

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 17, 2015

A. Cabellos-Aparicio (Ed.)  
Technical University of Catalonia  
D. Saucez (Ed.)  
INRIA  
July 16, 2014

**An Architectural Introduction to the LISP Location-Identity  
Separation System  
draft-ietf-lisp-introduction-04.txt**

**Abstract**

This document is an introductory overview of the Locator/ID Separation Protocol, it describes the major concepts and functional sub-systems of LISP and the interactions between them.

**Status of This Memo**

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2015.

**Copyright Notice**

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Prefatory Note</a>	<a href="#">4</a>
<a href="#">2.</a>	<a href="#">Part I</a>	<a href="#">4</a>
<a href="#">3.</a>	<a href="#">Initial Glossary</a>	<a href="#">5</a>
<a href="#">4.</a>	<a href="#">Background</a>	<a href="#">6</a>
<a href="#">5.</a>	<a href="#">Deployment Philosophy</a>	<a href="#">7</a>
<a href="#">5.1.</a>	<a href="#">Economics</a>	<a href="#">7</a>
<a href="#">5.2.</a>	<a href="#">Maximize Re-use of Existing Mechanism</a>	<a href="#">8</a>
<a href="#">6.</a>	<a href="#">LISP Overview</a>	<a href="#">8</a>
<a href="#">6.1.</a>	<a href="#">Basic Approach</a>	<a href="#">9</a>
<a href="#">6.2.</a>	<a href="#">Basic Functionality</a>	<a href="#">9</a>
<a href="#">6.3.</a>	<a href="#">Mapping from EIDs to RLOCs</a>	<a href="#">11</a>
<a href="#">6.4.</a>	<a href="#">Interworking With Non-LISP-Capable Endpoints</a>	<a href="#">11</a>
<a href="#">6.5.</a>	<a href="#">Security in LISP</a>	<a href="#">12</a>
<a href="#">7.</a>	<a href="#">Initial Applications</a>	<a href="#">13</a>
<a href="#">7.1.</a>	<a href="#">Provider Independence</a>	<a href="#">13</a>
<a href="#">7.2.</a>	<a href="#">Multi-Homing</a>	<a href="#">13</a>
<a href="#">7.3.</a>	<a href="#">Traffic Engineering</a>	<a href="#">14</a>
<a href="#">7.4.</a>	<a href="#">Routing</a>	<a href="#">15</a>
<a href="#">7.5.</a>	<a href="#">Mobility</a>	<a href="#">15</a>
<a href="#">7.6.</a>	<a href="#">Traversal Across Alternate IP Versions</a>	<a href="#">15</a>
<a href="#">7.7.</a>	<a href="#">Virtual Private Networks</a>	<a href="#">16</a>
<a href="#">7.8.</a>	<a href="#">Local Uses</a>	<a href="#">16</a>
<a href="#">8.</a>	<a href="#">Major Functional Subsystems</a>	<a href="#">17</a>
<a href="#">8.1.</a>	<a href="#">Data Plane - xTRs Overview</a>	<a href="#">17</a>
<a href="#">8.1.1.</a>	<a href="#">Mapping Cache Performance</a>	<a href="#">18</a>
<a href="#">8.2.</a>	<a href="#">Control Plane - Mapping System Overview</a>	<a href="#">18</a>
<a href="#">8.2.1.</a>	<a href="#">Mapping System Organization</a>	<a href="#">19</a>
<a href="#">8.2.2.</a>	<a href="#">Interface to the Mapping System</a>	<a href="#">20</a>
<a href="#">8.2.3.</a>	<a href="#">Indexing Sub-system</a>	<a href="#">20</a>
<a href="#">9.</a>	<a href="#">Examples of operation</a>	<a href="#">22</a>
<a href="#">9.1.</a>	<a href="#">An Ordinary Packet's Processing</a>	<a href="#">22</a>
<a href="#">9.2.</a>	<a href="#">A Mapping Cache Miss</a>	<a href="#">23</a>
<a href="#">10.</a>	<a href="#">Part II</a>	<a href="#">24</a>
<a href="#">11.</a>	<a href="#">Design Approach</a>	<a href="#">24</a>
<a href="#">12.</a>	<a href="#">xTRs</a>	<a href="#">25</a>
<a href="#">12.1.</a>	<a href="#">When to Encapsulate</a>	<a href="#">25</a>
<a href="#">12.2.</a>	<a href="#">UDP Encapsulation Details</a>	<a href="#">26</a>
<a href="#">12.3.</a>	<a href="#">Header Control Channel</a>	<a href="#">26</a>
<a href="#">12.3.1.</a>	<a href="#">Mapping Versioning</a>	<a href="#">26</a>
<a href="#">12.3.2.</a>	<a href="#">Echo Nonces</a>	<a href="#">27</a>
<a href="#">12.3.3.</a>	<a href="#">Instances</a>	<a href="#">27</a>
<a href="#">12.4.</a>	<a href="#">Probing</a>	<a href="#">28</a>
<a href="#">12.5.</a>	<a href="#">Mapping Lifetimes and Timeouts</a>	<a href="#">28</a>
<a href="#">12.6.</a>	<a href="#">Mapping Gleaning in ETRs</a>	<a href="#">29</a>
<a href="#">12.7.</a>	<a href="#">MTU Issues</a>	<a href="#">29</a>
<a href="#">12.8.</a>	<a href="#">Security of Mapping Lookups</a>	<a href="#">29</a>



<a href="#">12.9.</a>	<a href="#">ITR Mapping Cache Performance</a>	<a href="#">30</a>
<a href="#">13.</a>	<a href="#">The Mapping System</a>	<a href="#">31</a>
<a href="#">13.1.</a>	<a href="#">The Mapping System Interface</a>	<a href="#">32</a>
<a href="#">13.1.1.</a>	<a href="#">Map-Request Messages</a>	<a href="#">32</a>
<a href="#">13.1.2.</a>	<a href="#">Map-Reply Messages</a>	<a href="#">32</a>
<a href="#">13.1.3.</a>	<a href="#">Map-Register and Map-Notify Messages</a>	<a href="#">33</a>
<a href="#">13.2.</a>	<a href="#">The DDT Indexing Sub-system</a>	<a href="#">34</a>
<a href="#">13.3.</a>	<a href="#">Reliability via Replication</a>	<a href="#">35</a>
<a href="#">13.4.</a>	<a href="#">Security of the DDT Indexing Sub-system</a>	<a href="#">35</a>
<a href="#">13.5.</a>	<a href="#">Extended Capabilities</a>	<a href="#">36</a>
<a href="#">13.6.</a>	<a href="#">Performance of the Mapping System</a>	<a href="#">36</a>
<a href="#">14.</a>	<a href="#">Multicast Support in LISP</a>	<a href="#">37</a>
<a href="#">14.1.</a>	<a href="#">Basic Concepts of Multicast Support in LISP</a>	<a href="#">37</a>
<a href="#">14.2.</a>	<a href="#">Initial Multicast Support in LISP</a>	<a href="#">38</a>
<a href="#">15.</a>	<a href="#">Deployment Issues and Mechanisms</a>	<a href="#">39</a>
<a href="#">15.1.</a>	<a href="#">LISP Deployment Needs</a>	<a href="#">39</a>
<a href="#">15.2.</a>	<a href="#">Interworking Mechanisms</a>	<a href="#">40</a>
<a href="#">15.2.1.</a>	<a href="#">Proxy LISP Routers</a>	<a href="#">40</a>
<a href="#">15.2.2.</a>	<a href="#">LISP-NAT</a>	<a href="#">42</a>
<a href="#">15.3.</a>	<a href="#">Use Through NAT Devices</a>	<a href="#">42</a>
<a href="#">15.4.</a>	<a href="#">LISP and Core Internet Routing</a>	<a href="#">43</a>
<a href="#">16.</a>	<a href="#">Fault Discovery/Handling</a>	<a href="#">43</a>
<a href="#">16.1.</a>	<a href="#">Handling Missing Mappings</a>	<a href="#">44</a>
<a href="#">16.2.</a>	<a href="#">Outdated Mappings</a>	<a href="#">44</a>
<a href="#">16.2.1.</a>	<a href="#">Outdated Mappings - Updated Mapping</a>	<a href="#">44</a>
<a href="#">16.2.2.</a>	<a href="#">Outdated Mappings - Wrong ETR</a>	<a href="#">44</a>
<a href="#">16.2.3.</a>	<a href="#">Outdated Mappings - No Longer an ETR</a>	<a href="#">45</a>
<a href="#">16.3.</a>	<a href="#">Erroneous Mappings</a>	<a href="#">45</a>
<a href="#">16.4.</a>	<a href="#">Verifying ETR Liveness</a>	<a href="#">45</a>
<a href="#">16.5.</a>	<a href="#">Verifying ETR Reachability</a>	<a href="#">46</a>
<a href="#">17.</a>	<a href="#">Acknowledgments</a>	<a href="#">46</a>
<a href="#">18.</a>	<a href="#">IANA Considerations</a>	<a href="#">47</a>
<a href="#">19.</a>	<a href="#">Security Considerations</a>	<a href="#">47</a>
<a href="#">20.</a>	<a href="#">References</a>	<a href="#">47</a>
<a href="#">20.1.</a>	<a href="#">Normative References</a>	<a href="#">47</a>
<a href="#">20.2.</a>	<a href="#">Informative References</a>	<a href="#">49</a>
<a href="#">Appendix A.</a>	<a href="#">Glossary/Definition of Terms</a>	<a href="#">52</a>
<a href="#">Appendix B.</a>	<a href="#">Other Appendices</a>	<a href="#">55</a>
<a href="#">B.1.</a>	<a href="#">A Brief History of Location/Identity Separation</a>	<a href="#">55</a>
<a href="#">B.2.</a>	<a href="#">A Brief History of the LISP Project</a>	<a href="#">56</a>
<a href="#">B.3.</a>	<a href="#">Old LISP 'Models'</a>	<a href="#">56</a>
<a href="#">B.4.</a>	<a href="#">The ALT Mapping Indexing Sub-system</a>	<a href="#">57</a>
<a href="#">B.5.</a>	<a href="#">Early NAT Support</a>	<a href="#">58</a>
	<a href="#">Authors' Addresses</a>	<a href="#">59</a>



## **1. Prefatory Note**

{{Suggestion by editors: Remove all the sections before "LISP Overview" and write a straight-to-the-point introduction}}

{{Suggestion by editors: The draft should focus on describing the LISP architecture and avoid comparing loc/id split architectures with other approaches}}

This document is the first of a pair which, together, form what one would think of as the 'architecture document' for LISP (the 'Location-Identity Separation Protocol'). Much of what would normally be in an architecture document (e.g. the architectural design principles used in LISP, and the design considerations behind various components and aspects of the LISP system) is in the second document, the 'Architectural Perspective on LISP' document. [[I-D.ietf-lisp-perspective](#)]

This 'Architectural Introduction' document is primarily intended for those who are unfamiliar with LISP, and want to start learning about it. It is intended primarily for those working on LISP, but those working with LISP, and more generally anyone who wants to know more about LISP, may also find this document useful.

This document is intended to both be easy to follow and it is structured as a series of phases, each covering the entire system, but with ever-increasing detail. Reading only the first part of the document will give a good high-level view of the system; reading the complete document should provide a fairly detailed understanding of the entire system.

People who just want to get an idea of how LISP works might only read the first part; they can stop reading either just before, or just after [Section 9](#). People who are going to go on and read the protocol specifications (perhaps to implement LISP) should read the entire document.

This document is a descriptive document, not a protocol specification. Should it differ in any detail from any of the LISP protocol specification documents, they take precedence for the actual operation of the protocol.

## **2. Part I**



### 3. Initial Glossary

This initial glossary defines a few general terms which will be useful to have in hand when commencing reading this document. A complete glossary is available in [Appendix A](#).

A note about style: initial usage of a term defined in the glossary is denoted with double quotation marks ("). Other uses of quotations (e.g. for quotations, euphemisms, etc) use single quotation marks (').

- o Name: a name refers to an identifier for an object or entity. Names have both semantics (meaning) and syntax (form) [[RFC1498](#)].
- o Namespace: A group of names with matching semantics and syntax; they usually, but not always, refer to members of a class of identical objects.
- o Mapping: a binding between two names, one in each of two namespaces.
- o Delegation Hierarchy: an abstract rooted tree (in the graph theory sense of the term) which is a virtual representation of the delegation of a namespace into smaller and smaller blocks, in a recursive process.
- o Node: The general term used to describe any sort of communicating entity; it might be a physical or a virtual host, or a mobile device of some sort. It includes both entities which forward packets, and entities which create or consume packets.
- o Switch, Packet Switch: A packet switch, in the general meaning of that term. A device which takes in packets from its interfaces and forwards them on, either to a next-hop switch, or to the final destination. They may operate at either the network layer (e.g. ARPANET), or internetwork layer. [[Baran](#)][Heart][[RFC1812](#)]
- o Endpoint, end-end communication entity: The fate-sharing region at one end of an end-end communication; the collection of state related to both the reliable end-end communication channel, and the applications running there. [[Chiappa](#)]
- o Address: In this document, in current IP (IPv4 or IPv6) and similar networking suites, a "name" which has mixed semantics, in that it includes both identity ('who') and location ('where') semantics. [[Atkinson](#)]





- o Address Block, Block: A contiguous section of a namespace (e.g., IP addresses that belong to the same prefix).
- o Identifier: a name which has identity semantics.
- o Locator: name with only location semantics and not necessarily carried in every packet [[RFC1992](#)].
- o Site: A collection of hosts, routers, and networks under a single administrative control.
- o LISP site: a site separated from the rest of the network by LISP routers.
- o LISP node: A network element implementing LISP functionality; this means it can process some subset of LISP control and planes traffic.
- o LISP router: A network element implementing LISP data-plane functionality, i.e., a LISP node which can forward user traffic.
- o LISP host: A host which is behind (from the point of view of the rest of the network) a LISP router.

#### **4. Background**

It has gradually been realized in the networking community that networks, especially large networks, should deal quite separately with the identity and location of an endpoint - basically, who an endpoint is, and where it is ([[RFC1498](#)] [[RFC4984](#)]).

At the moment of this writing, in both IPv4 and IPv6, IP addresses indicate both where the named node is, as well as identify it for purposes of end-end communication; i.e. it has both location and identity properties. However, the separation of those two properties is a step which has been identified by the IRTF as a necessary evolutionary architectural step for the Internet [[RFC6115](#)] and the on-going LISP project is an attempt to provide a viable path towards this separation.

As an add-on to a large existing system, it has had to make certain compromises. (For a good example, see [[I-D.ietf-lisp-perspective](#)], Section "Residual Location Functionality in EIDs". However, if it reaches near-ubiquitous deployment, it will have two important consequences.

First, in effectively providing separation of location and identity, along with providing a distributed directory of the mappings between



them, 'Wheeler's Law' ('All problems in computer science can be solved by another level of indirection') will come into play, and the Internet technical community will have a new, immensely powerful, tool at its disposal. The fact that the namespaces on both sides of the mapping are global ones maximizes the power of that tool. (See [[I-D.ietf-lisp-perspective](#)], Section "Need for a Mapping System", for more on this.)

Second, because of a combination of the flexible capability built into LISP, and the breaking of the unification of location and identity names, further architectural evolution of the Internet becomes easily available; for example, new namespaces for location or identity could be designed and deployed. In other words, LISP is not a point solution to meet a particular need, but hopefully an 'escape hatch' which will allow further significant enhancement to the Internet's overall architecture. (See [[Future](#)] for more on this.)

## 5. Deployment Philosophy

{{Suggestion by editors: Remove the entire section, it can be summarized by the last paragraph. Furthermore the arguments used in this sections are hard to follow since LISP has not been described yet.}}

The deployment philosophy was a major driver for the design of LISP (architecture and engineering).

Experience over the last several decades has shown that having a viable deployment model for a new design is a key to the adoption of the solution. In general, it is comparatively easy to conceive of new network designs, but much harder to devise approaches which will actually get deployed throughout the global network. A new design may be fantastic but if it can not be successfully deployed (for whatever factors), it is useless.

LISP aims to achieve the near-ubiquitous deployment necessary for maximum exploitation of an architectural upgrade by i) minimizing the amount of change needed (most existing hosts and routers can operate unmodified); and ii) by providing significant benefits to early adopters.

### 5.1. Economics

{{Suggestion by editors: Remove this section, this has not been discussed by the WG}}

A key factor in successful adoption is economics: does the new design have benefits which outweigh its costs?



More importantly, this balance needs to hold for early adopters - because if they do not receive benefits to their adoption, the sphere of earliest adopters will not expand, and it will never get to widespread deployment.

This is particularly true of architectural enhancements, which are far less likely to be an addition which one can bolt onto the side of existing mechanisms, and often offer their greatest benefits only when widely (or ubiquitously) deployed.

Maximizing the cost-benefit ratio obviously has two aspects. First, on the cost side, by making the design as inexpensive as possible, which means in part making the deployment as easy as possible. Second, on the benefit side, by providing many new capabilities, which is best done not by loading the design up with lots of features or options (which adds complexity), but by making the addition powerful through deeper flexibility. The LISP community believes LISP has met both of these goals.

## **5.2. Maximize Re-use of Existing Mechanism**

{{Suggestion by editors: Remove/Summarize the entire section, the arguments used in this section are hard to follow since LISP has not been described yet.}}

One key part of reducing the cost of a new design is to minimize the amount of change required to existing, deployed, devices: the fewer devices need to be changed, and the smaller the change to those that do, the greater the likelihood of deployment.

Designs which absolutely require forklift upgrades to large amounts of existing gear are far less likely to succeed - because they have to have extremely large benefits to make their very substantial costs worthwhile.

It is for this reason that LISP, in most cases, initially requires no changes to almost all existing devices in the Internet (both hosts and routers); LISP functionality needs to be added in only a few places ( [Section 15.1](#))

## **6. LISP Overview**

LISP is an incrementally deployable architectural upgrade to the existing Internet infrastructure, one which provides separation of location and identity. It thus starts to separate the names used for identity and location of nodes, which are currently unified in IP addresses.



### **6.1. Basic Approach**

{{Suggestion by editors: Merge this section with the previous one (LISP Overview)}}

In LISP, the first key concept is that nodes have both an identifier called an Endpoint IDentifier (EID) and an associated Routing Locator (RLOC). A node may be associated with more than one RLOC, or the RLOC may change over time (e.g., if the node is mobile), but it would normally always have the same EID.

The second key concept is that if one wants to be as forward-looking as possible, conceptually one should think of the two kinds of names (EIDs and RLOCs) as naming different classes of entities.

On the one hand, EIDs are used to name nodes - or rather, their end-end communication entities. RLOC(s), on the other hand, name interfaces, i.e. places to which the system of routers sends packets.

This distinction, the formal recognition of different kinds of entities ("endpoints" and interfaces), and their association with the two different classes of names, is also important. Clearly recognizing interfaces and endpoints as distinctly separate classes of objects is another improvement to the existing Internet architecture.

An important insight in LISP is that it initially uses existing IP addresses for both of these kinds of names, as opposed to some similar earlier deployment proposals for separation of location and identity (e.g. [[RFC1992](#)]), which proposed using a new namespace for locators. This choice minimizes LISP's deployment cost, as well as providing the ability to easily interact with un-modified hosts and routers.

The capability to use namespaces other than IP addresses for both kinds of names is already built in LISP, which is expected to greatly increase the long-term benefits, flexibility, and power of LISP ([[AFI](#)], [[I-D.ietf-lisp-lcaf](#)]).

### **6.2. Basic Functionality**

{{Suggestion by editors: Rewrite this section: It is using non-standard terminology to refer to standard LISP concepts ('enhance' instead of encapsulate) It is using subjective terms (e.g., 'near' the source) It is missing key LISP aspects such as that RLOC packets are forwarded in the RLOC space }}





The basic operation of LISP, as it currently stands, is quite simple. LISP augmented packet switches, "LISP routers", near the source and destination of packets intercept traffic, and 'enhance' the packets for the trip between the LISP switches.

The LISP router near the original source (the Ingress Tunnel Router, or ITR ) looks up additional information about the destination of the packet, and then wraps the packet in an outer header, one which contains additional information related to LISP operation.

The LISP router near the destination, the (the Egress Tunnel Router, or ETR) removes that header, leaving the original, un-modified, packet to be sent on to the original destination node.

The overall processing is shown below, in Figure 1:

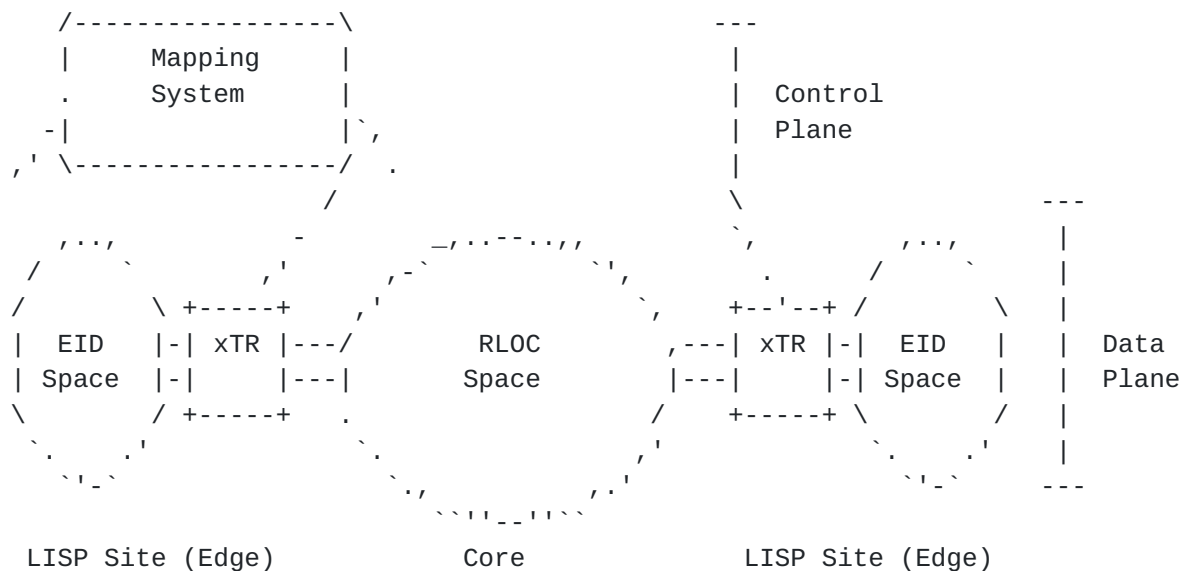


Figure.- An schema of the LISP Architecture

Figure 1: Basic LISP Packet Flow

To retrieve that additional information, the ITR uses the information in the original packet about the identity of its ultimate destination, i.e. the destination EID address. It uses the destination EID to look up the current location (the RLOC) of that EID.

The lookup is performed through a mapping system, which is the heart of LISP: it is a distributed directory of mappings from EIDs to



RLOCs. The destination RLOC(s) is (are) normally the address(es) of the ETR(s) near the ultimate destination.

The ITR then generates a new outer header for the original packet, with that header containing the ETR's RLOC as the wrapped packet's destination, and the ITR's own address (i.e. the RLOC usually associated with the original source) as the wrapped packet's source, and sends it off.

When the packet arrives at the ETR, that outer header is stripped off, and the original packet is forwarded to the original ultimate destination for normal processing.

Return traffic is handled similarly, often (depending on the network's configuration) with the original ITR and ETR switching roles. The ETR and ITR functionality is usually co-located in a single LISP router; these are normally denominated as xTRs.

### **6.3. Mapping from EIDs to RLOCs**

The mappings from EIDs to RLOCs are provided by a distributed, and potentially replicated, database, the "mapping database", which is the heart of LISP. Here, and in other places in LISP, the replication is not a deep architectural concept, simply an engineering device to obtain reliability via potential redundancy.

Entities which need EID-to-RLOC mappings get them from the mapping system, which is a collection of sub-systems through which clients can find and obtain mappings as discussed in more details in [Section 8.2](#) and [Section 13](#).

Mappings are normally distributed via a pull mechanism (i.e., not pre-loaded, but requested on demand). Once obtained by an ITR, they are cached for performance reasons.

### **6.4. Interworking With Non-LISP-Capable Endpoints**

It is clearly crucial to provide the capability for easy interoperation between "LISP hosts" - i.e. they are behind xTRs, and their EIDs are in the mapping database - and existing non-LISP-using hosts (often called 'legacy' hosts) or legacy "sites".

To allow interoperation between devices hosted in a LISP site and devices not belonging to a LISP site (non-LISP site), an interworking mechanism based on proxies has been designed. Proxy ITRs (PITR) encapsulate packets sent from non-LISP sites to LISP sites while Proxy ETRs (PETR) decapsulate packets sent from LISP sites to non-LISP sites. More details are given in [Section 15.2.1](#).



## 6.5. Security in LISP

{{Suggestion by editors: Rewrite this section: (first 3 paragraphs) It contains a general discussion as well as strong statements that are not supported by any WG document. (last 3 paragraphs) It enumerates the security mechanisms specified in LISP but does not describe them.}}

To provide a brief overview of security in LISP, it is definitely understood that LISP needs to be highly *\_securable\_*, especially in the long term; over time, the attacks mounted by 'bad guys' are becoming more and more sophisticated. So LISP, like DNS, needs to be *\_capable\_* of providing 'the very best' security there is.

At the same time, there is a conflicting goal: it must be deployable at a viable cost. That means two things: First, as an experiment, we cannot expect to create the complete security apparatus which we might see in the finished product, including both design and implementation. Second, security needs to be flexible, so that we don't overload the users with more security than they need at any point.

To accomplish these divergent goals, the approach taken is to first analyze what LISP needs for security. [[I-D.ietf-lisp-threats](#)]. Then, steps can be taken to ensure that the appropriate 'hooks' (such as packet fields) are included at an early stage, when doing so is still easy. Over time, additional mechanisms will be fully specified, implemented, and deployed.

LISP does already include a number of security mechanisms; in particular, requesting mappings can be secured (see [Section 12.8](#), "Security of Mapping Lookups"), as can registering of xTRs (see [Section 13.1.3](#), "Map-Register and Map-Notify Messages"); the key database of the mapping system is also secured (see [Section 13.4](#), "Security of the DDT Indexing Sub-system").

The existing security mechanisms, and their configuration (which is mostly manual at this point) currently in LISP are felt to be adequate for the needs of the on-going early stages of deployment; experience will indicate when improvements are required (within the constraints of the conflicting goal given above).

For more on LISP's security philosophy; see [[I-D.ietf-lisp-perspective](#)], Section 7 "Security", where it is laid out in some detail.



## **7. Initial Applications**

{{Suggested by editors: Move this section after [section 8](#), the applications should not be described before LISP has been fully described.

{{Suggested by Noel: Reorder the whole section in popularity order?}}

As previously mentioned, it is felt that LISP will provide even the earliest adopters with some useful capabilities, and that these capabilities will drive early LISP deployment.

It is very important to note that even when used only for interoperation with existing un-modified hosts, use of LISP can still provide benefits to the site which has deployed it - and, perhaps even more importantly, can do so to both sides. This characteristic acts to further enhance the utility for early adopters of LISP.

Note also that this section only lists some early applications and benefits. See [[I-D.ietf-lisp-perspective](#)], in the Section "Goals of LISP", for a more extensive discussion of some of what LISP might ultimately provide.

### **7.1. Provider Independence**

Provider independence (i.e. the ability to easily change one's Internet Service Provider) is a good example of the utility of separating location and identity.

To limit global routing table size addresses need to be aggregated. To that aim networks can use provider aggregatable addresses ([\[RFC4116\]](#)) which means that the IP prefixes of networks are sub-prefixes of their provider. This solution allows to reduce scalability issues of the global routing table but it means that when a network switches providers it has to re-number all its devices which is known to be complex in current networks [[RFC5887](#)].

Having separate namespaces for location and identity greatly reduces the problems involved with re-numbering; an organization which moves retains its EIDs (which are how most other parties refer to its nodes), but is allocated new RLOCs, and the mapping system can quickly provide the updated mapping from the EIDs to the new RLOCs.

### **7.2. Multi-Homing**

{{Suggested by editors: This section should be extended, it is unclear the benefits that LISP brings to multihoming (e.g, traffic engineering, decoupled multihoming from the data-plane).





Multi-homing is another place where the value of separation of location and identity became apparent. There are several different sub-flavours of the multi-homing problem - e.g. depending on whether one wants open TCP connections to keep working, etc - and other axes as well (e.g. site multi-homing versus host multi-homing).

In particular, for the 'keep open connections up' case, without separation of location and identity, with most currently deployed implementations, the only currently feasible approach is to use provider-independent addresses - which moves the problem into the global routing system, with attendant costs. This approach is also not really feasible for host multi-homing.

### **7.3. Traffic Engineering**

{{Suggestion by editors: Merge this section with the previous one, TE is not practical without multihoming}}

{{Needs a fix - not sure what.}}

Traffic engineering (TE) [[RFC3272](#)], desirable though this capability is in a global network, is currently somewhat problematic to provide in the Internet. The problem, fundamentally, is that this capability was not foreseen when the Internet was designed, so the support for it via hacks is neither clean, nor flexible.

TE is, fundamentally, a routing issue. However, the current Internet routing architecture, which is basically the Baran design of fifty years ago [[Baran](#)] (a single large, distributed computation), is ill-suited to provide TE. The Internet seems a long way from adopting a more-advanced routing architecture, although the basic concepts for such have been known for some time. [[RFC1992](#)]

Although the identity-location mapping layer is thus a poor place, architecturally, to provide TE capabilities, it is still an improvement over the current routing tools available for this purpose (e.g. injection of more-specific routes into the global routing table).

In addition, instead of the entire network incurring the costs (through the routing system overhead), when using a mapping layer to provide TE, the overhead is limited to those who are actually communicating with that particular destination.

LISP includes a number of features in the mapping system to support TE. (described in [Section 8.2](#), "Control Plane - Mapping System Overview", below); more details about using LISP for TE can be found in [[I-D.farinacci-lisp-te](#)].



Also, a number of academic papers have explored how LISP can be used to do TE, and how effective it can be. See the online LISP Bibliography ([[Bibliography](#)]) for information about them.

#### **[7.4.](#) Routing**

{{Suggestion by editors: Remove this section, LISP is not a routing protocol.}}

Multi-homing and Traffic Engineering are both, in some sense, uses of LISP for routing, but there are many other routing-related uses for LISP.

One of the major original motivations for the separation of location and identity in general, and thus LISP, was to reduce the growth of the routing tables in the "Internet core", the part where routes to all ultimate destinations must be available. LISP is expected to help with this; for more detail, see [Section 15.4](#), "LISP and Core Internet Routing", below.

LISP may also have more local applications in which it can help with routing; see, for instance, [[CorasBGP](#)].

#### **[7.5.](#) Mobility**

{{Suggestion by editors: Remove this section, mobility is not in charter.}}

Mobility is yet another place where separation of location and identity is a key part of a clean, efficient and high-functionality solution. Considerable experimentation has been completed on doing mobility with LISP.

The mobility provided by LISP allows active sessions to survive moves (provided of course that there is not a period of inaccessibility which exceeds a timeout). LISP mobility also will typically have better packet stretch (i.e. increase in path length) compared to traditional mobility schemes, which use a home agent.

#### **[7.6.](#) Traversal Across Alternate IP Versions**

Note that LISP inherently supports intermixing of various IP versions for packet carriage; IPv4 packets might well be carried in IPv6, or vice versa, depending on the network's configuration.

This capability allows an island of operation of one type to be automatically tunneled over a stretch of infrastructure which only supports the other type.



### **7.7. Virtual Private Networks**

{{Suggestion by editors: Remove this section, not covered by any WG document}}

L2 and L3 {{Need to add text here - This used to be part of 'Local' below, but we decided this was so important it deserved its own section. Maybe move this up further, as it seems to be the most important 'early adopter' application?}}

The mapping-and-encapsulation nature of LISP makes it a perfect candidate for VPN solutions. In this case, private parts of the VPN form LISP sites and the IP addresses of devices in the private parts are composing EID spaces. The interconnection between the LISP sites is the public network and IP addresses in the interconnection compose the RLOC space. A major advantage of using LISP to construct VPN resides in the simplicity of the configurations: each xTR (i.e., VPN end) is configured to query the mapping system to retrieve mappings making the glue between the public (RLOC space) and the private (EID space) of the VPN.

This includes support of multi-tenancy where several private networks can be carried over the same underlayer network. Thanks to the instance-id field, multi-tenant network can even use the exact same addresses as the xTR distinguishes the tenant based on the value of the instance-id. The multiple address family support in LISP mappings also allows to build private networks using a different addressing family than the carrier (e.g., IPv6 over IPv4).

### **7.8. Local Uses**

{{Suggestion by editors: Remove this section. The section contains a general discussion about the applicability of LISP in intra-domain scenarios, however it does not describe any specific application.}}

LISP has a number of use cases which are within purely organizationally-local contexts, i.e. not in the larger Internet. These fall into two categories: uses seen on the Internet (above), but here on a private (and usually small scale) setting; and applications which do not have a direct analog in the larger Internet, and which apply only to local deployments.

Among the former are multi-homing and IP version traversal. {{This was marked to be deleted - why? The next part doesn't make sense without this first?}}

Among the latter class, non-Internet applications which have no analog on the Internet, are the following example applications:



virtual machine mobility in data centers; non-IP EID types such as L2 MAC addresses, or application specific data.

Several of the applications listed in this section are the ones which have been most popular for LISP in practise; these include virtual networks, and virtual machine mobility.

These often show a synergistic tendency, in that a site which installs LISP to do one, often finds that then becomes a small matter to use it for the second. Given all the things which LISP can do, it is hoped that this synergistic effect will continue to expand LISP's uses.

{{Preceeding paragraphs should probably get moved up into VPN section?}}

## **8. Major Functional Subsystems**

{{Suggestion by editors: This section should appear before since it describes key aspects of LISP (e.g., LISP decoupled control and data-plane).}}

LISP has two major functional sub-systems: the xTRs which form the data-plane of LISP; and the mapping system which forms the control plane that maintains and distributes the mapping database.

The purpose and operation of each is described at a high level below, and then, later on, in a fair amount of detail, in separate sections on each ([Section 12](#) and [Section 13](#)).

### **8.1. Data Plane - xTRs Overview**

{{Suggestion by editors: Extend this section, it misses key LISP data-plane aspects, such as the map-cache.}}

xTRs are routers that have been augmented with extra functionality in both the data and control planes. The data plane functions in ITRs include deciding which packets need to be given LISP processing (since packets to non-LISP hosts may be sent as they are); i.e. looking up the mapping; encapsulating (wrapping) the packet; and sending it to the ETR.

This encapsulation is done using UDP [[RFC0768](#)], along with an additional outer IP header (to hold the source and destination RLOCs). To the extent that traffic engineering features are in use for a particular EID, the ITRs implement them as well.





In the ETR, the data plane simply decapsulates (unwraps) the packets, and forwards the it to the destination.

Control plane functions in ITRs include: asking for EID-to-RLOC mappings via Map-Request packets; handling the returning reply control messages (i.e., Map-Reply packets), which contain the EID-to-RLOC mapping; managing the local mapping cache and checking for the reachability of destination ETR's RLOCs.

In the ETR, control plane functions include participating in the reachability tests (see [Section 16.4](#)); interacting with the mapping sub-system to let it know what mapping this ETR can provide (see [Section 8.2.2](#)); and answering Map-Request packets from ITRs for those mappings.

#### **8.1.1. Mapping Cache Performance**

{{Suggestion by editors: Push this section further in the document, the cache performance is not a "Major LISP subsystem"}}

As mentioned, studies have been performed to verify that caching mappings in ITRs is viable, in practical engineering terms. These studies not only verified that such caching is feasible, but also provided some insight for designing ITR "mapping caches".

Briefly, they took lengthy traces of all packets leaving a large site, over a period of a week or so, and used those to drive simulations which showed how many mappings would be required. It also allowed analysis of how much control traffic (for loading needed mappings) would result, using various cache sizes and replacement algorithms. These studies provide an insight into how well LISP can be expected to perform, and scale, over time.

A more extended look at the results us given below, in [Section 12.9](#), "xTR Mapping Cache Performance".

#### **8.2. Control Plane - Mapping System Overview**

{{Suggestion by editors: Rewrite: Replace EID block terminology (along with its description) with the very common term "prefix". Describe Map-Request/Map-Reply LISP signaling messages.}}

The mapping system's entire purpose is to give ITRs on-demand access to the mapping database, which is a distributed, and potentially replicated, database which holds mappings between EIDs (identity) and RLOCs (location), along with needed ancillary data (e.g. lifetimes).



To be exact, it contains mappings between EID blocks and RLOCs (the block size is given explicitly, as part of the syntax). Support for blocks is both for minimizing the administrative configuration overhead, as well as for operational efficiency; e.g. when a group of EIDs are behind a single xTR. In IP blocks are delimited by their IP prefix.

However, the block may be, and sometimes is, as small as a single EID. However, since mappings are only loaded upon demand, if smaller blocks become predominant, then the increased size of the overall database is far less problematic than if the Internet's routing tables came to be dominated by such small entries.

A particular EID (or EID block) may be associated to than one RLOC, and may change the association with time.

Also, in general, throughout LISP, the address family of EIDs and RLOCs is explicitly mentioned in control packet.

Finally, the mapping from an EID (or EID block) contains not just the RLOC(s), but also (for each RLOC for any given EID entry) priority and weight fields (to allow allocation of load between several RLOCs at a given priority); this allows a certain amount of traffic engineering to be accomplished with LISP.

#### **8.2.1. Mapping System Organization**

{{Suggestion by editors: Rewrite: Describe key Mapping System components: Map-Server/Map-Resolver}}

The mapping system is split into three functional sub-systems.

The first is the actual mappings themselves, collectively the mapping database; they are held by the ETRs, and an ITR which needs a mapping gets it (effectively) directly from the ETR. This co-location of the authoritative version of the mappings, and the forwarding functionality which it describes, is an instance of fate-sharing.  
[\[Clark\]](#)

To find the appropriate ETR(s) to query for the mapping, the second two sub-systems form an indexing system, itself also based on a distributed, potentially replicated database. It provides information on which ETR(s) are authoritative sources for the various EID-to-RLOC mappings which are available. The two sub-systems which form it are the client interface sub-system, and indexing sub-system (which holds and provides the actual information).



### **8.2.2. Interface to the Mapping System**

{{Suggestion by editors: has been rewritten: This section should appear earlier since it describes key LISP components (Map-Request/Map-Reply/Map-Server/Map-Resolver}}}

The client interface to the indexing system from an ITR's point of view is not with the indexing sub-system directly; rather, it is through the Map-Resolvers (MRs) and Map-Servers (MSs).

To obtain the mapping for an EID, the ITRs sends Map-Request packet to an MR. The MR then uses the indexing sub-system to allow it to forward the Map-Request to an appropriate Map-Server (MS), which in turn sends the Map-Request on to the appropriate ETR. The latter is authoritative for the actual mappings for the EID.

The ETR then sends the mappings for that EID in the form of aMap-Reply packets, which is sent directly to the ITR, without passing through the indexing sub-system and MR. The details of the indexing sub-system are thus hidden from the ITRs.

Note that in proxy mode, the MS replies on behalf of the ETR. When this the case, the map-replies is tagged to indicate that the answer is not delivered from the authoritative ETR but from the MS instead.

The interface to the indexing system from an ETR's point of view is made through MSes. ETRs send Map-Register packets to their designated MSes. The effect of a Map-Register is to inform the MS about the mappings maintained by ETRs such that the MS can transmit the Map-Requests is receives to the appropriate ETRs.

The MS optionally replies Map-Register packets with a Map-Notify packet to confirm the registration. The details of the indexing sub-system are thus likewise hidden from ETRs.

The fact that the details of the indexing sub-system are entirely hidden from xTRs gives considerably flexibility to this aspect of LISP. As long as any potential indexing sub-system can track where mappings are, it could potentially be used; this would allow the actual indexing sub-system to be replaced without needing to modify the xTRs.

### **8.2.3. Indexing Sub-system**

{{suggestion by editor: rename the section to "DDT"}}}

The current indexing sub-system is the Delegated Database Tree (DDT), which is conceptually similar to DNS ([[I-D.ietf-lisp-ddt](#)],



[[RFC1034](#)]). DDT replaces the earlier LISP+ALT indexing sub-system ([[RFC6836](#)]). The seamless migration to DDT while LISP+ALT was previously used validated the concept of having a client-interface sub-system between the indexing sub-system, and the clients.

#### **8.2.3.1. DDT Overview**

Conceptually, DDT is similar to the DNS, in DDT the delegation of the EID namespace is instantiated as a delegation hierarchy, a tree of DDT nodes, starting with the root DDT node. Each vertex is responsible for a block of the EID namespace.

The root node is responsible for the entire EID space; any DDT node can delegate part(s) of its EID block to child DDT node(s). The child node(s) can in turn further delegate (necessarily smaller) blocks of namespace to their children, through as many levels as are needed (for operational, administrative, etc, needs).

Just as with DNS, any particular vertex in the DDT delegation tree may be instantiated in one or more DDT servers. Multiple (redundant) servers for a given node would be used for reasons of performance, reliability, and robustness. Obviously, all the servers which instantiate a particular nodes in the tree have to have identical data about that node.

#### **8.2.3.2. Use of DDT by MRs**

An MR which wants to find a mapping for a particular EID first interacts with the DDT servers which instantiate the nodes of the LISP delegation hierarchy tree, discovering (by querying the servers for information about DDT nodes) the chain of delegations which cover that EID. Eventually it is directed to an MS, which is servicing an ETR which is authoritative for that EID.

Also, again like DNS, MRs may cache information they receive about the delegations in the delegation tree. This means that once an MR has been in operation for a while, it will usually have much of the delegation information cached locally (especially the top levels of the delegation tree). This allows them, when passed a request for a mapping by an ITR, to usually forward the mapping request to the appropriate MS without having to interact with all the DDT servers on the path down the delegation tree, in order to find any particular mapping.

Thus, a typical resolution cycle would usually involve looking at some locally cached delegation information, perhaps loading some missing delegation entries into their delegation cache, and finally sending the Map-Request to the appropriate MS.





It should also be noted that the delegation tree is fairly static, since it reflects EID block allocations, which are themselves fairly static. This stability has several important consequences. First, it increases the performance of the mapping system, since the sub-system almost never needs to be re-queried for information about intermediate vertices. {{comment from editor: does not understand the next sentence...}}Second, it is not necessary to include a mechanism to find out-dated delegations. [[LISP-TREE](#)]

This contrasts with the mappings, which may change at a high rate - changes which have no impact on the indexing sub-system. The potentially high dynamics of mappings explains why mappings are delivered directly from ETRs (or MSes in proxy mode) to ITRs and hence not cached at the MR level. LISP is designed to make sure that changes in the mappings are detected and acted upon fairly quickly; this allows LISP to provide a number of capabilities, such as mobility.

## **9. Examples of operation**

To aid in comprehension, a few examples are given of user packets traversing the LISP system. The first shows the processing of a typical user packet which is LISP forwarded, i.e. what the vast majority of user packets will see. The second shows what happens when the first packet to a previously-unseen ultimate destination (at a particular ITR) is to be processed by LISP.

### **[9.1.](#) An Ordinary Packet's Processing**

{{Suggestion by editors: This section should be earlier in the document structure, examples are important -particularly for engineers- to explain how systems work}}

This case follows the processing of a typical , {{comment from editors: text missing}} when the packet has made its way through the local site to an ITR, which in this case is a border router for the site, the border router looks up the destination address - an EID - in its local mapping cache. For EIDs which are IP addresses, this lookup uses the IP longest prefix matching algorithm.

It finds a mapping, which instructs it to wrap the packet in an outer header - an IP packet, containing a UDP packet which contains a LISP header - and then the user's original packet (see [Section 12.2](#) for the reasons for this particular choice). The destination address in the outer header is set by the ITR to the RLOC of the destination ETR.



The encapsulated packet is then sent off through the RLOC space (e.g., Internet), using normal Internet routing.

On arrival at the destination ETR, the ETR will notice that it is listed as the destination in the outer header. It will examine the packet, detect that it is a LISP packet, and unwrap it. It will then examine the header of the user's original packet, and forward it internally, through the local site, to the ultimate destination.

At the ultimate destination, the packet will be processed, and may produce a return packet, which follows the exact same process in reverse - with the exception that the roles of the ITR and ETR are swapped.

### 9.2. A Mapping Cache Miss

{{Suggestion by editors: (same as previous section)}}

If a host sends a packet, and it gets to the ITR, and the ITR determines that it does not yet have a mapping cache entry which covers that destination EID, then additional processing ensues; it has to look up the mapping in the mapping system (as previously described in [Section 6.2](#)).

The overall processing is shown below, in Figure 2:

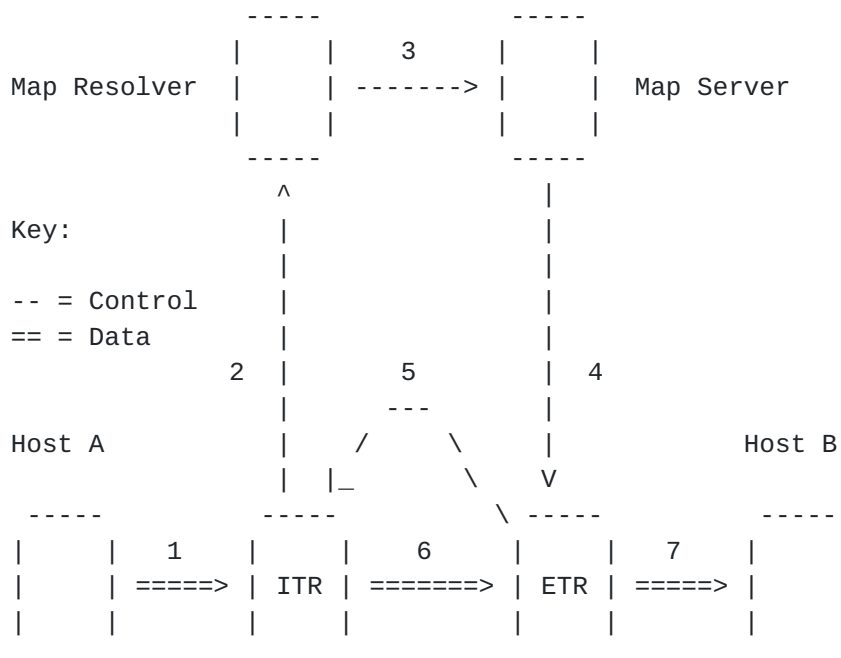


Figure 2: Packet flow with missing mapping



1. Source-EID sends packet (to Dest-EID) to ITR
2. ITR sends Map-Request to Map-Resolver
3. Map-Resolver delivers Map-Request to Map-Server
4. Map-Server delivers Map-Request to ETR
5. ETR returns Map-Reply to ITR; ITR caches EID-to-RLOC(s) mapping
6. ITR uses mapping to encapsulate to ETR; sends user packet to ETR
7. ETR decapsulates packet, delivers to Dest-EID

The ITR first sends a Map-Request packet, giving the destination EID it needs a mapping for, to its MR. The MR will look in its cache of delegation information to find the vertex which is the most specific in the delegation tree for that destination EID . If it does not have the address of an appropriate MS, it will query the DDT system, recursively if need be, in order to eventually find the address of such an MS.

When it has the MS's address, it will send the Map-Request on to the MS, which then usually sends it on to an appropriate ETR. The ETR sends a Map-Reply to the ITR which needs the mapping; from then on, processing of user packets through that ITR to that ultimate destination proceeds as above.

To the best of our knowledge, in all LISP implementations, the original user packet will have been discarded, and not queued waiting for the mapping to be returned. When the host retransmits such a packet, the mapping will be there, and the packet will be forwarded. Alternatively, it might have been forwarded using a PITR to avoid being dropped ([Section 6.4](#)).

## **10. Part II**

{{comment from editors: is that the role of an introductory document to enter in such level of details and discuss (mostly) all fields of the protocol? }}

## **11. Design Approach**

{{Suggestion by editors: Remove this section, it does not describe/discuss anything relevant, it is only providing a reference to another non-WG document.}}



Before describing LISP's components in more detail below, it is worth pointing out that what may seem, in some cases, like odd (or poor) design approaches do in fact result from the application of a thought-through, and consistent, design philosophy used in creating them. {{Subjective: maybe JMH, Dino can help with better words?}}

This design philosophy is covered in detail in [\[I-D.ietf-lisp-perspective\]](#), Section "Design", and readers who are interested in the 'why' of various mechanisms should consult that; reading it may make clearer the reasons for some engineering choices in the mechanisms given here.

## **[12.](#) xTRs**

As mentioned above (in [Section 8.1](#)), xTRs are the essential LISP data plane elements. This section explores some advanced topics related to xTRs.

{{Suggestion by editors: remove next paragraph, brings nothing}}

Careful rules have been specified for both TTL and ECN [[RFC3168](#)] to ensure that passage through xTRs does not interfere with the operation of these mechanisms. In addition, care has been taken to ensure that traceroute works when xTRs are involved.

### **[12.1.](#) When to Encapsulate**

An ITR knows that an ultimate destination is running LISP (remember that the actual destination machine itself probably knows nothing about LISP), and thus that it should perform LISP processing on a packet (including potential encapsulation) if it has an entry in its local mapping cache that covers the destination address (it is then called an EID).

Conversely, if the cache contains a negative entry (indicating that the ITR has previously attempted to find a mapping that covers this EID, and it has been informed by the mapping system that no such mapping exists), it knows the ultimate destination is not running LISP, and the packet can be forwarded natively (i.e. not LISP-encapsulated).

Note that the ITR cannot simply depend on the appearance, or non-appearance, of the destination in the routing tables in the Internet core, as a way to tell if a destination is a LISP node or not. That is because mechanisms to allow interoperation of LISP sites and legacy sites necessarily involve advertising LISP sites' EIDs into the Internet core; in other words, LISP sites which need to





interoperate with legacy nodes will appear in the Internet core routing tables, along with non-LISP sites.

## **12.2. UDP Encapsulation Details**

Use of UDP (instead of, say, a LISP-specific protocol number) was driven by the fact that many routers and middle boxes filter out unknown protocols, so adopting a non-UDP encapsulation would have compromised the initial deployment of LISP.

The UDP source port used for encapsulating packet is computed with a 5-way hash of the original source and destination in the inner header, along with the ports, and the protocol. This is because many ISPs use multiple parallel paths (so-called Equal Cost Multi-Path), and load-share across them. Using such a hash in the source-port in the outer header both allows LISP traffic to be load-shared, and also ensures that packets from individual connections are delivered in order (since most ISPs try to ensure that packets for a particular flow follow a single path, and hence do not become disordered).

The UDP checksum is zero because the inner packet usually already has a end-end checksum, and the outer checksum adds no value ([\[Saltzer\]](#)). {{Suggestion by editors: not sure that next statement is correct}} In most existing hardware, computing such a checksum (and checking it at the other end) would also present a major load, for no benefit.

## **12.3. Header Control Channel**

{{Suggestion by editors: Rewrite this section to improve readability, use standard terminology (e.g., Cache Coherence/Reachability instead of "Header Control Channel"). Split the mechanisms depending on its goal (cache coherence/reachability) and describe them under the same context.}}

LISP provides a multiplexed channel in the encapsulation header. It is mostly (but not entirely) used for control purposes. The general concept is that the header starts with a flags field, and it also includes two data fields, the contents and meaning of which vary, depending on which flags are set. This allows these fields to be multiplexed among a number of different low-duty-cycle functions, while minimizing the space overhead of the LISP.

### **12.3.1. Mapping Versioning**

One important use of the multiplexed control channel is mapping versioning; i.e. the discovery of when the mapping cached in an ITR is outdated. To allow an ITR to discover this, identifying sequence numbers are applied to different versions of a mapping [\[RFC6834\]](#).



This allows an ITR to easily discover when a cached mapping has been updated by a more recent variant.

Version numbers are available in control messages (Map-Replies), but the initial concept is that to limit control message overhead, the versioning mechanism should primarily use the multiplexed user data header control channel.

Versioning can operate in both directions: an ITR can advise an ETR what version of a mapping it is currently using (so the ETR can notify it if there is a more recent version), and ETRs can let ITRs know what the current mapping version is (so the ITRs can request an update, if their copy is outdated).

#### **12.3.2. Echo Nonces**

Another important use of the header control channel is for a mechanism known as the Nonce Echo, which is used as an efficient method for ITRs to check the reachability of neighbour ETRs.

Basically, an ITR which has to determine that an ETR is up, and reachable, sends a nonce to that ETR, carried in the encapsulation header; when that ETR (also acting as an ITR) sends some other user data packet back to the ITR (acting in turn as an ETR), that nonce is carried in the header of that packet, allowing the original ITR to confirm that its packets are reaching that ETR.

Note that a lack of a response is not necessarily proof that something has gone wrong - but it strongly suggests that something has, so other actions (e.g. a switch to an alternative ETR, if one is listed; a direct probe; etc) are advised. (See [Section 16.5](#) for more about Echo Nonces.)

#### **12.3.3. Instances**

{{Suggestion by editors: Move and extend this section: - Instance ID is not a cache-coherence/reachability algorithm. Describe where and what is the Instance ID field Explain its applications}}

The LISP data-plane header is also used to support VPN and multi-tenant networks. Since there is only one destination UDP port used for carriage of user data packets, and the source port is used for multiplexing, there is no other way to differentiate among different destination EID spaces (which are often overlapped in VPNs and multi-tenant networks).



#### **12.4. Probing**

RLOC-Probing is a mechanism method that an ITR can use to determine with that an ETR is up and reachable from the ITR. As a side-benefit, it gives RTT estimates.

To probe the reachability of an RLOC, an ITR sends a specially marked Map-Request directly to the ETR it wishes information about; that ETR sends back a specially marked Map-Reply. A Map-Request message and a Map-Reply message are used, rather than a special probing control-message pair, because as a side-benefit the ITR can discover if the mapping has been updated since it cached it.

{{Suggestion from editors: remove the following sentence as it is not motivated by strong arguments}} The probing mechanism is rather heavy-weight and expensive (compared to mechanisms like the Echo-Nonce), since it costs a control message from each side, so it should only be used sparingly.

If the number of active neighbour ETRs of the ITR is large, use of RLOC-Probing to check on their reachability will result in considerable control traffic; such control traffic has to be spread out to prevent a load peak.

Obviously, if RLOC-Probing is the only mechanism being used to detect unreachable neighbour ETRs, the rate at which RLOC-Probing is done will control the timeliness of the detection of loss of reachability. There is thus a tradeoff between overhead and responsiveness, particular when an ITR has a large fanout of neighbour ETRs.

#### **12.5. Mapping Lifetimes and Timeouts**

{{Suggestion by editors: Remove this section, TTL is a simple well-known concept, we suggest to include a sentence (and hence remove this section) in the control-plane section stating that mappings include a TTL.

Mappings come with a Time-To-Live, which indicate how long the creator of the mapping expects them to be useful for.

Mappings might also be discarded before the TTL expires, depending on what strategies the ITR is using to maintain its cache; if the maximum cache size is fixed, or the ITR needs to reclaim memory, mappings which have not been used recently may be discarded. (After all, there is no harm in so doing; a future reference will merely cause that mapping to be reloaded.)

{{Contents may change before TTL expires?}}



### **12.6. Mapping Gleaning in ETRs**

{{Suggestion by editors: Remove this section, gleaning is discouraged since it rises many security concerns.}}

As an optimization to the mapping acquisition process, ETRs are allowed to glean mappings from incoming user data packets, and also from incoming Map-Request control messages. This is not secure, and so any such mapping must be verified by sending a Map-Request to get an authoritative mapping.

The value of gleaning is that most communications are two-way, and so if host A is sending packets to host B (therefore needing B's EID->RLOC mapping), very likely B will soon be sending packets back to A (and thus needing A's EID->RLOC mapping). Without gleaning, this would sometimes result in a delay, and the dropping of the first return packet; this is felt to be very undesirable.

### **12.7. MTU Issues**

Several mechanisms have been proposed for dealing with packets which are too large to transit the path from a particular ITR to a given ETR.

In one, called the stateful approach, the ITR keeps a per-ETR record of the maximum size allowed, and sends an ICMP Too Big message to the original source host when a packet which is too large is seen.

In the other, referred to as the stateless approach, for IPv4 packets without the DF bit set, too-large packets are fragmented, and then the fragments are forwarded; all other packets are discarded, and an ICMP Too Big message returned.

### **12.8. Security of Mapping Lookups**

{{Suggestion from editors: would remove all what is related to security and instead put a small summary in the security section and reference the threats document}}

LISP provides an optional mechanism to secure the obtaining of mappings by an ITR. [[I-D.ietf-lisp-sec](#)] It provides protection against attackers generating spurious Map-Reply messages (including replaying old Map-Replies), and also against over-claiming attacks (where a malicious ETR by claims EID-prefixes which are larger than what have been actually delegated to it).

In summary, the ITR provides a One-Time Key with its Map-Request; this key is used by both the MS (to sign an affirmation that it has





delegated that EID block to that ETR), and indirectly by the ETR (to sign the mapping that it is returning to the ITR).

The specification for LISP-SEC suggests that the ITR-MR stage be cryptographically protected, and indicates that the existing mechanisms for securing the ETR-MS stage are used to protect Map-Requests also. It does assume that the channel from the MR to the MS is secure.

### **12.9. ITR Mapping Cache Performance**

As mentioned previously ([Section 8.1.1](#)), a substantial amount of simulation work has been performed to predict, and understand, the performance of the mapping cache in xTRs.

Briefly, however, the first, [[Iannone](#)], was performed in the very early stages of the LISP effort, to verify that that caching approach was feasible.

Packet traces of all traffic over the external connection of a large university over a week-long period were collected; simulations driven by these recording were then performed. A variety of control settings on the cache were used, to study the effects of varying the settings.

First, the simulation gave the cache sizes that would result from such a cache design: it showed that the resulting cache sizes ranged from 7,500 entries, up to about 100,000 (depending on factors such as traffic and entry retention time). Using some estimations as to how much memory mapping entries would use, this indicated cache sizes of between roughly 100 Kbytes and a few Mbytes.

Of more interest, in a way, were the results regarding two important measurements of the effectiveness of the cache: i) the hit ratio (i.e. the share of references which could be satisfied by the cache), and ii) the miss rate (since control traffic overhead is one of the chief concerns when using a cache). These results were also encouraging: miss (and hence lookup) rates ranged from 30 per minute, up to 3,000 per minute.

Significantly, this was substantially lower than the amount of observed DNS traffic, which ranged from 1,800 packets per minute up to 15,000 per minute. The results overall showed that using a demand-loaded cache was an entirely plausible design approach: both cache size, and the control plane traffic load, were definitely feasible.



The second, [[Kim](#)], was in general terms similar, except that it used data from a large ISP, one with about three times as many users as the previous study. It used the same cache design philosophy (the cache size was not fixed), but slightly different, lower, retention time values.

The results were similar: cache sizes ranges from 20,000 entries to roughly 60,000; the miss rate ranged from very roughly 400 per minute to very roughly 7,000 per minute, similar to the previous results.

Finally, a third study, [[CorasCache](#)], examined the effect of using a fixed size cache, and a purely Least Recently Used (LRU) cache eviction algorithm (i.e. no timeouts). It also tried to verify that models of the performance of such a cache (using previous theoretical work on caches) produced results that conformed with actual empirical measurements.

It used yet another set of packet traces; using a cache size of around 50,000 entries produced a miss rate of around  $1 \times 10^{-4}$ ; again, definitely viable, and in line with the results of the other studies.

### **13. The Mapping System**

{{Suggestion by editors: This section is somewhat redundant with respect to [section 8.2](#), we suggest to move this section to Part I since it provides some missing details.}}

As discussed already in [Section 8.2](#), the LISP mapping system is an important part of LISP's control plane: it i) maintains the database of mappings between EIDs, and the RLOCs at which they are to be found, and ii) provides those mappings to ITRs which request them, so that the ITRs can send traffic for a given EID to the correct RLOC(s) for that EID.

[RFC1034] has this to say about the DNS name to IP address database and mapping system:

"The sheer size of the database and frequency of updates suggest that it must be maintained in a distributed manner, with local caching to improve performance. Approaches that attempt to collect a consistent copy of the entire database will become more and more expensive and difficult, and hence should be avoided."

and this observation applies equally to the LISP mapping database and mapping system.

To briefly recap, the mapping system is split into three parts: i) an indexing sub-system, which keeps track of where all the mappings are



kept; ii) the interface to the indexing system (which remains the same, even if the actual indexing system is changed); and iii) the mappings themselves (collectively, the mapping database), the authoritative copies of which are always held by ETRs.

### **13.1. The Mapping System Interface**

As mentioned in [Section 8.2.2](#), both of the interfaces to the mapping system (from ITRs, and ETRs) are standardized, so that the more numerous xTRs do not have to be modified when the mapping indexing sub-system is changed.

This section describes the interfaces in a little more detail; for details, see [\[RFC6833\]](#).

#### **13.1.1. Map-Request Messages**

{{Suggestion from editors: should come much earlier as it is an essential message in LISP}}

The Map-Request message contains a number of fields, the two most important of which are the requested EID block identifier (remember that individual mappings may cover a block of EIDs, not just a single EID), and the Address Family Identifier (AFI) for that EID block.

Other important fields are the source EID (and its AFI), and one or more RLOCs for the source EID, along with their AFIs. The source EID and RLOCs allow ETRs to customize their answer.

Finally, the message includes a long nonce, for simple, efficient protection against offpath attackers and a variety of other fields and control flag bits.

#### **13.1.2. Map-Reply Messages**

{{Suggestion from editors: should come much earlier as it is an essential message in LISP}}

The Map-Reply message looks similar, except it includes the mapping entry for the requested EID(s), which contains one or more RLOCs and their associated data. Note that the reply may cover a larger block of the EID namespace than the request; most requests will be for a single EID, the one which prompted the query.

If there are no mappings available at all for the EID(s) requested, a Negative Map-Reply message will be returned. This is a Map-Reply message with flag bits set to indicate that fact.



For each RLOC in the entry, there is the RLOC, its AFI, priority and weight fields (see [Section 8.2](#)), and multicast priority and weight fields (see [Section 14](#)).

#### **[13.1.2.1](#). Solicit-Map-Request Messages**

Solicit-Map-Request (SMR) messages are actually not another message type, but a variant of Map-Request messages. They include a special flag which indicates to the recipient that it should send a new Map-Request message, to refresh its mapping, because the ETR has detected that the one it is using is out-dated.

SMR's, like most other control traffic, should be rate-limited.

#### **[13.1.3](#). Map-Register and Map-Notify Messages**

{{Suggestion by editors: As noted by the author add a paragraph describing how a xTR de-registers an EID-to-RLOC mapping.}}

{{Suggestion by editors: add a small paragraph to explain what Map-Registers are used for}}

The Map-Register message contains one or several mapping records, each with an individual Time-To-Live (TTL). Each of the records contains an EID (potentially, a block of EIDs) and its AFI, a version number for this mapping (see [Section 12.3.1](#) and a list of RLOCs and their AFIs.

{{Suggestion by editors: do not understand the following paragraph}} Each RLOC entry also includes the same data as in the Map-Replies (i.e. priority and weight); this is because in some circumstances it is advantageous to allow the MS to proxy reply on the ETR's behalf to Map-Request messages, and the MS needs this information when it does so.

Map-Notify messages have the exact same contents as Map-Register messages; they are purely acknowledgements.

The entire interaction is authenticated by use of a shared key, configured in the MS and ETR. Although the protocol does already allow for replacement of the encryption algorithm, it does not support automated key management (although it appears to fall under the exclusions in [[RFC4107](#)]).

LISP does not specify messages for de-registering an association with a MS. The de-registration is hence ensured by the expiration of a timer: if a MS does not receive Map-Register messages within given timeout, it de-register the association.





### **13.2. The DDT Indexing Sub-system**

{{Suggestion from the editors: this section does not add any fundamental value to the DDT overview section, propose to remove it to only keep the overview; too much details that should not appear in an intro document}}

As previously mentioned in [Section 8.2.3](#), DDT is the current indexing sub-system in LISP.

The overall functioning is conceptually fairly simple; an MR which needs a mapping starts at a server for the root DDT node (there will normally be more than one such server available, for both performance and robustness reasons), and through a combination of cached delegation information, and repetitive querying of a sequence of DDT servers, works its way down the delegation tree until it arrives at an MS which is authoritative for the block of EID which holds the destination EID in question.

The interaction between MRs and DDT servers is as follow. The MR sends to the DDT server a Map-Request control message. The DDT server uses its data (which is configured, and static) to see whether it is directly peered to an MS which can answer the request, or if it has a child (or children, if replicated) which is responsible for that portion of the EID blocks.

If it has children configured which are responsible, it will reply to the MR with another kind of LISP control message, a Map-Referral message, which provides information about the delegation of the block containing the requested EID. This step is secured via authentication.

The Map-Referral also gives the addresses of DDT servers for that block and the MR can then send Map-Requests to any one (or all) of them. In addition, the Map-Referral includes keying material for the children, which allows any information provided by them to be cryptographically verified.

Control flags in the Map-Referral indicate to the querying MR whether the referral is to another DDT server, an MS, or an ETR. {{All three? Check}} If the former, the MR then sends the Map-Request to the child DDT server, repeating the process.

If the second, the MR then interacts with that MS, and usually the block's ETR(s) as well, to cause a mapping to be sent to the ITR which queried the MR for it. Recall that some MS's provide Map-Replies on behalf of an associated ETR, in so-called 'proxy mode', so in such cases the Map-Reply will come from the MS, not the ETR.



Delegations are cached in the MRs, so that once an MR has received information about a delegation, it usually will not need to look that up again. Once it has been in operation for a short while, there will usually only be a limited amount of delegation information which it has not yet asked about - probably only the last stage in a delegation to a leaf MS.

### **13.3. Reliability via Replication**

{{Suggestion by editors: Remove this section, describes operational practices/policies of the DDT infrastructure, this is beyond the scope of this document.}}

Everywhere throughout the mapping system, robustness to operational failures is obtained by replicating data in multiple instances of any particular node (of whatever type). Map-Resolvers, Map-Servers, DDT nodes, ETRs - all of them can be replicated, and the protocol supports this replication.

There are generally no mechanisms specified yet to ensure coherence between multiple copies of any particular data item (e.g. the copies of delegation data for a particular block of namespace, in DDT sibling servers) - this is currently a manual responsibility.

If and when LISP protocol adoption proceeds, an automated layer to perform this functionality can easily be layered on top of the existing mechanisms.

The deployed DDT system actually uses anycast [[RFC4786](#)], along with replicated servers, to improve both performance and robustness.

### **13.4. Security of the DDT Indexing Sub-system**

{{Suggestion by editors: Remove this section, provides unnecessary details of DDT.}}

In summary, securing the mapping indexing system is divided into two parts: the interface between the clients of the system (MR's) and the mapping indexing system itself, and the interaction between the DDT servers which make it up.

The client interface provides only a single model, using the canonical public-private key system (starting from a trust anchor), in which the child's public key is provided by the parent, along with the delegation. When the child returns any data, it can sign the data, and the requestor can use that signature to verify the data. This requires very little configuration in the clients.



The interface between the DDT servers allows for choices between a number of different options, allowing the operators to trade off among configuration complexity, security level, etc. This is based on experience with DNSSEC ([\[RFC4033\]](#)), where configuration complexity has been a major stumbling block to deployment.

### **13.5. Extended Capabilities**

{{Suggestion by editors: Remove this section, not discussed in any WG document.}}

In addition to the priority and weight data items in mappings, LISP offers other tools to enhance functionality, particularly in the traffic engineering area.

One is requestor-specific mappings, i.e. the ETR may return different mappings to the enquiring ITR, depending on the identity of the ITR. This allows very fine-tuned traffic engineering, far more powerful than routing-based TE.

### **13.6. Performance of the Mapping System**

Prior to the creation of DDT, a large study of the performance of the previous mapping system, ALT ([\[RFC6836\]](#)), along with a proposed new design called TREE (which used DNS to hold delegation information) provided considerable insight into the likely performance of the mapping systems at larger scale (See [\[LISP-TREE\]](#)).

The basic structure and concepts of DDT are identical to those of TREE, so the performance simulation work done for that design applies equally to DDT.

{{Suggestion from editors: next sentence is redundant with previous parts of the document}} In that study, as with earlier LISP performance analyses, extensive large-scale simulations were driven by lengthy recordings of actual traffic at several major sites; one was the site in the first study ([\[Iannone\]](#)), and the other was an even large university, with roughly 35,000 users.

The results showed that a system like DDT, which caches information about delegations, and allows the MR to communicate directly with the servers for the lower vertices on the delegation hierarchy based on cached delegation information, would have good performance, with average resolution times on the order of the MR to MS RTT. This verified the effectiveness of this particular type of indexing system.



A more recent study, [[Saucez](#)], has measured actual resolution times in the deployed LISP network; it took measurements from a variety of locations in the Internet, with respect to a number of different target EIDs. Average measured resolution delays ranged from roughly 175 msec to 225 msec, depending on the location.

#### **14. Multicast Support in LISP**

{{Suggestion by editors: Rewrite this section, it provides too many details. Reduce it to a brief description of LISP Multicast}}

LISP and its intrinsic separation of identity from location is well suited for Multicast ([[RFC3170](#)], [[RFC5110](#)]), and the definition of mappings in the current specifications already allows multicast capability ([[RFC6830](#)]).

{{Say something about sources.}}

{{Suggestion by editors: remove the paragraph below, motivation for multicast is out of the scope of this document}}

Multicast is an important requirement, for a number of reasons: doing multiple unicast streams is inefficient, as it is easy to use up all the upstream bandwidth; without multicast a server can also be saturated fairly easily in doing the unicast replication; etc.

Since it is important for LISP to work well with multicast; doing so has been a significant focus in LISP throughout its entire development.

Further very significant improvements to multicast support in LISP are in progress; see [[Improvements](#)], Section "Multicast" for more on them.

##### **14.1. Basic Concepts of Multicast Support in LISP**

This section introduces some of the basic principles of multicast support in LISP.

Since group addresses name distributed collective entities, in general they cannot have a single RLOC also. Since they usually refer to collections of entities the notion of endpoint identity must be

A multicast source at a LISP site may not be able to become the root of a distribution tree in the core if it uses its EID as its identity for that distribution tree (i.e. a distribution tree (S-EID, G)); that is because there may not be a route to its EID in the core





(assuming that its section of the core even supports multicast; not all parts of the core do).

Therefore, outside the LISP site, multicast state for the distribution tree (S-RLOC, G) needs to be built instead, where S-RLOC is the RLOC of the ITR that the multicast source inside the LISP site will be sending its traffic through.

Multicast LISP requires no packet format changes to existing multicast packets (both control, and user data). The initial multicast support in LISP uses existing multicast control mechanisms exclusively; improvements currently being worked on provide LISP-specific control mechanisms (see [[Improvements](#)]).

#### **14.2. Initial Multicast Support in LISP**

{{Suggestion by editors: remove this paragraph}}

Readers who wish to fully understand multicast support need to consult the appropriate specifications: LISP multicast issues are discussed in [[RFC6830](#), [Section 11](#)]; and see [[RFC6831](#)] for the full details of current multicast support in LISP.

With the multicast operation defined in [[RFC6831](#)]), destination group addresses are not mapped; only the source address (when the original source is inside a LISP site) needs to be mapped, both during distribution tree setup, as well as actual traffic delivery.

In other words, while LISP's mapping capability is used, at this stage it is only applied to the source, not the destination (as with most LISP activity). Thus, in LISP-encapsulated multicast packets in this mode, the inner source is the EID, and the outer source is the ITR's RLOC; both inner and outer destinations are the group's multicast address.

Note that this does mean that if the group is using separate source-specific trees for distribution, there isn't a separate distribution tree outside the LISP site for each different source of traffic to the group from inside the LISP site; they are all grouped together under a single source, the RLOC.

The issue of encapsulation is complex, because if the rest of the group outside the LISP site includes some members which are at other LISP sites (i.e. packets to them have to be encapsulated), and some members at legacy sites (i.e. encapsulated packets would not be understood), there is no simple answer. (The situation becomes even more complex when one considers that as hosts leave and join the group, it may switch back and forth between mixed and homogenous.)



This issue is too complex to fully cover here; see [Section 9.2.](#), "LISP Sites with Mixed Address Families", in [[RFC6831](#)], for complete coverage of this issue.

Basically, there are multicast equivalents of some of the legacy interoperability mechanisms used for unicast; mPITRs and mPETRs (multicast-capable PITRs and PETRs). When mixed groups are a possibility, two choices are available: i) send two copies (one encapsulated, and one not) of all traffic, or ii) employ mPETRs to distribute non-encapsulated copies to legacy group members.

## **[15.](#) Deployment Issues and Mechanisms**

{{Suggestion by editors: Remove this section, provides unnecessary details. Add a reference to the deployment RFC.}}

This section discusses several deployment issues in more detail. With LISP's heavy emphasis on practicality, much work has gone into making sure it works well in the real-world environments most people have to deal with.

### **[15.1.](#) LISP Deployment Needs**

As mentioned earlier ([Section 5.2](#)), LISP requires no change to almost all existing hosts and routers. Obviously, however, one must deploy something to run LISP. Exactly what that has to be will depend greatly on the details of the site's existing networking gear, and choices it makes for how to achieve LISP deployment.

The primary requirement is for one or more xTRs. These may be existing routers, just with new software loads, or it may require the deployment of new devices.

{{Suggestion by editors: next paragraph is barely understandable}}

LISP also requires a certain amount of LISP-specific support infrastructure, such as MRs, MSs, the DDT hierarchy, etc. However, much of this will either i) {{for the case where you are adding a new site using existing LISP infrastructure}} already be deployed, and if the new site can make arrangements to use it, it need do nothing else; or ii) those functions the site must provide may be co-located in other LISP devices (again, either new devices, or new software on existing ones).



## **15.2. Interworking Mechanisms**

One aspect which has received a lot of attention are the mechanisms previously referred to (in [Section 6.4](#)) to allow interoperation of LISP sites with so-called legacy sites which are not running LISP (yet).

{{Suggestion by editors: remove next paragraph as it talks about features (NAT based transition) not covered by the WG}}

There are two main approaches to such interworking: proxy routers (PITRs and PETRs), and an alternative mechanism using a router with combined NAT and LISP functionality; these are described in more detail here.

### **15.2.1. Proxy LISP Routers**

{{Suggestion by editors: Move this section to Part I. PxTRs are an important aspect of LISP and as such, should be described before. Furthermore simplify it, it currently contains too many details plus an additional discussion on the impact of Pitr that is out of the scope of the document.}}

PITRs (proxy ITRs) serve as ITRs for traffic from legacy hosts to nodes in LISP sites. PETRs (proxy ETRs) serve as ETRs for LISP traffic to legacy host.

Note that return traffic to a legacy host from a LISP-using node does not necessarily have to pass through an ITR/PETR pair - the original packets can usually just be sent directly to the ultimate destination. However, for some kinds of LISP operation (e.g. mobile nodes), this is not possible; in these situations, the PETR is needed.

#### **15.2.1.1. PITRs**

To serve as ITRs for traffic from legacy hosts to nodes in LISP sites, PITRs they have to advertise into the existing legacy backbone Internet routing the availability of whatever ranges of EIDs (i.e. of nodes using LISP) they are proxying for, so that legacy hosts will know where to send traffic to those LISP nodes. Pitr implies that EID prefixes are advertised in BGP.

This technique may have an impact on routing table in the Internet core, but it is not clear yet exactly what that impact will be; it is very dependent on the collected details of many individual deployment decisions.



A PITR may cover a group of EID blocks with a single EID advertisement to the core, in order to reduce the number of routing table entries added. In fact, at the moment, aggressive aggregation of EID announcements is performed, precisely to minimize the number of new announced routes added by this technique.

At the same time, if a site does traffic engineering with LISP instead of fine-grained BGP announcement, that will help keep table sizes down (and this is true even in the early stages of LISP deployment). The same is true for multi-homing. {{Maybe mixing two concepts? LISP TE tools will still apply to traffic between PITR and LISP site.}}

As mentioned previously ([Section 12.1](#)), an ITR at another LISP site can avoid using a PITR (i.e. it can detect that a given ultimate destination is not a legacy host, if a PITR is advertising it into the Internet core) by checking to see if a LISP mapping exists for that ultimate destination.

#### **15.2.1.2. PETRs**

PETRs (proxy ETRs) serve as ETRs for LISP traffic to legacy hosts, for cases where a LISP node cannot send packets to such hosts without encapsulation. That typically happens for one of two reasons.

First, it will happen in places where some device is implementing Unicast Reverse Path Forwarding (uRPF), to prevent a variety of negative behaviour; originating packets with the original source's EID in the source address field will result in them being filtered out and discarded.

{{Suggestion by editors: would adding an example be useful?}}

Second, it will happen when a LISP site wishes to send packets to a non-LISP site, and the path in between does not support the particular IP protocol version used by the original source along its entire length. Use of a PETR on the other side of the gap will allow the LISP site's packet to 'hop over' the gap, by utilizing LISP's built-in support for mixed protocol encapsulation.

PETRs are generally used by specific ITRs, which have the location of their PETRs configured into them. In other words, unlike normal ETRs, PETRs do not have to register themselves in the mapping database, on behalf of any legacy sites they serve.

Also, allowing an ITR to always send traffic leaving a site to a PETR does avoid having to choose whether or not to encapsulate packets; it





can just always encapsulate packets, sending them to the PETR if it has no specific mapping for the ultimate destination.

#### **15.2.2. LISP-NAT**

{{Suggestion by editors: Remove this section, LISP-NAT is not a WG document neither an inter-networking mechanisms.}}

A LISP-NAT router, as previously mentioned, combines LISP and NAT functionality, in order to allow a LISP site which is internally using addresses which cannot be globally routed to communicate with non-LISP sites elsewhere in the Internet. (In other words, the technique used by the PITR approach simply cannot be used in this case.)

To do this, a LISP-NAT performs the usual NAT functionality, and translates a host's source address(es) in packets passing through it from an inner value to an outer value, and storing that translation in a table, which it can use to similarly process subsequent packets (both outgoing and incoming). [[RFC6832](#)]

{{Suggestion by editors: remove the following paragraph, does not bring anything}}

There are two main cases where this might apply:

- o Sites using non-routable global addresses
- o Sites using private addresses [[RFC1918](#)]

#### **15.3. Use Through NAT Devices**

{{Suggestion by editors: Remove this section, LISP-NAT is not a WG document neither an inter-networking mechanisms.}}

NATs are both ubiquitous, and here to stay for a long time to come. [[RFC1631](#)] Thus, in the actual Internet of today, having any new mechanisms function well in the presence of NATs (i.e. with LISP xTRs behind a NAT device) is absolutely necessary.

LISP has produced a variety of mechanisms to do this. An experimental mechanism to support them had major limitations; it, and its limitations, are described in [Appendix B.5](#).



#### **15.4. LISP and Core Internet Routing**

{{Suggestion by editors: Remove this section, this is already explained in lisp-deployment}}

One of LISP's original motivations was to control the growth of the size of routing tables in the Internet core, the part where routes to all destinations must be available. As LISP becomes more widely deployed, it can help with this issue, in a variety of ways.

In covering this topic, one must recognize that conditions in various stages of LISP deployment (in terms of ubiquity) will have a large influence. [[RFC7215](#)] introduced useful terminology for this progression, in addition to some coverage of the topic (see [Section 5](#), "Migration to LISP"):

The loosely defined terms of early transition phase, late transition phase, and LISP Internet phase refer to time periods when LISP sites are a minority, a majority, or represent all edge networks respectively.

In the early phases of deployment, two primary effects will allow LISP to have a positive impact on the routing table growth:

- o Using LISP for traffic engineering instead of BGP
- o Aggregation of smaller PI sites into a single PIR advertisement

The first is fairly obvious (doing TE with BGP requires injecting more-specific routes into the Internet core routing tables, something doing TE with LISP avoids); the second is not guaranteed to happen (since it requires coordination among a number of different parties), and only time will tell if it does happen.

#### **16. Fault Discovery/Handling**

{{Suggestion by editors: Remove this section, although this section helps understanding how many of the LISP mechanisms work (particularly the ones described in [section 12](#)) it provides an unnecessary level of (operational) detail. This level of understanding should be achieved reading the main LISP spec. }}

The structure of LISP gives rise to a moderate number of failure modes.



### **16.1. Handling Missing Mappings**

To handling missing mappings, the ITR calls for the mapping, and in the meantime can either discard traffic to that ultimate destination (as many ARP implementations do) [[RFC0826](#)], or, if dropping the traffic is deemed undesirable, it can forward them via a PITR.

### **16.2. Outdated Mappings**

If a mapping changes once an ITR has retrieved it, that may result in traffic to the EIDs covered by that mapping failing. There are three cases to consider:

- o When the ETR to which traffic is being sent is still a valid ETR for that EID, but the mapping has been updated (e.g. to change the priority of various ETRs)
- o When the ETR traffic is being sent to is still an ETR, but no longer a valid ETR for that EID
- o When the ETR traffic is being sent to is no longer an ETR

#### **16.2.1. Outdated Mappings - Updated Mapping**

A 'mapping versioning' system, whereby mappings have version numbers, and ITRs are notified when their mapping is out of date, has been added to detect this, and the ITR responds by refreshing the mapping. [[RFC6834](#)]

#### **16.2.2. Outdated Mappings - Wrong ETR**

If an ITR is holding an outdated cached mapping, it may send packets to an ETR which is no longer an ETR for that EID.

It might be argued that if the ETR is properly managing the lifetimes on its mapping entries, this cannot happen, but it is a wise design methodology to assume that cannot happen events will in fact happen (as they do, due to software errors, or, on rare occasions, hardware faults), and ensure that the system will handle them properly.

ETRs can easily detect cases where this happens, after they have unwrapped a user data packet; in response, they send a Solicit-Map-Request to the source ITR to cause it to refresh its mapping.



### **16.2.3. Outdated Mappings - No Longer an ETR**

In another case for what can happen if an ITR uses an outdated mapping, the destination of traffic from an ITR might no longer be a LISP node at all. In such cases, one might get an ICMP Destination Unreachable (Port Unreachable subtype) error message. However, one cannot depend on that - and in any event, that would provide an attack vector, so it should be used with care. (See [\[RFC6830\]](#), [Section 6.3](#) for more about this.)

The following mechanism will work, though. Since the destination is not an ETR, the echoing reachability detection mechanism (see [Section 12.3.2](#), "Echo Nonces") will detect a problem. At that point, the backstop mechanism, Probing, will kick in. Since the destination is still not an ETR, that will fail, too.

At that point, traffic will be switched to a different ETR, or, if none are available, a reload of the mapping may be initiated.

### **16.3. Erroneous Mappings**

{{Suggestion by editors: this section brings nothing, remove it}}

Again, this should not happen, but a good system should deal with it. However, in practise, should this happen, it will produce one of the prior two cases (the wrong ETR, or something that is not an ETR), and will be handled as described there.

### **16.4. Verifying ETR Liveness**

The ITR, like all packet switches, needs to detect, and react, when its neighbour ceases operation. As LISP traffic is effectively always uni-directional (from ITR to ETR), this could be somewhat problematic.

Solving a related problem, "neighbour ETR" "reachability" below) subsumes handling this fault mode, however.

{{Suggestion by editors: remove next paragraph }}

Note that the two terms - liveness and reachability - are not synonymous (although they are often confused). Liveness is a property of a node - it is either up and functioning, or it is not. Reachability is only a property of a particular pair of nodes.

If packets sent from a first node to a second are successfully received at the second, it is reachable from the first. However, the second node may at the very same time not be reachable from some





other node. Reachability is always a ordered pairwise property, and of a specified ordered pair.

### **16.5. Verifying ETR Reachability**

A more significant issue than whether a particular ETR is up or not is, as mentioned above, that although the ETR may be up, attached to the network, etc, an issue in the network, between a source ITR, and the ETR, may prevent traffic from the ITR from getting to the ETR. (Perhaps a routing problem, or perhaps some sort of access control setting.)

The one-way nature of LISP traffic makes this situation hard to detect with simple and non-intrusive techniques. In line with the LISP design philosophy this problem is then attacked not with a single mechanism (which would be complex) but with a collection of simple mechanisms.

They are reliance on the underlying routing system (which can of course only reliably provide a negative reachability indication, not a positive one), the echo nonce (which depends on some return traffic from the destination xTR back to the source xTR), and finally RLOC probing, in the case where no positive echo is returned.

{{Suggestion by editors: remove next paragraph }}

(The last is not the first choice, as due to the large fan-out expected of LISP router, reliance on it as a sole mechanism would produce a fair amount of overhead.)

## **17. Acknowledgments**

This document was initiated by Noel Chiappa, and much of the core philosophy came from him. We acknowledge the important contributions he has made to this work and thank him for his past efforts.

The author would like to start by thanking all the members of the core LISP group for their willingness to allow him to add himself to their effort, and for their enthusiasm for whatever assistance he has been able to provide.

He would also like to thank (in alphabetical order) Michiel Blokzijl, Peter Chiappa, Vina Ermagan, Dino Farinacci, Vince Fuller and Vasileios Lakafofosis for their review of, and helpful suggestions for, this document. (If I have missed anyone in this list, I apologize most profusely.)



A special thank you goes to Joel Halpern, who almost invariably, when asked, promptly returned comments on intermediate versions of this document. Grateful thanks go also to Darrel Lewis for his help with material on non-Internet uses of LISP, and to Dino Farinacci and Vince Fuller for answering detailed questions about some obscure LISP topics.

A final thanks is due to John Wrocklawski for the author's organizational affiliation, and to Vince Fuller for help with XML. This memo was created using the xml2rfc tool.

I would like to dedicate this document to the memory of my parents, who gave me so much, and whom I can no longer thank in person, as I would have so much liked to be able to.

## **18. IANA Considerations**

This document makes no request of the IANA.

## **19. Security Considerations**

This memo does not define any protocol and therefore creates no new security issues.

## **20. References**

### **20.1. Normative References**

- [AFI] IANA, , "Address Family Indicators (AFIs)",  
<http://www.iana.org/assignments/address-family-numbers> ,  
January 2011.
- [I-D.ermagan-lisp-nat-traversal]  
Ermagan, V., Farinacci, D., Lewis, D., Skriver, J., Maino,  
F., and C. White, "NAT traversal for LISP", [draft-ermagan-lisp-nat-traversal-05](#) (work in progress), February 2014.
- [I-D.farinacci-lisp-te]  
Farinacci, D., Kowal, M., and P. Lahiri, "LISP Traffic  
Engineering Use-Cases", [draft-farinacci-lisp-te-06](#) (work  
in progress), March 2014.
- [I-D.ietf-lisp-ddt]  
Fuller, V., Lewis, D., Ermagan, V., and A. Jain, "LISP  
Delegated Database Tree", [draft-ietf-lisp-ddt-01](#) (work in  
progress), March 2013.



[I-D.ietf-lisp-lcaf]

Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", [draft-ietf-lisp-lcaf-05](#) (work in progress), May 2014.

[I-D.ietf-lisp-perspective]

Chiappa, J., "An Architectural Perspective on the LISP Location-Identity Separation System", [draft-ietf-lisp-perspective-00](#) (work in progress), February 2013.

[I-D.ietf-lisp-sec]

Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", [draft-ietf-lisp-sec-06](#) (work in progress), April 2014.

[I-D.ietf-lisp-threats]

Saucez, D., Iannone, L., and O. Bonaventure, "LISP Threats Analysis", [draft-ietf-lisp-threats-10](#) (work in progress), July 2014.

[I-D.meyer-lisp-mn]

Farinacci, D., Lewis, D., Meyer, D., and C. White, "LISP Mobile Node", [draft-meyer-lisp-mn-10](#) (work in progress), January 2014.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, [RFC 768](#), August 1980.

[RFC0791] Postel, J., "Internet Protocol", STD 5, [RFC 791](#), September 1981.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", [RFC 2460](#), December 1998.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", [RFC 6830](#), January 2013.

[RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", [RFC 6831](#), January 2013.

[RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", [RFC 6832](#), January 2013.



- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", [RFC 6833](#), January 2013.
- [RFC6834] Iannone, L., Saucez, D., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning", [RFC 6834](#), January 2013.
- [RFC7215] Jakab, L., Cabellos-Aparicio, A., Coras, F., Domingo-Pascual, J., and D. Lewis, "Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations", [RFC 7215](#), April 2014.

## **[20.2. Informative References](#)**

- [Atkinson]  
R. Atkinson, , "Revised draft proposed definitions", RRG list message, Message-Id: 808E6500-97B4-4107- 8A2F-36BC913BE196@extremenetworks.com, 11 June 2007, <http://www.ietf.org/mail-archive/web/ram/current/msg01470.html> , .
- [Baran] P. Baran, , "On Distributed Communications Networks", IEEE Transactions on Communications Systems Vol. CS-12 No. 1, pp. 1-9 , March 1964.
- [Bibliography]  
J.N. Chiappa, , "LISP (Location/Identity Separation Protocol) Bibliography", <http://www.chiappa.net/~jnc/tech/lisp/LISPbiblio.html> , July 2013.
- [Chiappa] J.N. Chiappa, , "Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture", Personal draft (work in progress), 1999, <http://www.chiappa.net/~jnc/tech/endpoints.txt> , 1999.
- [Clark] D. D. Clark, , "The Design Philosophy of the DARPA Internet Protocols", Proceedings of the Symposium on Communications Architectures and Protocols SIGCOMM '88', pp. 106-114 , 1988.
- [CorasBGP]  
F. Coras, D. Saucez, L. Jakab, A. Cabellos-Aparicio, and J. Domingo-Pascual, , "Implementing a BGP-free ISP Core with LISP", Proceedings of the Global Communications Conference (GlobeCom), IEEE, pp. 2772-2778 , December 2012.





**[CorasCache]**

J. Kim, L. Iannone, and A. Feldmann, , "On the Cost of Caching Locator/ID Mappings", Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I (NETWORKING '11)', IFIP, pp. 367-378 , May 2011.

**[Future]** J.N. Chiappa, , "Potential Long-Term Developments With the LISP System", [draft-chiappa-lisp-evolution-00](#) (work in progress) (NOT EXISTING) , October 2012.

**[Heart]** F. E. Heart, R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden, , "The Interface Message Processor for the ARPA Computer Network", Proceedings AFIPS SJCC, Vol. 36, pp. 551-567 , 1970.

**[I-D.farinacci-lisp]**

Farinacci, D., "Locator/ID Separation Protocol (LISP)", [draft-farinacci-lisp-00](#) (work in progress), January 2007.

**[IEN19]** J. F. Shoch, , "Inter-Network Naming, Addressing, and Routing", IEN (Internet Experiment Note) 19 , January 1978.

**[Iannone]** L. Iannone and O. Bonaventure, , "On the Cost of Caching Locator/ID Mappings", Proceedings of the 3rd International Conference on emerging Networking EXperiments and Technologies (CoNEXT'07)', ACM, pp. 1-12 , December 2007.

**[Improvements]**

J.N. Chiappa, , "An Overview of On-Going Improvements to the LISP Location-Identity Separation System", [draft-chiappa-lisp-improvements-00](#) (work in progress) (NOT EXISTING) , September 2013.

**[Kim]** L. Iannone and O. Bonaventure, , "A Deep Dive Into the LISP Cache and What ISPs Should Know About It", Proceedings of the 3rd International Conference on emerging Networking EXperiments and Technologies (CoNEXT'07)', ACM, pp. 1-12 , December 2007.

**[LISP-TREE]**

L. Jakab, A. Cabellos-Aparicio, F. Coras, D. Saucez, and O. Bonaventure, , "LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System", IEEE Journal on Selected Areas in Communications', Vol. 28, No. 8, pp. 1332-1343 , October 2010.



- [NIC8246] A. McKenzie and J. Postel, , "Host-to-Host Protocol for the ARPANET", NIC 8246, Network Information Center, SRI International, Menlo Park, CA , October 1977.
- [NSAP] International Organization for Standardization, , "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", ISO Standard 7489.1984 , 1984.
- [RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, [RFC 826](#), November 1982.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, [RFC 1034](#), November 1987.
- [RFC1498] Saltzer, J., "On the Naming and Binding of Network Destinations", [RFC 1498](#), August 1993.
- [RFC1631] Egevang, K. and P. Francis, "The IP Network Address Translator (NAT)", [RFC 1631](#), May 1994.
- [RFC1812] Baker, F., "Requirements for IP Version 4 Routers", [RFC 1812](#), June 1995.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", [BCP 5](#), [RFC 1918](#), February 1996.
- [RFC1992] Castineyra, I., Chiappa, N., and M. Steenstrup, "The Nimrod Routing Architecture", [RFC 1992](#), August 1996.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", [RFC 3168](#), September 2001.
- [RFC3170] Quinn, B. and K. Almeroth, "IP Multicast Applications: Challenges and Solutions", [RFC 3170](#), September 2001.
- [RFC3272] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and X. Xiao, "Overview and Principles of Internet Traffic Engineering", [RFC 3272](#), May 2002.
- [RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", [RFC 4026](#), March 2005.



- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", [RFC 4033](#), March 2005.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", [BCP 107](#), [RFC 4107](#), June 2005.
- [RFC4116] Abley, J., Lindqvist, K., Davies, E., Black, B., and V. Gill, "IPv4 Multihoming Practices and Limitations", [RFC 4116](#), July 2005.
- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", [BCP 126](#), [RFC 4786](#), December 2006.
- [RFC4984] Meyer, D., Zhang, L., and K. Fall, "Report from the IAB Workshop on Routing and Addressing", [RFC 4984](#), September 2007.
- [RFC5110] Savola, P., "Overview of the Internet Multicast Routing Architecture", [RFC 5110](#), January 2008.
- [RFC5887] Carpenter, B., Atkinson, R., and H. Flinck, "Renumbering Still Needs Work", [RFC 5887](#), May 2010.
- [RFC6115] Li, T., "Recommendation for a Routing Architecture", [RFC 6115](#), February 2011.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", [RFC 6836](#), January 2013.
- [Saltzer] J. H. Saltzer, D. P. Reed, and D. D. Clark, , "End-To-End Arguments in System Design", ACM TOCS, Vol 2, No. 4, pp 277-288 , November 1984.
- [Saucez] D. Saucez, L. Iannone, and B. Donnet, , "A First Measurement Look at the Deployment and Evolution of the Locator/ID Separation Protocol", ACM SIGCOMM Computer Communication Review', Vol. 43 No. 2, pp. 37-43 , April 2013.

## [Appendix A](#). Glossary/Definition of Terms

{{Suggestion by editors: remove and use those from RFCs instead }}

- o EID, Enpoint Identifier: Originally defined as a name for an endpoint, one with purely identity semantics, and globally unique, and with syntax of relatively short fixed length. It is used in



the LISP work to mean the identifier of a node; it is the input to an EID->RLOC lookup in the mapping system; it is usually an IP address. The source and destination addresses of the innermost header in a LISP packet are usually EIDs.

- o RLOC, Routing Locator: a LISP-specific term meaning the locator associated with an entity identified by an EID; as such, it is often the output of an EID->RLOC lookup in the mapping system; it is usually an IP address, and of an ETR. The source and destination addresses of the outermost header in a LISP packet are usually RLOCs.
- o ITR, Ingress Tunnel Router: a LISP router at the border of a LISP site which takes user packets sent to it from inside the LISP site, encapsulates in a LISP header, and then sends them across the Internet to an ETR; in other words, the start of a tunnel from the ITR to an ETR.
- o ETR: Egress Tunnel Router: a LISP router at the border of a LISP site which decapsulates user packets which arrive at it encapsulated in a LISP header, and sends them on towards their ultimate destination; in other words, the end of the tunnel from an ITR to the ETR.
- o Neighbour ETR: Although an ITR and ETR may be separated by many actual physical hops, at the LISP level, they are direct neighbours; so any ETR which an ITR sends traffic to is a neighbour ETR of that ITR.
- o xTR: An xTR refers to a LISP router which functions both as an ITR and an ETR (which is typical), when the discussion involves packet flows in both directions through the router, which results in it alternately functioning as an ITR and then as an ETR.
- o Reachable; Reachability; Neighbour ETR Reachability: The ability of an ITR to be able to send packets to a neighbour ETR, or the property of an ITR to be able to send such packets.
- o Liveness: Whether a LISP node of any kind is up and operating, or not; or the property of a LISP node to be in such a state.
- o MR, Map Resolver: A LISP node to which ITRs send requests for mappings.
- o MS, Map Server: A LISP node with which ETRs register mappings, to indicate their availability to handle incoming traffic to the EIDs covered in those mappings.





- o Mapping System: The entire ensemble of data and mechanisms which allow clients - usually ITRs - to find mappings (from EIDs to RLOCs). It includes both the mapping database, and also everything used to gain access to it - the MRs, the indexing sub-system, etc.
- o Mapping Database: The term mapping database refers to the entire collection of EID-to-RLOC mappings spread throughout the entire LISP system. It is a subset of the mapping system.
- o Mapping Cache: A collection of copies of EID-to-RLOC mappings retained in an ITR; not the entire mapping database, but just the subset of it that an ITR needs in order to be able to properly handle the user data traffic which is flowing through it.
- o Indexing Sub-system: the entire ensemble of data and mechanisms which allows MRs to find out which ETR(s) hold the mapping for a given EID or EID block. It includes both the data on namespace delegations, as well as the nodes which hold that data, and the protocols used to interact with those nodes.
- o DDT Vertex; Vertex: a node (in the graph theory sense of the term) in the (abstract) LISP namespace delegation hierarchy.
- o DDT Server: an actual machine, which one can send packets to, in the DDT server hierarchy - which is, hopefully, a one-to-one projection of the LISP address delegation hierarchy (although of course a single DDT vertex may turn into several sibling servers). Some documents refer to these as DDT nodes but this document does not use that term, to prevent confusion with DDT vertex.
- o PITR: Proxy ITR; an ITR which is used for interworking between a LISP-speaking node or site, and legacy nodes or sites; in general, it acts like a normal ITR, but does so on behalf of LISP nodes which are receiving packets from a legacy node.
- o PETR: Proxy ETR; an ETR which is used for interworking between a LISP-speaking node or site, and legacy nodes or sites; in general, it acts like a normal ETR, but does so on behalf of LISP nodes which are sending packets to a legacy node.
- o RTR: Re-encapsulating Tunnel Router; a data plane 'anchor point' used by a LISP-speaking node to perform functions that can only be performed in the core of the network. One use is for LISP-speaking node behind a NAT device to send and receive traffic through the NAT device; see



- o Internet core: That part of the Internet in which there are no 'default' entries in routing tables, but where the routing tables hold entries for every single reachable destination in the Internet. (Sometimes referred to colloquially as the DFZ, or 'Default Free Zone'.)

## [Appendix B](#). Other Appendices

### [B.1](#). A Brief History of Location/Identity Separation

It was only gradually realized in the networking community that networks (especially large networks) should deal quite separately with the identity and location of a node; the distinction between the two was more than a little hazy at first.

The ARPANET had no real acknowledgment of the difference between the two. [[Heart](#)][[NIC8246](#)] The early Internet also co-mingled the two ([[RFC0791](#)]), although there was recognition in the early Internet work that there were two different things going on. [[IEN19](#)]

This likely resulted not just from lack of insight, but also the fact that extra mechanism is needed to support this separation (and in the early days there were no resources to spare), as well as the lack of need for it in the smaller networks of the time. (It is a truism of system design that small systems can get away with doing two things with one mechanism, in a way that usually will not work when the system gets much larger.)

The ISO protocol architecture took steps in this direction [[NSAP](#)], but to the Internet community the necessity of a clear separation was definitively shown by Saltzer. [[RFC1498](#)] Later work expanded on Saltzer's, and tied his separation concepts into the fate-sharing concepts of Clark. [[Clark](#)], [[Chiappa](#)]

The separation of location and identity is a step which has recently been identified by the IRTF as a critically necessary evolutionary architectural step for the Internet. [[RFC6115](#)] However, it has taken quite some time for this requirement to be generally accepted by the Internet engineering community at large, although it seems that this may finally be happening.

Unfortunately, although the development of IPv6 presented a golden opportunity to learn from this particular failing of IPv4, that design failed to recognize the need for separation of location and identity.



## **B.2. A Brief History of the LISP Project**

{{Suggestion by editors: remove that makes no sense in this document  
}}

The LISP system for separation of location and identity resulted from the discussions of this topic at the Amsterdam IAB Routing and Addressing Workshop, which took place in October 2006. [[RFC4984](#)]

A small group of like-minded personnel from various scattered locations within Cisco, spontaneously formed immediately after that workshop, to work on an idea that came out of informal discussions at the workshop. The first Internet-Draft on LISP appeared in January, 2007, along with a LISP mailing list at the IETF. [[I-D.farinacci-lisp](#)]

Trial implementations started at that time, with initial trial deployments underway since June 2007; the results of early experience have been fed back into the design in a continuous, ongoing process over several years. LISP at this point represents a moderately mature system, having undergone a long organic series of changes and updates.

LISP transitioned from an IRTF activity to an IETF WG in March 2009, and after numerous revisions, the basic specifications moved to becoming RFCs at the start of 2013 (although work to expand and improve it, and find new uses for it, continues, and undoubtedly will for a long time to come).

## **B.3. Old LISP 'Models'**

LISP, as initially conceived, had a number of potential operating modes, named 'models'. Although they are now obsolete, one occasionally sees mention of them, so they are briefly described here.

- o LISP 1: EIDs all appear in the normal routing and forwarding tables of the network (i.e. they are 'routable'); this property is used to 'bootstrap' operation, by using this to load EID->RLOC mappings. Packets were sent with the EID as the destination in the outer wrapper; when an ETR saw such a packet, it would send a Map-Reply to the source ITR, giving the full mapping.
- o LISP 1.5: Similar to LISP 1, but the routability of EIDs happens on a separate network.
- o LISP 2: EIDs are not routable; EID->RLOC mappings are available from the DNS.



- o LISP 3: EIDs are not routable; and have to be looked up in a new EID->RLOC mapping database (in the initial concept, a system using Distributed Hash Tables). Two variants were possible: a 'push' system, in which all mappings were distributed to all ITRs, and a 'pull' system in which ITRs load the mappings they need, as needed.

#### **B.4. The ALT Mapping Indexing Sub-system**

LISP initially used an indexing sub-system called ALT. [\[RFC6836\]](#) ALT re-purposed a number of existing mechanisms to provide an indexing system, which allowed an experimental LISP initial deployment to become operational without having to write a lot of code, ALT was relatively easily constructed from basically unmodified existing mechanisms; it used BGP running over virtual tunnels using GRE.

ALT proved to have a number of issues which made it unsuitable for large-scale use, and it has now been superseded by DDT. A complete list of these is not possible here, but the issues mostly were of two kinds: technical issues which would have arisen at large scale, and practical operational issues which appeared even in the experimental deployment.

The biggest operational issues was the effort involved in configuring, and maintain the configuration, of the virtual tunnels over which ALT ran (including assigning the addresses for the ends, etc); also, managing the multiple disjoint routing tables required was difficult and confusing (even for those who were very familiar with ALT). Debugging faults in ALT was also difficult; and finally, because of ALT's nature, administrative issues (who pays for what, who controls what, etc) were problematic.

However, ALT would have had significant technical issues had it been used at a larger scale.

The most severe (and fundamental) issue was that since all traffic on the ALT had to transit the 'root' of the ALT tree, those locations would have become traffic 'hot-spots' in a large scale deployment.

In addition, optimal performance would have required that the ALT overall topology be restrained to follow the EID namespace allocation; however, it was not clear that this was feasible. In any event, even optimal performance was still less than that in alternatives. The ALT was also very vulnerable to misconfiguration.

See [\[LISP-TREE\]](#) for more about these issues: the basic structure and operation of DDT is identical to that of TREE, so the conclusions drawn there about TREE's superiority to ALT apply equally to DDT.





In particular, the big advantage of DDT over the ALT, in performance terms, is that it allows MRs to interact directly with distant DDT servers (as opposed to the ALT, which always required mediation through intermediate servers); caching of information about those distant servers allows DDT to make extremely effective use of this capability.

The ALT did have some useful properties which its replacement, DDT, did not, e.g. the ability to forward data directly to the destination, over the ALT, when no mapping was available yet for the destination. However, these were minor, and heavily outweighed by its problems.

A recent study, [[Saucez](#)], measured actual resolution times in the deployed LISP network during the changeover from ALT to DDT, allowing direct comparison of the performance of the two systems. The study took measurements from a variety of locations in the Internet, with respect to a number of different target EIDs. The results indicate that the performance was almost identical; there was more variance with DDT (perhaps due to the effects of caching), but the greatly improved scalability of DDT as compared to ALT made that effect acceptable.

#### **B.5. Early NAT Support**

The first mechanism used by LISP to support operation through a NAT device, described here, has now been superseded by the more general mechanism proposed in [[I-D.ermagan-lisp-nat-traversal](#)]. That mechanism is, however, based heavily on this mechanism. The initial mechanism had some serious limitations, which is why that particular form of it has been dropped.

First, it only worked with some NATs, those which were configurable to allow inbound packet traffic to reach a configured host. The NAT had to be configured to know of the ETR.

Second, since NATs share addresses by using ports, it was only possible to have a single LISP node behind any given NAT device. That is because LISP expects all incoming data traffic to be on a specific port, so it was not possible to have multiple ETRs behind a single NAT (which normally would have only one global IP address to share). Even looking at the source host and port would not necessarily help, because some source ITR could be sending packets to both ETRs, so packets to either ETR could also have the identical source host/port. In short, there was no way for a NAT with multiple ETRs behind it to know which ETR the packet was for.



To support operation behind a NAT, there was a pair of new LISP control messages, LISP Echo-Request and Echo-Reply, which allowed the ETR to discover its temporary global address. The Echo-Request was sent to the configured Map-Server, and it replied with an Echo-Reply which included the source address from which the Echo Request was received (i.e. the public global address assigned to the ETR by the NAT). The ETR could then insert that address in any Map-Reply control messages which it sent to correspondent ITRs.

Echo-Request and Echo-Reply have been replaced by Info-Request and Info-Reply in the replacement, [NAT-Traversal], where they perform very similar functions; the main change is the addition of the {{{xxx - probably the port, etc to allow multiple XTRs behind a NAT}}}.

#### Authors' Addresses

Albert Cabellos-Aparicio (Ed.)  
Technical University of Catalonia  
C/Jordi Girona, s/n  
BARCELONA 08034  
Spain

Email: [jacabello@ac.upc.edu](mailto:jacabello@ac.upc.edu)

Damien Saucez (Ed.)  
INRIA  
2004 route des Lucioles BP 93  
06902 Sophia Antipolis Cedex  
France

Email: [damien.saucez@inria.fr](mailto:damien.saucez@inria.fr)

