control Planes messages format of LISP.

When starting, the *LispEtrItrApplication* will register the content of the LISP database (EIDs for which it is authoritative) to its Map-Server. After that, it awaits requests or messages from the Data Plane or other LISP devices.

When the Data Plane doesn't find the destination RLOC associated with a packet, it will drop the packet and notify the control plane, by sending a packet through

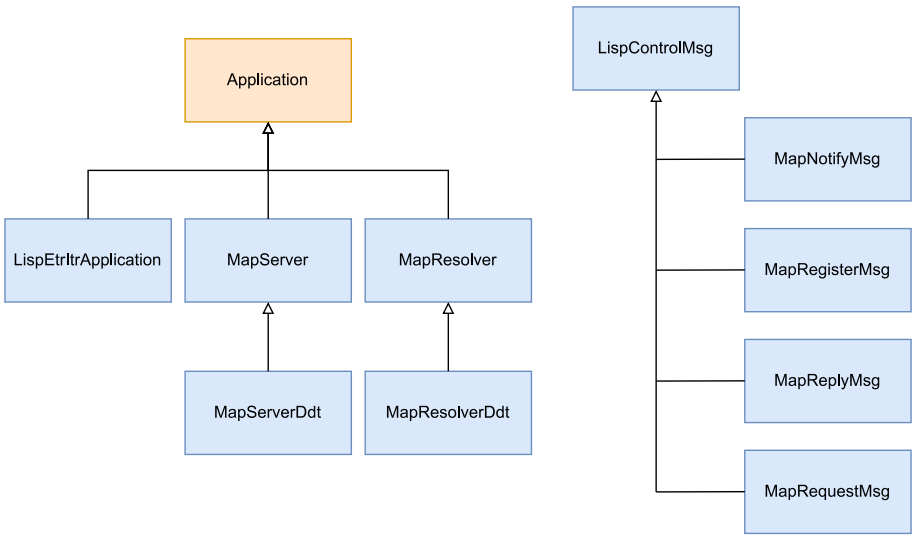


Figure 5.7: Control Plane architecture in UML
[14]

the LISP mapping socket. Upon reception of this packet, the Control Plane will send a Map-Request to the Map-Resolver. When it receives the Map Reply sent by either the destination's xTR or its Map-Server, it notifies the Data Plane so that the mapping entry can be added to the LISP cache. Both situations are shown in Figures C.1 and C.2 of the annex.

### 5.2.4 Map-Resolver and Map-Server

In the current LISP implementation, only a simplified version of LISP-DDT [38] has been developed. Instead of having a complete distributed and hierarchical architecture like in the DNS, everything is resumed to one Map-Server and one Map-Resolver serving all clients. The Map-Server is in charge of storing the EID-to-RLOC mappings in its LISP database during the registration process, and either redirecting to the authoritative xTR or directly replying with the proper mapping when receiving mapping requests. The Map-Resolver is a simple proxy that forwards the requests received from the xTRs to the Map-Server.

## 5.3 Modification to NS-3

A certain number of modification have been done to the LISP implementation, either to improve the code and address some shortcomings or to implement the solutions described in Chapter 3.

### Packet buffering

As outlined in the Data Plane subsection, when the xTR does not find the RLOC associated to the destination EID while encapsulating a packet, the default behavior is to drop the packet and generate a Map-Request. After that, the xTR waits for the Map-Reply and for a possible re-transmission of the discarded packet. For our measures, it would induce far too much additional delay as we would need to

wait for the default TCP re-transmission delay to trigger before having the client request sent. To change this behavior, the ns-3 IP Data Plane *ipv4-l3-protocol* and LISP Data Plane *Lisp-over-ipv4-impl* and *lisp-over-ip* have been modified to buffer a packet when a Map-Request is required, and send this packet when the corresponding Map-Reply is received.

## Map-Request fields

In the base implementation of LISP in ns-3, mapping requests did not contain the *Source EID Address* field which contain the EID of the client that triggered the Map-Request, as described in RFC 9301 [3]. Modifications have been made to the LISP Control Plane to both the MapRequestMsg message format and the *lisp-etr-itr-application* operation to include this field in map-request when needed. This allows the implementation of the LISP RLOC privacy, which requires the source address of the client to be known to generate a customized RLOC.

## Map-Server Proxy map Reply

When registering an EID prefix to a Map-Server, an xTR can set the proxy map reply bit in its *Map-Register* message to indicate to the Map-Server that it can answer to mapping requests for this prefix on behalf of the xTR.

Modifications have been made to both *lisp-etr-itr-application* and *map-server-ddt* to include the proxy map reply bit in their processing of the *Map-Register* message. The Map-Server has also been modified to answer to Map-Request when needed.

## Map-Resolver Caching

When sending a mapping request, a Map-Resolver in LISP-DDT must first go through the distributed architecture to find the authoritative Map-Server and xTR. In the same way as in the DNS infrastructure, this adds to the delay observed when initiating a connection to a server. Modifications have been made to *map-resolver-ddt* to implement a simple caching mechanism.

With caching, clients that have xTRs sharing the same Map-Resolver will only suffer from the additional delay if the authoritative xTR for a given EID is not present in the Map-Resolver cache. This is very similar to an ISP providing a DNS proxy service to its clients as this proxy will usually do some caching as well.

## Privacy extensions

The ns-3 LISP Data Plane *Lisp-over-ipv4-impl* has been modified to include the address checking required for both the EID and RLOC privacy. During the decapsulation process, code has been added to check, if needed, that the proper destination EID and RLOC are used to contact the server. Modifications have also been made to the LISP Data Plane *lisp-over-ipv4-impl* encapsulating and decapsulating processes to include the proper translation of source and destination addresses for the passive solution.

The LISP Control Plane *lisp-etr-itr-application* has been modified to include the dynamic generation of the RLOC addresses included in the map replies.

## 5.4 Extension to NS-3

New classes have been added to the existing ns-3 implementation.

Since the IP privacy model and several LISP privacy models use application layer redirection, three applications have been added to represent the client, the server and the entrance module in the simulations. They can use TCP or UDP and exchange custom-made application-layer messages to communicate. These messages can either contain data, or redirection data such as the new IP to contact. The state machine of the server, the client and the entrance module can be found in annex B.

A short overview is given.

1. The client will start by initiating a connection with the IP it is configured with, either the entrance or the server. When it receives a message that is not a redirection, it logs the connection delay. Then, it sends data until the end of the simulation.
2. The entrance module is a simple application that replies with a redirection message to any received message. This redirection message contains an IP address generated with the encryption process described in Chapter 2.
3. The server will accept any incoming connection that passes the destination address check, if address checking is required. After accepting a connection, the server receives packets and will log at the end of the simulation information about the transmission such as the quantity of data exchanged and the total duration of the transmission.

### Clients queuing

Modifications have been made to all privacy related code to include clients queuing delays. Without this modification to the basic ns-3 implementation, the server would be able to handle an infinite number of client requests, without additional delay for the clients. These modifications allow the code to schedule arriving requests after the last request that has been scheduled but has not yet been processed.

# Chapter 6

# Evaluation and Comparison

The aim of this thesis is to compare the privacy models developed for the LISP protocol with the original model developed for IP networks. This chapter will begin by exploring the choices and methodology used for the comparison of these models, it will then iterate through multiple different metrics and compare each solution.

## 6.1 Methodology

The collection of data is done using the ns-3 implementation, the base topology used in the simulation are depicted in Figure 2.3 and Figure 4.1, with links of 1 Gbs and 5 ms of delay. The clients, servers, and entrance module use TCP for their connections and the tracing system of ns-3 is used to gather the data during and at the end of the simulation, as depicted in the state machines of both the client and the server in Appendix B.

Each model has been run using three different scenarios to gather data:

1. Clients behind the same xTR sending requests one after another.
2. Clients behind **different** xTRs, using the same Map-Resolver, sending requests one after another.
3. Another scenario with hundreds of simultaneous clients, used to measure the scalability of our models.

The three scenarios will allow the comparison of performance of all the models presented in situations where caching is or is not available and the effect of simultaneous clients on the delay perceived. Each scenario has been run once for each model, as ns-3 is deterministic and we do not make use of any non-deterministic value, each run yields the same results.

### 6.1.1 Timing models

As explained in section 5.1, timing models must be explicitly given to the ns-3 simulator so that an accurate measure of the delay can be made. To do so, the sources of delay must first be identified.

From Figure 2.4 and Figure 4.2, we can identify several sources:

- The delay of the links used in the topology.
- The packet processing in Routers and xTRs.
- The time needed for the Client to change connection.
- The address generation and check performed at the server or at the xTR.
- The LISP mapping system.

To determine values for these sources, we need to make some assumptions. First, we consider the LISP and IP networks to be in identical deployment situations. We do not compare the existing IP network with an existing LISP network, such as the LISP beta network. To make this comparison, we would need to model the difference of average RTT between IP routers and between LISP xTRs from these real-life implementations. It is important to keep in mind that the goal of thesis is to compare privacy models. Therefore, the focus will not be on studying the differences between IP and LISP, as this has been done in previous works, but rather on the differences between several privacy models developed on top of these two protocols. A second assumption is to consider routers and xTRs processing to be instantaneous, we consider these delays negligible, except for privacy related computations such as address generation and checking.

For privacy related delays, we use the worst-case values given by empirical data gathered by S. Hao et al. [12], which are represented in Figure 2.5 and Figure 2.6. Finally, there are two options for determining the resolution of the mapping system. The first would be to estimate the resolution delay using data from the LISP beta network, which also uses LISP-DDT for the mapping system. The second option is to consider LISP-DDT to be similar to the DNS infrastructure and use DNS resolution delays as a reference. Since the comparison is made using the same deployment context for both LISP and IP networks, the second option has been used in this project. The mapping resolution delay used is the average DNS lookup delay measured by sending web requests to a dataset of the most popular websites (Alexa top 1 million). The average delay measured was of 20ms.

### 6.1.2 Results

The results of the first scenario are given in Figure 6.1 and Figure 6.2. Figure 6.1 presents the connection delay experienced by the first client sending a request to the server or entrance module, broken down into three components: the delay caused by the LISP mapping system (mapping resolution delay), the delay caused by the links (propagation delay), and the delay caused by privacy mechanisms.

We see that when there is no caching (e.g., when it is the first request sent to a service), this first client experiences the full mapping delay. Addressless LISP,

with its three mapping resolution, suffers the most with a total mapping delay of nearly 150 ms, while the other models have a total mapping delay of 100 ms for two mapping resolutions. The privacy delay reflects the impact of both the processing delay caused by the generation and verification of addresses and the redirection at the client side. As expected, it is the redirection that has the greatest impact, with a delay of nearly 10 ms for the active redirection models. In terms of propagation delay, the solutions using an active redirection have the highest delay because the client must establish two TCP connections before reaching the server. Since TCP is used in our simulations, this means going through the TCP three-way handshake twice.

Figure 6.2 shows the delay experienced by the clients sending requests after the first one. Thanks to map caching at the xTR, these clients don't experience any mapping resolution delay. This figure highlights the significant difference between the models' propagation delay.

Finally, Figure 6.3 displays the results when clients are behind different xTRs and cannot benefit from mapping caching at the xTR. In this case, caching at the Map-Resolver is used to reduce delay. When caching is available, it reduces each mapping resolution delay by half.

The third scenario's results are depicted in Figure 6.4, which shows the evolution of the delay experienced by each client. This delay increases, as clients experiences more and more queuing delay due to the privacy mechanisms.

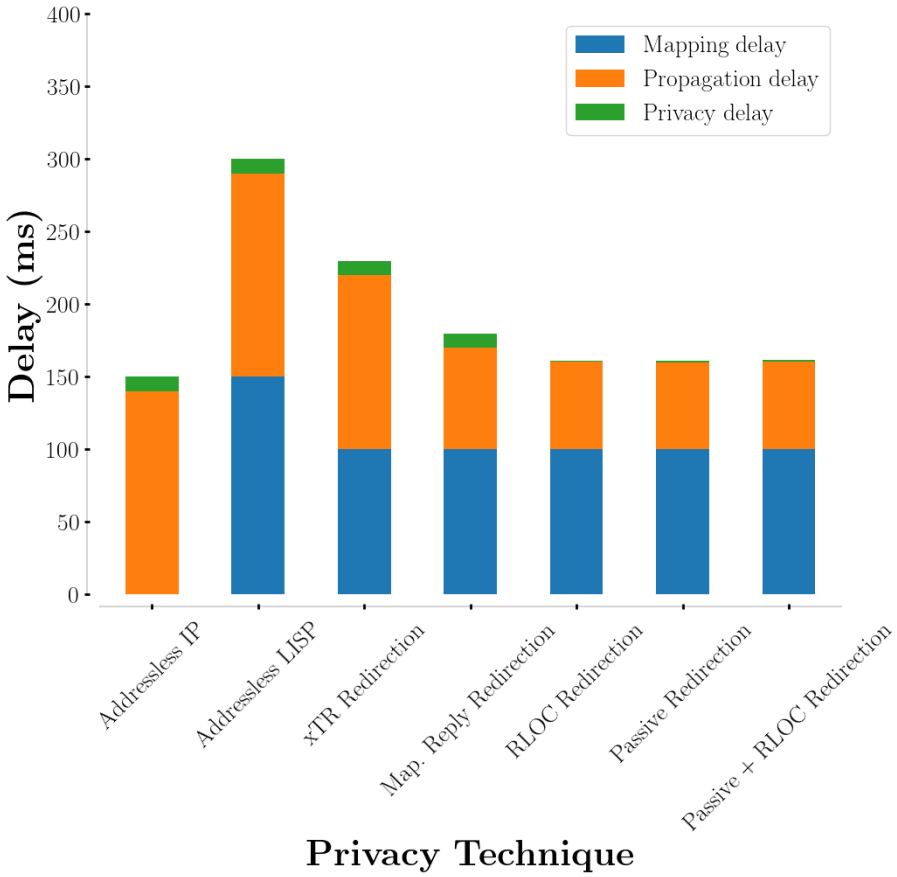


Figure 6.1: Delay perceived with no caching

## 6.2 Interpretation and Comparison

With these results gathered, we can now compare these models using the following metrics:

1. Connection delay
2. Scalability in worst-case scenario
3. Impact on the throughput of the connection
4. Implementation complexity and deployability
5. Privacy and security properties

### 6.2.1 Delay and scalability

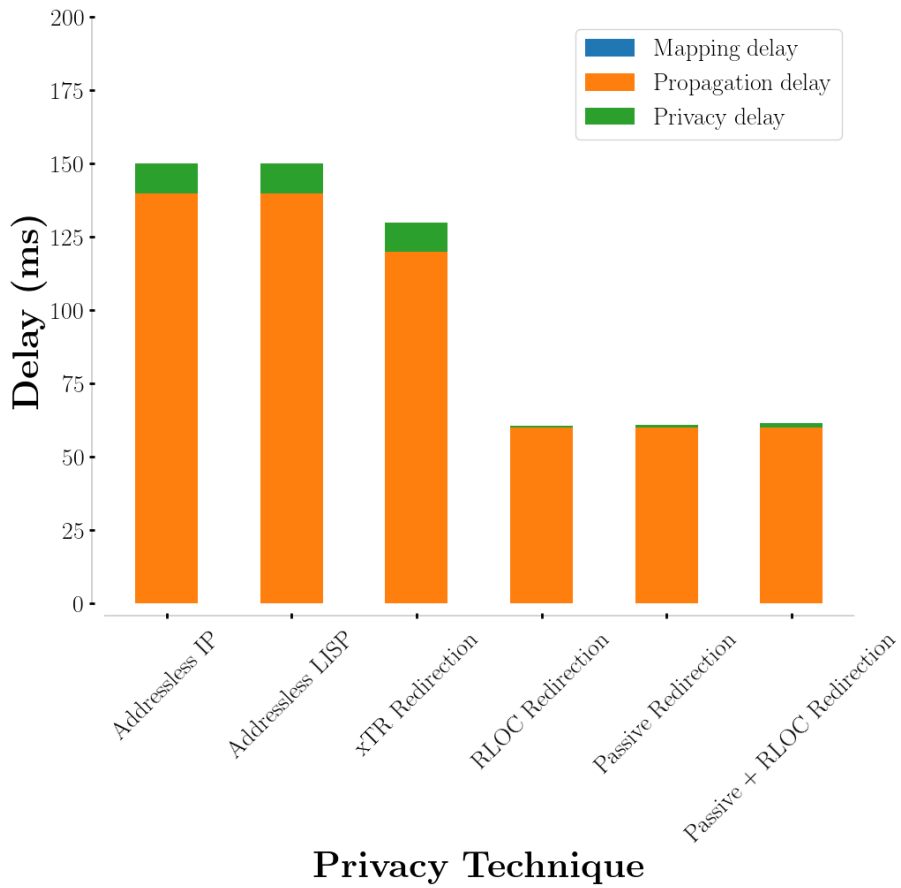


Figure 6.2: Delay perceived with xTR Caching

As expected, Figure 6.1 shows that the mapping resolution delay has a big impact on the connection delay experienced by a client, when there are no caching mechanisms in place or in case of cache miss. The first "naive" model described in this work performs far worse than the other models because of the three mapping resolution that the xTRs must go through to reach each element of the topology. The following figures demonstrate that this delay can be heavily mitigated by the mechanisms quoted in previous section.

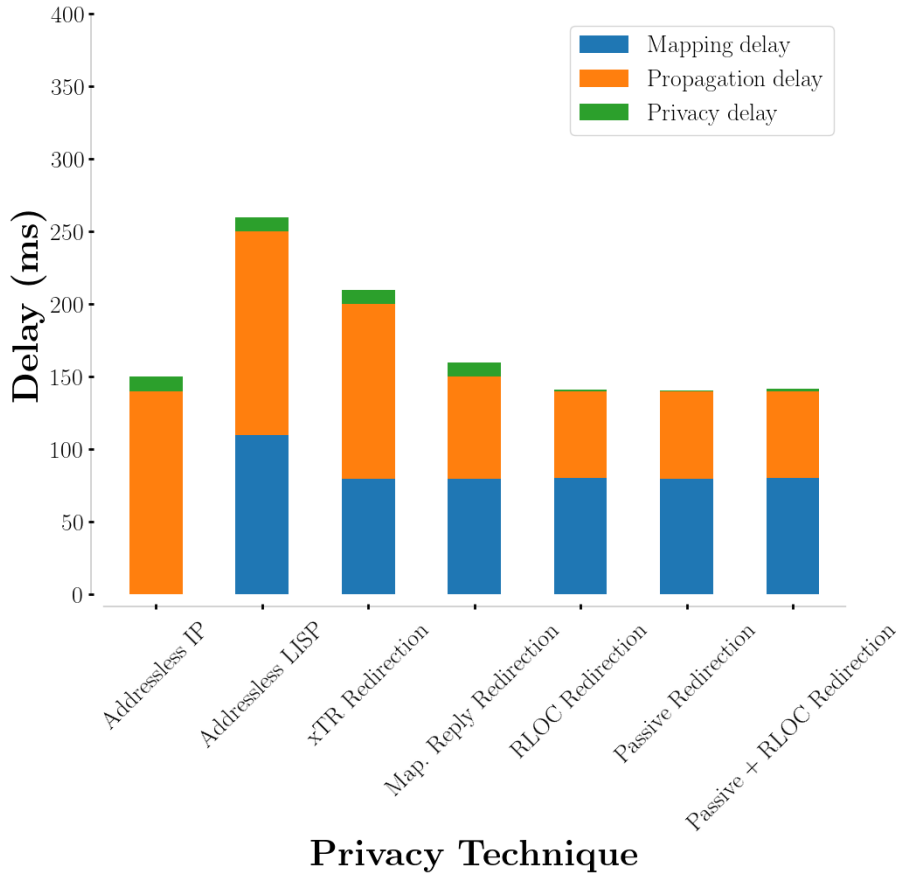


Figure 6.3: Delay perceived in case of Map-Resolver caching

After that first client, any clients behind the same xTR will experience a much smaller connection delay, as depicted in Figure 6.2, thanks to xTR caching. Analyzing the connection delay for these clients, we see that the passive solutions performs far better than the active ones, which is an obvious consequence of removing the need for client redirection and the additional connection establishment associated.

For Clients that are behind different xTRs, but with these xTRs using the same Map-Resolver, mapping delay will also be slightly reduced after the first request. In this case, for the active solution, sending a redirection request upon reception of a mapping request for the service EID is the best solution in terms of delay. Indeed, the redirection message, in the case of a TCP connection, arrives before the first SYN packet of a client can reach its destination IP. In the other active solutions, two complete TCP connections must be established for the client to be able to contact the server.

In conclusion, even in the worst case scenario, for the first request sent to the service, the solutions implemented using the LISP protocol do perform with only a small increase in perceived delay. If we do not consider the mapping delay, these models can achieve far better delays than the original IP model. It will thus depend on the ISP topology and positioning of xTRs, but as mapping resolution is the main source of delay for LISP networks, it can be expected that topologies would be built to take full advantage of the mechanisms that exist to reduce it.

The same observation can be made for scalability, with the notable exception of

the naive solution that suffers from clients synchronization due to the two mapping resolution that the clients' xTR must perform.

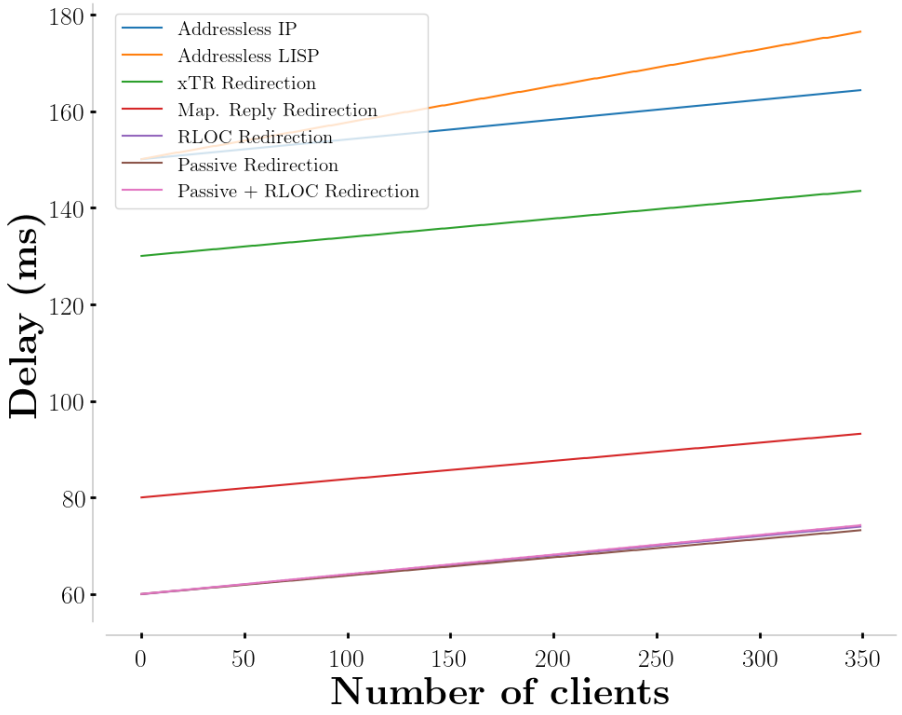


Figure 6.4: Evolution of the delay perceived by successive clients

### 6.2.2 Impact on throughput

As Hao et al. measured in their real-life implementation, the IP addressless model does not have an impact on throughput. This is confirmed by the measurements done with the ns-3 implementation for both the IP and LISP models. The experiment has been conducted once for each model, with links of 100mb/s and 5 ms of delay. One shortcoming of these measurements, however, is that it doesn't include any kind of timing model for the passive solution, that has to replace addresses of each packet coming to/from the server. Due to the lack of data about the impact of Nat devices on delay in the last few years, this delay is considered negligible.

| Solution | Throughput (mb/s) |
|---|---|
| *IP Addressless empiric* | 82.15 |
| *IP ns-3* | 90 |
| *IP Addressless ns-3* | 89.81 |
| *LISP models ns-3* | 88.24 |

Table 6.1: Measured throughput for each model

The two differences we can notice are the difference between real-life measurement and ns3 measurements, ns-3 values are optimistic due to both the unknown conditions of the real life measurements and the lacking of timing modeling of all and

every delay involved. The other noticeable difference is the difference in throughput between IP and LISP models, which is due to the LISP protocol overhead, both in packet size and xTR processing.

### 6.2.3 Implementation complexity and deployability

In terms of implementation complexity, the models are fairly comparable. Using the xTR to perform an application layer redirection would require a complete internet stack in the xTR, but nowadays, it is fairly common for routers to have support for higher layers. Moreover, this would only concern xTRs, which are located on the border of a LISP site. The passive solution requires processing that are very similar to NATs, but again, these are often co-located in routers.

Aside from the naive LISP solution, the LISP models are far easier to adopt since they do not require an additional device in the topology, that must be configured, maintained, and available at all time. The server does not need to be modified as address checking is performed by the xTR. Additionally, models that do not use application layer redirection works for all applications, which makes them very simple yet interesting to adopt.

### 6.2.4 Privacy and security properties

In Chapter 2, three types of privacy were defined: identity, location and complete privacy, which includes both. The addressless model provides both identity and location privacy, as it protects IP addresses which serve as both the locator and the identifier of a device in an IP network. As presented in Chapter 4, the LISP models outlined in this work can provide either identity or location privacy, or both, like the IP model, by combining the RLOC privacy solution with any of the other models. When both are provided, the client is only communicating with a service, rather than a server, and has no knowledge of the service provider's topology.

In terms of security, the active privacy models presented provide the same features, as they function similarly: the IP addresses used to reach the server are specific to a client's IP, only obtainable by going through an intermediate device and are only valid during a certain timeframe. The passive "redirection", however, is made at the network layer which could make it more vulnerable to network scanning. But, as we described in subsection 4.3.4 of Chapter 4, it is not the case thanks to mechanisms such as port checking in place to mitigate those risks. Besides reducing server identifiability, both active and passives models are not very well suited for mitigating server attacks and should be used in combination of other mechanisms to complement server's security. As said in the Addressless paper, the usage of counter-measures at the transport and application layer would be far more efficient at mitigating these vulnerabilities than both the IP addressless privacy model or the one LISP model presented here.

# Chapter 7

# Conclusion

In this work, we provided an overview of the LISP protocol, which was developed to address the scalability issues of the current IP architecture. By using two address spaces to decouple the identifier function from the locator function of an IP address, LISP addresses the growing RIB size in the DFZ. RLOCs are used to denote attachment points of networks and are allocated with respect to the topology. With their static nature, they lead to heavy aggregation in the DFZ. With these in place, EIDs can be more dynamic and allocated according to an organization's policies without impacting the Internet core. LISP introduces a facility for translating between these two spaces, the Mapping System, which can also be used to provide mechanisms for easily implementing both multihoming and Traffic Engineering.

## Privacy

As a network protocol, various aspects of its Data Plane and Control Plane are still being discussed, such as security. Work has been done to secure both the Data Plane and Control Plane, but little to none addresses devices' privacy. With the increasing amount of data exchanged in the current Internet architecture, privacy has become a critical concern for the many actors relying on its services.

This work has taken the first step towards providing anonymization to the end-devices involved in a LISP communication. Since LISP is composed of two address spaces, we divided privacy in two components: Identity and Location privacy. Building upon a model designed for server anonymization in IP networks, we presented a series of models using either passive or active redirection methods to reduce server identifiability as much as possible. All of these models have been implemented in the ns-3 network simulator, using and extending the current LISP implementation.

## Results

Interesting results were obtained through these simulations. Based on the IP model, using active redirection mechanisms, we obtained models for identity privacy that are much easier to deploy and maintain, as they do not require an additional device to perform the redirection. The delay due to the redirection method is also decreased

by taking advantage of the LISP mapping system. We showed that redirecting clients when receiving a Map Request for the server leads to the best results. We also developed a passive redirection mechanism that does not require redirecting the client from one IP to another. In addition to the significant decrease in delay, we showed that, even though the service could be reached using its public IP address, which was not the case for active redirection mechanisms, the identifiability of the servers was greatly decreased. Basic port-based rules were also added to the passive solution to limit its exposure to attacks. The key advantages of this passive model were a heavily decreased delay, ease of deployment, and the ability to be used for clients privacy with only a slight modification.

Finally, we introduced a very simple method for location privacy, relying on either the Mapping System's Map Server or the xTR to answer specific RLOCs requests. Even when combining this solution with the passive one to provide complete privacy, the results were still far better than the other models while maintaining privacy for both servers and clients.

This work showed that it is possible to achieve privacy at a lower cost than what was designed for the IP protocol. Clients experience better delays, the models are easier to deploy and maintain, and they can be easily deployed on the client side. LISP models do suffer, like any models that use LISP, from LISP mapping delays, but proper caching mechanisms can limit this delay so that only the first request to a service experiences it.

# Discussion and future works

These results, however, can only be compared with each other and should not be considered an approximation of the delay a client would experience in a real-life implementation. Many simplifications have been made, to the mapping system and the timing models, due to a lack of available data. These results should be considered as a comparison point between models and not as an indication of the feasibility of these models in a real-life settings. Simulations remain simulations, and it would require more data and work to establish an approximation of real life results.

In this work, we mainly explored solutions built on top of the LISP protocol. These methods have the advantage of being easier to deploy on top of an existing architecture than built-in solutions, but they do nevertheless add to the overall complexity of the existing system. They also tend to be limited by the existing mechanisms. For a protocol that has not yet been deployed, such as LISP, it would be interesting to consider and study the advantages that could be bought by built-in solutions, in the same way that has been done for data plane confidentiality.

In conclusion, this work explored the potential of using LISP to provide anonymization to end devices in a communication. By implementing and comparing various models in a simulation environment, we demonstrated the feasibility of using LISP for this purpose. The LISP protocol presents a promising solution for addressing the scalability issues of the current Internet architecture while also providing the opportunity for enhanced security and privacy for end devices.

# Appendix A

# Chapter 1: Additional figures

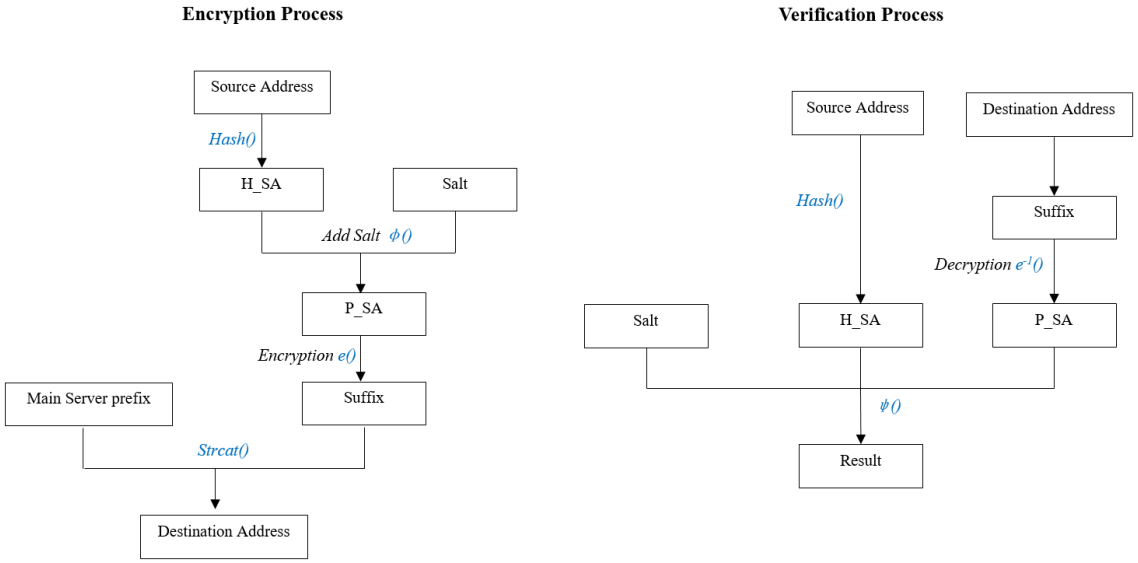


Figure A.1: Address generation and verification process [12]

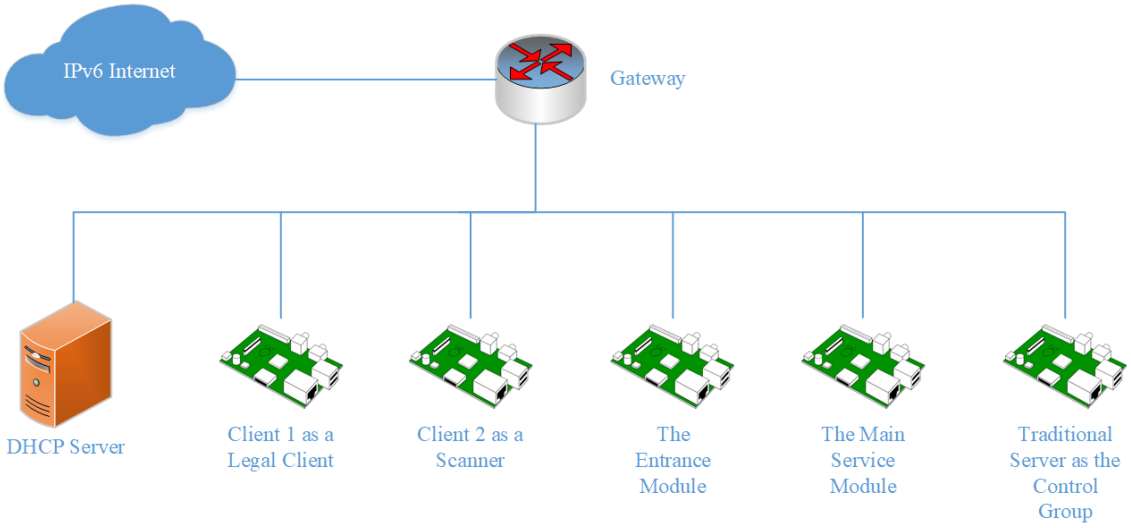


Figure A.2: Topology used for empiric measurements of performance [12]

# Appendix B

# Chapter 2: Additional figures

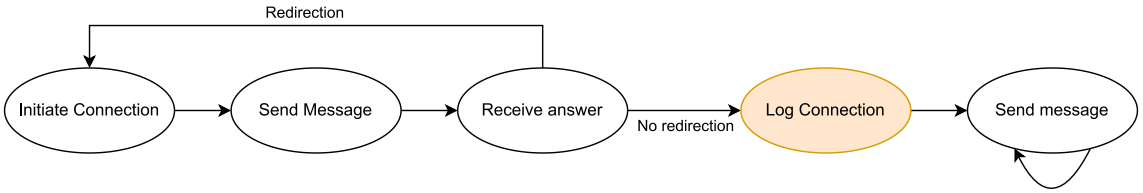


Figure B.1: Client state machine

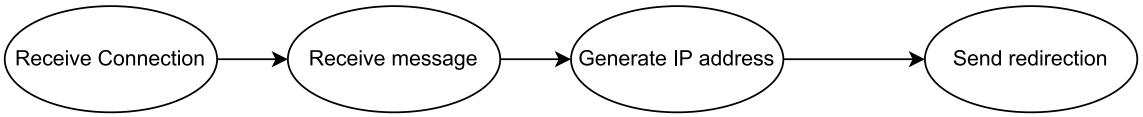


Figure B.2: Entrance module state machine

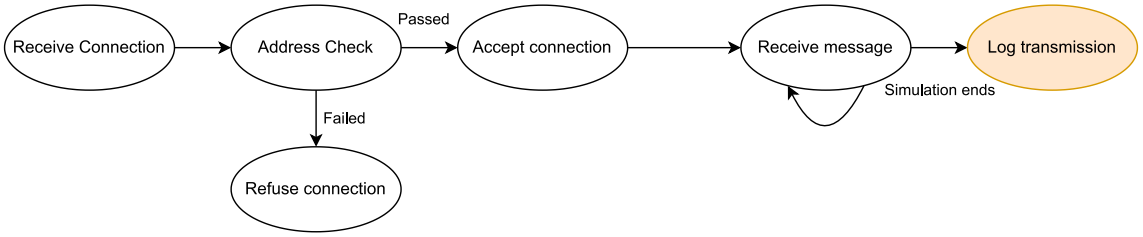


Figure B.3: Server state machine

# Appendix C

# Chapter 5: Additional figures


Figure C.1: Map-Request process

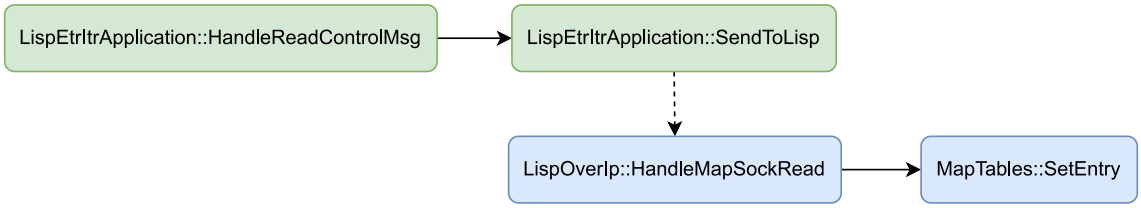


Figure C.2: Map-Reply process

# Bibliography


[1] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, and A. Cabellos-Aparicio, "The Locator/ID Separation Protocol (LISP)." RFC 9300, Oct. 2022.

[2] L. Zhang, K. Fall, and D. Meyer, "Report from the IAB Workshop on Routing and Addressing." RFC 4984, Sept. 2007.

[3] D. Farinacci, F. Maino, V. Fuller, and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control Plane." RFC 9301, Oct. 2022.

[4] F. Maino, V. Ermagan, A. Cabellos-Aparicio, and D. Saucez, "Locator/ID Separation Protocol Security (LISP-SEC)." RFC 9303, Oct. 2022.

[5] S. Farrell and H. Tschofenig, "Pervasive Monitoring Is an Attack." RFC 7258, May 2014.

[6] A. Cooper, H. Tschofenig, D. B. D. Aboba, J. Peterson, J. Morris, M. Hansen, and R. Smith, "Privacy Considerations for Internet Protocols." RFC 6973, July 2013.

[7] D. S. E. Deering and B. Hinden, "Internet Protocol, Version 6 (IPv6) Specification." RFC 8200, July 2017.

[8] T. Fiebig, K. Borgolte, S. Hao, C. Kruegel, and G. Vigna, "Something from nothing (there): Collecting global ipv6 datasets from dns," in *Proceedings of Passive and Active Measurement Conference 2018* (N. Spring and G. Riley, eds.), Lecture Notes in Computer Science, pp. 1–12, Springer Science, 2017. International Conference on Passive and Active Network Measurement, PAM 2018 ; Conference date: 26-03-2018 Through 27-03-2018.

[9] G. Song, L. He, Z. Wang, J. Yang, T. Jin, J. Liu, and G. Li, "Towards the construction of global ipv6 hitlist and efficient probing of ipv6 address space," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, 2020.

[10] Z. Liu, Y. Xiong, X. Liu, W. Xie, and P. Zhu, "6tree: Efficient dynamic discovery of active addresses in the ipv6 address space," *Computer Networks*, vol. 155, pp. 31–46, 2019.

[11] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson, "Target generation for internet-wide ipv6 scanning," in *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, (New York, NY, USA), p. 242–253, Association for Computing Machinery, 2017.

[12] S. Hao, R. Liu, Z. Weng, D. Chang, C. Bao, and X. Li, "Addressless: A new internet server model to prevent network scanning," *PLOS ONE*, vol. 16, no. 2, p. e0246293, 2021. Number: 2.

[13] "Ns-3, a discrete-event network simulator for internet systems.." `https://www.nsnam.org/`. Accessed: 2023-01-06.

[14] E. Marechal, "Master thesis : Simulating LISP with NS3," Master's thesis, Université de Liège, Liège, Belgique, 2019.

[15] E. Commission, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)," *OJ*, 2016-04-27.

[16] "Nmap scanning tool.." `https://nmap.org/`. Accessed: 2022-09-30.

[17] "Zmap scanning tool." `https://zmap.io/`. Accessed: 2022-09-30.

[18] L. I. Antoine Fressancourt, "Over-the-top or built-in approaches to improve privacy at the network layer," 2021. IETF meeting 112 ; Conference date: 06-11-2021 Through 12-11-2021.

[19] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3." RFC 8446, Aug. 2018.

[20] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap." RFC 6071, Feb. 2011.

[21] M. Holdrege and P. Srisuresh, "IP Network Address Translator (NAT) Terminology and Considerations." RFC 2663, Aug. 1999.

[22] D. T. Narten, T. Jinmei, and D. S. Thomson, "IPv6 Stateless Address Autoconfiguration." RFC 4862, Sept. 2007.

[23] T. Mrugalski, M. Siodelski, B. Volz, A. Yourtchenko, M. Richardson, S. Jiang, T. Lemon, and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)." RFC 8415, Nov. 2018.

[24] L. Zhang, D. Thaler, and G. M. Lebovitz, "IAB Thoughts on IPv6 Network Address Translation." RFC 5902, July 2010.

[25] F. Gont, S. Krishnan, D. T. Narten, and R. P. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6." RFC 8981, Feb. 2021.

[26] N. Hakiem, A. Priantoro, M. U. Siddiqi, and T. Hashim, "Generation of cryptographic one-to-many mapping IPv6 address using S-AES," Pages: E18.

[27] N. Hakiem and M. U. Siddiqi, "One-to-many reversible mapping for IPv6 address generation: simulation software development," vol. 47, p. 10.

[28] F. Gont, "A method for generating semantically opaque interface identifiers with IPv6 stateless address autoconfiguration (SLAAC)." Issue: RFC 7217 Num Pages: 19.

[29] S. Han, V. Liu, Q. Pu, S. Peter, T. Anderson, A. Krishnamurthy, and D. Wetherall, "Expressive privacy control with pseudonyms," p. 12.

[30] T. Aura, "Cryptographically Generated Addresses (CGA)." RFC 3972, Mar. 2005.

[31] G. Huston, "BGP in 2021 - The BGP Table," 2022. APNIC annual report.

[32] Y. Rekhter, S. Hares, and T. Li, "A Border Gateway Protocol 4 (BGP-4)." RFC 4271, Jan. 2006.

[33] B. Hinden, "New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG." RFC 1955, June 1996.

[34] "LISP Beta-Network." `https://lisp4.net.cba.upc.edu/beta-network/`. Accessed: 2022-09-29.

[35] T. S. Vinit Jain, Yves Louis, "Lisp architecture," 2020. Cisco Press.

[36] L. Yue, "Future internet services based on LIPS technology," 2018.

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

[38] V. Fuller, D. Lewis, V. Ermagan, A. Jain, and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)." RFC 8111, May 2017.

[39] D. Saucez, L. Iannone, and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Threat Analysis." RFC 7835, Apr. 2016.

[40] D. Farinacci and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality." RFC 8061, Feb. 2017.

[41] A. Rodriguez-Natal, V. Ermagan, A. Cabellos-Aparicio, S. Barkai, and M. Boucadair, "Publish/Subscribe Functionality for LISP," Internet-Draft draft-ietf-lisp-pubsub-09, Internet Engineering Task Force, June 2021. Work in Progress.

[42] T. Henderson, "Ns3 training, session 1." Ns3 annual meeting 2017, 2017.

[43] L. Agbodjan, "Towards a lisp simulator," Master's thesis, Université de Liège, Liège, Belgique, 2016.

[44] D. Farinacci, D. Lewis, D. Meyer, and C. White, "LISP Mobile Node," Internet-Draft draft-ietf-lisp-mn-12, Internet Engineering Task Force, July 2022. Work in Progress.

[45] T. Piron, "Master thesis : Implementation and evaluation of LISP publish/-subscribe functionality in NS3," Master's thesis, Université de Liège, Liège, Belgique, 2022.

[46] V. Ermagan, D. Farinacci, D. Lewis, F. Maino, M. Portoles-Comeras, J. Skriver, C. White, A. L. Brescó, and A. Cabellos-Aparicio, "NAT traversal for LISP," Internet-Draft draft-ermagan-lisp-nat-traversal-19, Internet Engineering Task Force, May 2021. Work in Progress.

[47] A. Rodriguez-Natal, V. Ermagan, A. Cabellos-Aparicio, S. Barkai, and M. Boucadair, "Publish/Subscribe Functionality for LISP," Internet-Draft draft-ietf-lisp-pubsub-09, Internet Engineering Task Force, June 2021. Work in Progress.