LISP Working Group
Internet-Draft
Intended status: Informational
Expires: April 24, 2014

J. N. Chiappa
Yorktown Museum of Asian Art

October 21, 2013

An Architectural Introduction to the LISP Location-Identity Separation System
draft-ietf-lisp-introduction-03

Abstract

LISP is an upgrade to the architecture of the IP internetworking system, one which separates location and identity properties (previously intermingled in IP addresses). This document is an introductory overview of the entire LISP system, and focuses on describing the major concepts and functional sub-systems of LISP, and the interactions between them.

Table of Contents

1. Prefatory Note

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. [Perspective]

This 'Architectural Introduction' document is primarily intended for those who 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 also to give the reader a choice as to how much they wish to know about LISP. 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, "Examples of Operation". People who are going to go on and read the protocol specifications (perhaps to implement LISP) should read the entire document.

Note: 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 (').

- Name: In this document, and in much of computer science, a 'name' simply refers to an identifier for an object or entity. Names have both semantics (meaning) and syntax (form). [RFC1498]
- Namespace: A group of "names" with matching semantics and syntax; they usually, but not always, refer to members of a class of identical objects.
- Mapping: In this document, a connection (or binding, to use the computer science term) between two names, one in each of two namespaces.

- 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.
- 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. It was deliberately chosen for use in this document precisely because its definition is not fixed, and therefore unlikely to cause erroneous images in the minds of readers.
- 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]
- 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]
- IPvN: IPv4 ([RFC791]) or IPv6 ([RFC2460]); the two are so similar, in fundamental architecture, that in much discussion about their capabilities, limitations, etc statements about the apply equally to both, and to continually say 'IPv4 and IPv6' quickly becomes tedious.
- Address: In this document, and in current "IPvN" and similar networking suites, a "name" which has mixed semantics, in that it includes both identity ('who') and location ('where') semantics. [Atkinson]
- Address Block, Block: A contiguous section of a namespace, usually IPvN addresses; for the latter, it will normally be on a bit boundary, using the standard 'prefix/length' selection indication.
- Identifier: Here, and in current networking discussions, a "name" which has purely identity semantics.
- Locator: Originally defined as a "name" with only location semantics, and one that was not necessarily carried in every packet (as was widely assumed of "addresses") [RFC1992], it is now generally taken, including here, to mean a "name" with purely location semantics.
- Site: A collection of hosts, routers and networks under a single administrative control.
- LISP site: A single node, or a set of network elements in an edge network under the administrative control of a single organization; they are separated from the rest of the network by "LISP routers".
- LISP node: A IPvN "node" which has been enhanced with LISP functionality; generally this means it can process some subset of LISP control plane traffic.
- LISP router: A IPvN "switch" which has been enhanced with LISP functionality; a LISP node which can forward user traffic.
- LISP host: A IPvN 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]) (A more detailed history of this evolution is in Appendix B.1, "A Brief History of Location/Identity Separation".)

At the moment, 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 recently been identified by the IRTF as a necessary evolutionary architectural step for the Internet. [RFC6115]

The on-going LISP project is an attempt to provide a viable path towards this separation. (A brief history of the LISP project can be found in Appendix B.2, "A Brief History of the LISP Project".)

As an add-on to a large existing system, it has had to make certain compromises. (For a good example, see [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 [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 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

The deployment philosophy was a major driver for much of the design of LISP: to some degree of the architecture, and to a very large measure, the engineering.

Experience over the last several decades has shown that having a viable 'deployment model' for a new design is absolutely key to the success of that design. 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 or will not be successfully deployed (for whatever factors), it is useless.

This absolute primacy of what is hoped is a viable deployment model is what has lead to some painful compromises in the design; and the extreme focus on a viable deployment model (including economics) is one of the key design guides of LISP.

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

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

One key part of reducing the cost of a new design is to absolutely 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 lower the pain (and thus 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 (see Section 15.1, "LISP Deployment Needs", for more).

LISP also initially re-uses, where-ever possible, existing protocols. The 'initially' must be stressed - careful attention has also long been paid to the long-term future (see [Future]), and larger changes become feasible as deployment increases.

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 "IPvN" "addresses".

The separation into names with purely location and purely identity semantics is usually - but not necessarily - not perfect, for reasons which are driven by the deployment philosophy (above), and explored in more detail elsewhere (in [Perspective], Section "Namespaces-EIDs-Residual").

6.1. Basic Approach

In LISP, the first key concept is that nodes have both an 'identifier' (a name which serves only to provide a persistent handle for the node), called an "EID" (short for 'endpoint identifier'), and an associated 'locator' (a name which says _where_ the node is, in the network's connectivity structure), called an "RLOC" (short for 'routing locator').

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_.

EIDs name nodes - or rather, their end-end communication entities (see [Chiappa] for more). RLOC(s), on the other hand, name interfaces, i.e. places to which the system of routers sends packets. (These will usually be on the "LISP routers", in the early stages of LISP deployment; see below for more.)

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 IPvN 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 minimized 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 IPvN addresses for both kinds of names is already built in, which is expected to greatly increase the long-term benefits, flexibility, and power of the LISP "mapping" layer. [AFI][LCAF]

6.2. Basic Functionality

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 some of that additional information.

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:

(to be added)

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 address; in LISP, this is the EID of the ultimate destination. 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) will normally be 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 mappings get them from the "mapping system", which is a collection of sub-systems through which clients can find and obtain mappings. (The mapping system will be discussed in more detail below, in Section 8.2, "Control Plane - Mapping System Overview" and Section 13, "The Mapping System".)

Mappings are are normally distributed via a 'pull' mechanism; in other words, they are generally not pre-loaded, but requested on demand. Once obtained by an ITR, they are cached by the ITR, for performance reasons.

Extensive studies, including large-scale simulations driven by lengthy recordings of actual traffic at several major sites, have been performed to verify that this 'pull and cache' approach is viable, in practical engineering terms. (This subject will be discussed in more detail in Section 12.9, "xTR Mapping Cache Performance", below, including references to the studies.)

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 such interoperation, a number of mechanisms have been designed. One approach uses proxy LISP routers, called "PITRs" (proxy ITRs) and "PETRs" (proxy ETRs), to provide LISP functionality during interaction with legacy hosts. Another approach uses a router with combined LISP and NAT ([RFC1631]) functionality, named a LISP-

NAT.

(See Section 15.2.1, "Proxy LISP Routers", and Section 15.2.2, "LISP-NAT", respectively, for details of each, and their respective advantages and disadvantages.)

6.5. Security in LISP

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 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. [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 [Perspective], Section "Security", where it is laid out in some detail.

7. Initial Applications

{{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 imporant 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 [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.

The problem is simple: for the global routing to scale, addresses need to be aggregated; i.e. things which are close in the overall network's connectivity need to have closely related addresses (so-called "provider aggregatable" addresses). [RFC4116] However, if this principle is followed, it means that when an entity switches providers (i.e. it moves to a different 'place' in the network), it has to re-number, a painful undertaking. [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

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 addressses - 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

{{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 forseen 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 [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

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

Mobility is yet another place where separation of location and identity is obviously 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 inaccessability 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 infrastucture which only supports the other type.

While the machinery of LISP may seem too heavy-weight to be good for such a mundane use, this is not intended as a 'sole use' case for deployment of LISP. Rather, it is something which, if LISP is being deployed anyway (for its other advantages), is an added benefit that one gets 'for free'.

7.7. Virtual Private Networks

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?}}

This includes support of VPN's for segmentation and multi-tenancy (i.e. a spatially separated private VPN whose components are joined together using the public Internet as a backbone).

7.8. Local Uses

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; other non-IP EID types such as local network 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

LISP has only two major functional sub-systems - the collection of LISP "packet switches" (the xTRs), which form the 'data plane' of LISP; and the "mapping system", the most important part of the 'control plane', which manages 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 (Sections Section 12, "xTRs", and Section 13, "The Mapping System", respectively).

8.1. Data Plane - xTRs Overview

xTRs are packet switches which 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 [RFC768] (for reasons to be explained below, in Section 12.2, "UDP Encapsulation Details"), along with an additional outer IPvN 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 now-normal packets to the ultimate destination.

Control plane functions in ITRs include: asking for {EID->RLOC} mappings via request control messages (Map-Request packets); handling the returning reply control messages (Map-Reply packets), which contain the requested information; managing the local "mapping cache" of "mappings"; checking for the "reachability" and "liveness" of their neighbour ETRs; and checking for outdated mappings and requesting updates.

In the ETR, control plane functions include participating in the reachability and liveness function (see Section 16.4, "Verifying ETR Liveness"); interacting with the mapping sub-system to let it know what mapping this ETR can provide (see Section 8.2.2, "Interface to the Mapping System"); and answering requests from ITRs for those mappings (ditto).

8.1.1. Mapping Cache Performance

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.

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

Obviously, these studies are all snapshots of a particular point in time, and as the Internet continues its life-cycle they will increasingly become out-dated. However, they are useful because they provide an insight into how well LISP can be expected to perform, and scale, over time.

8.2. Control Plane - Mapping System Overview

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.

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 have more than one RLOC, or may change its RLOC(s), while keeping its basic identity.

Also, in general, throughout LISP, anyplace a name (EID, RLOC, etc) appears in a control packet, the packet format also includes an Address Family Identifier (AFI) for that name. [AFI] The inclusion of the AFI allows LISP (and in particular, the mapping system interface, as embodied in those control packets) a great deal of flexibility. (See [Perspective], Section "Namespaces" for more on this.)

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

The "mapping system" is actually split into what are effectively three major functional sub-systems (although the latter two are closely integrated, and appear to most entities in the LISP system as a single sub-system).

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 -> 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

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 client-interface sub-system, which is provided by LISP nodes called Map-Resolvers (MRs) and Map-Servers (MSs).

ITRs send request control messages (Map-Request packets) to an MR. (This interface is probably the most important standardized interface in LISP - it is the key to the entire system.)

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 contents of all mappings for those EID namespace blocks which have been delegated to it.

The ETR then formulates reply control messages (Map-Reply packets), which are sent to the ITR. The details of the indexing sub-system are thus hidden from the ITRs.

(Note that in some cases, it is desirable for the MS to reply on behalf of the ETR, in so-called 'proxy' mode. This behaviour can be selected when the ETR registers with the MR, described immediately below.)

Similarly, the client interface to the indexing system from an ETR's point of view is through LISP nodes called Map-Servers (MSs). ETRs send registration control messages (Map-Register packets) to an MS, which makes the information about the mappings which the ETR indicates it is authoritative for available to the indexing sub-system.

The MS formulates a reply control message (the Map-Notify packet), which confirms the registration, and is returned to the ETR. The details of the indexing sub-system are thus likewise hidden from the 'ordinary' 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 clients - as has happened once already (see below).

8.2.3. Indexing Sub-system

The current indexing sub-system is the Delegated Database Tree (DDT), which is very similar to DNS ([DDT], [RFC1034]). Unlike DNS, the actual mappings are not handled by DDT; DDT, as the indexing sub-system, merely identifies the ETRs which hold the actual mappings.

DDT replaces an earlier indexing sub-system, ALT (Appendix B.4, "The ALT Mapping Indexing Sub-system"); this swap 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 fairly simple: like DNS, in DDT the delegation of the EID namespace ([Perspective], Section "Namespaces-XEIDs") is instantiated as a "delegation hierarchy", a tree of "DDT vertices", starting with the 'root' DDT vertex. Each vertex is responsible for a "block" of the EID namespace.

The 'root' vertex is reponsible for the entire namespace; any DDT vertex can 'delegate' part(s) of its block of the namespace to child DDT vertex(s). The child vertex(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 vertex would be used for reasons of performance, reliability and robustness. Obviously, all the servers which instantiate a particular vertex in the tree have to have identical data about that vertex; if they do not, when a Map-Request is sent to one that does not have consistent information with its other sibling(s), incorrect results will be returned.

Also, although the delegation hierarchy is a strict tree, a single DDT server could be authoritative for more than one block of the EID namespace (i.e. it could be a server for more than one vertex).

Eventually, leaf vertices in the delegation hierarchy statically delegate EID namespace blocks to MS's, which are DDT terminal servers; i.e. a leaf of the tree is reached when the delegation points to an MS instead of to another DDT vertex. {{Straighten out.}}

The MS is in direct communication with the ETR(s) which both i) are authoritative for the mappings for that block, and ii) handle traffic to all nodes in that block of EID namespace.

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 "vertices" of the LISP "delegation hierarchy" tree, discovering (by querying the servers for information about DDT vertices) the chain of delegations which cover that EID. Eventually it is directed to an MS, which is the 'door' to an ETR which is authoritative for that EID.

Also, again like DNS, MRs cache information they receive about the delegations in the delegation tree. This means that once an MR has been in operation for 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 mappping.

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 namespace 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. 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. 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

This case follows the processing of a typical user packet (for instance, a normal TCP data or acknowledgment packet associated with an already-open TCP connection) - i.e. not the first packet sent from a given source to a given destination - as it makes its way from the original source host to the ultimate destination.

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 IPvN addresses, this lookup usually uses the usual IPvN 'longest prefix match' 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, "UDP Encapsulation Details", 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 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

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, "Basic Functionality").

The overall processing is shown below, in Figure 2:

```
                        Mapping System

                    -----            -----
                   |     |    3     |     |
     Map Resolver  |     | -------> |     |  Map Server
                   |     |          |     |
                    -----            -----
                      ^                |
         Key:         |                |
                      |                |
     -- = Control     |                |
     == = Data        |                |
                   2  |       5        |  4
                      |      ---       |
     Host A           |    /     \     |              Host B
                      |  |_       \    V
     -----          -----          \ -----          -----
    |     |   1    |     |    6     |     |   7    |     |
    |     | =====> | ITR | =======> | ETR | =====> |     |
    |     |        |     |          |     |        |     |
     -----          -----            -----          -----
```

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.

Often 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 queued, or perhaps it was forwarded using a PITR. (Section 6.4, "Interworking With Non-LISP-Capable Endpoints")

10. Part II

11. Design Approach

Before describing LISP's components in more detail below, it it 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 in [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, "Data Plane - xTRs Overview"), xTRs are the basic data-handling nodes in LISP, and, as such, form the LISP data plane - although of necessity they are also involved in some control plane functions. This section explores some advanced topics related to xTRs.

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 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 an ultimate 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 filter out 'unknown' protocols, so adopting a non-UDP encapsulation would have made the initial deployment of LISP harder.

The UDP source port in the encapsulated packet is a 5-way hash of the original source and ultimate 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 {source, source port, destination, destination port} tuple flow along a single path, and 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] In most exising 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

LISP provides a multiplexed channel in the encapsulation header. It is mostly (but not entirely) used for control purposes. (See [Perspective], Section "Architecture-Piggyback" for a longer discussion of the architectural implications of performing control functions with data traffic.)

The general concept is that the header starts with an '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 encapsulation header.

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 mappping. [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).

At the moment version numbers are manually assigned, and ordered.

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 wishes to ensure that an ETR is up, and "reachable", sends a nonce to that ETR, carried in the encapsulation header; when that ETR (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, "Verifying ETR Reachability", for more about Echo Nonces.)

12.3.3. Instances

Another use of these header fields is for 'Instances' - basically, support for VPN's across backbones. [RFC4026] Since there is only one destination UDP port used for carriage of user data packets, and the source port is used for multiplexing (above), there is no other way to differentiate among different destination address namespaces (which are often overlapped in VPNs).

12.4. Probing

RLOC-Probing (see [RFC6830], Section 6.3.2. "RLOC-Probing Algorithm" for details) is a mechanism method that an ITR can use to determine with certainty that an ETR is up and reachable from the ITR. As a side-benfit, it gives a rough RTT estimates.

It is quite a simple mechanism - an ITR simply 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 and Map-Reply 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.

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. However, it has the advantages of providing information quickly (a single RTT), and being a simple, direct, robust way of doing so.

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.

A further observation is that unless what are likely unreasonable amounts of RLOC Probing are being done, Echo Nonce will generally provide faster notification of loss of reachability (unless there is little or no bi-directional traffic between the ITR and ETR). {{ENs help reduce the amount of probing when both are in use}}

12.5. Mapping Lifetimes and Timeouts

Mappings come with a Time-To-Live, which indicate how long the creator of the mapping expects them to be useful for. The TTL may also indicate that the mapping should not be cached at all, or it can indicate that it has no particular lifetime, and the recipient can chose how long to store it.

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

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. (See further discussion of the security implications of this in [Perspective], Section "Security-xTRs".)

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

LISP provides an optional mechanism to secure the obtaining of mappings by an ITR. [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-Rquests also. It does assume that the channel from the MR to the MS is secure (otherwise an attacker could obtain the OTK from the Map-Request and use it to forge a reply).

12.9. xTR Mapping Cache Performance

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

For a comprehensive survey of this work, see [Perspective], Section "Mapping Cache Performance", and the references; full details are too lengthy to include here.

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 satisified 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 1x10-4; again, definitely viable, and in line with the results of the other studies.

13. The Mapping System

As discussed already in Section 8.2, "Control Plane - Mapping System Overview", 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.

RFC 1034 ("DNS Concepts and Facilities") 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, "Interface to the Mapping System", both of the inferfaces 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 precaution has already allowed the mapping system to be upgraded during LISP's evolution, when ALT was replaced by DDT.)

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

13.1.1. Map-Request Messages

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. {{Not quite right, Dino will clarify. - Also two sets of RLOCs.}} Multiple RLOCs are included to ensure that at least one is in a form which will allow the reply to be returned to the requesting ITR, and the source EID is used for a variety of functions, including 'gleaning' (see Section 12.6, " Mapping Gleaning in ETRs").

Finally, the message includes a long nonce, for simple, efficient protection against offpath attackers (see [Perspective], Section "Security-xTRs" for more), and a variety of other fields and control flag bits.

13.1.2. Map-Reply Messages

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, "Control Plane - Mapping System Overview"), and multicast priority and weight fields (see Section 14, "Multicast Support in LISP" for more about multicast support in LISP).

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. {{Look at how probe is handled, do similar here - take out 'not xxx', say what they are.}} 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, is rate-limited.

13.1.3. Map-Register and Map-Notify Messages

The Map-Register message contains authentication information, and a number of 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, "Mapping Versioning"), and a number of RLOCs and their AFIs.

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 (see [Mobility]).

Map-Notify messages have the exact same contents as Map-Register messages; they are purely acknowledgements (although planned LISP functionality extensions may give them other functions as well).

The entire interaction can be 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]).

{{Deregistering??}}

13.2. The DDT Indexing Sub-system

As previously mentioned in Section 8.2.3, "Indexing Sub-system", the "indexing sub-system" in LISP is currently the DDT system.

The overall functioning is conceptually fairly simple; an MR which needs a "mapping" starts at a server for the root "DDT vertex" (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 (responsible?) for the block of EID namespace 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 namespace.

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; see Section 13.4, "Security of the DDT Indexing Sub-system", for more.

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 is has not yet asked about - probably only the last stage in a delegation to a 'leaf' MS.

As describe below (Section 13.6, "Performance of the Mapping System"), an extensive modeling and performance evaluation has verified that DDT provides acceptable performance, as well as scalability. [LISP-TREE]

13.2.1. Map-Referral Messages

Map-Referral messages look almost identical to Map-Reply messages, except that the RLOCs potentially name either i) the DDT servers for other DDT vertices (children in the delegation tree), or ii) terminal MSs.

13.3. Reliability via Replication

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.

{{About replication - we don't talk about how that rep occurs}}
{{Reliablity through rep is much sturdier - provide good ref}}

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. {{Not just DDT, other places as well.}}

13.4. Security of the DDT Indexing Sub-system

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.

See [Perspective], Section "Security-Mappings" for more.

13.5. Extended Capabilities

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. {{Policy-based?}}

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 ([ALT]), 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], in particular Section V, "Mapping System Comparison".)

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.

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

Multicast ([RFC3170], [RFC5110]) , since LISP is all about separating identity from location, and although a multicast group in some sense has an identity, it certainly does not have _a_ location.

{{Say something about sources.}}

Multicast is am 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 (although they may, after future improvements in multicast support in LISP, have multiple RLOCs); also, since they usually refer to collections of entities, they aren't really EIDs either.

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], Section "Multicast", for more).

14.2. Initial Multicast Support in LISP

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.

In the current simple operating mode (covered 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 lumped 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 joint 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) etc. 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

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, "Maximize Re-use of Existing Mechanism"), 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.

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, "Interworking With Non-LISP-Capable Endpoints") to allow interoperation of LISP sites with so-called 'legacy' sites which are not running LISP (yet).

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

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 hosts (for cases where a LISP node cannot send packets directly to such hosts, without encapsulation).

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.

This technique obviously has 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. {{Check on text elsewhere for effects on routing table size, specifically advertizement of large blocks.}}

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 to minimize the number of new announced routes added by this technique.) {{BGP tools can be used to restrict the direction and scope of these advertisements.}}

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.}}

{{Maybe reword, as we changed the target section.}} As mentioned previously (Section 12.1, "When to Encapsulate"), 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.

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 chose 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. However, this is not advised: as mentioned, it is easy to tell if something is a legacy destination.

15.2.2. LISP-NAT

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]

There are two main cases where this might apply:
- Sites using non-routable global addresses
- Sites using private addresses [RFC1918]

15.3. Use Through NAT Devices

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, "Early NAT Support". A more recent proposed mechanism, which avoids those limitations, is described in [Improvements], Section "Improved NAT Support".

15.4. LISP and Core Internet Routing

One of LISP's original motivations was to try and 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. {{Give ref for why large rout tables bad.}}

{{Does applications make forward ref to this section?}}

In covering this topic, one must recognize that conditions in various stages of LISP deployment (in terms of ubiquity) will have a large influence. [Deployment] 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:

- Using LISP for traffic engineering instead of BGP

- Aggregation of smaller PI sites into a single PITR 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.

{{Add xref to text moved to "Improvments" document.}}

16. Fault Discovery/Handling

The structure of LISP gives rise to a moderate number of 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) [RFC826], 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:

- 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)
- When the ETR traffic is being sent to is still an ETR, but no longer a valid ETR for that EID
- When the ETR traffic is being sent to is no longer an ETR
- {{No longer an ETR, but still a LISP node - another case to consider.}}

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 (if, perhaps not in the most expeditious, or 'clean' way - they are, after all, very unlikely to happen). {{Make less run on, easier to understand.}}

ETRs can easily detect cases where this happpens, after they have un-wrapped 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 Unreachble 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, "Routing Locator Reachability" 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

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.

Note that the two terms - "liveness" and "reachability" - are _not_ synonmous (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. {{Really property of path - if only one path, property of pair, otherwise of path.}}

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 in a way which is economic, robust and fast. Two out of the three are usually not to hard, but all three at the same time - as is highly desirable for this particular issue - are harder.

In line with the LISP design philosophy ([Perspective], Section "Design-Theoretical"), this problem is attacked not with a single mechanism (which would have a hard time meeting all those three goals simultaneously), but with a collection of simpler, cheaper mechanisms, which collectively will usually meet all three.

They are reliance on the underlying routing system (which can of course only reliably provide a negative reachabilty 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 direct 'pinging', in the case where no positive echo is returned.

(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

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 Lakafosis 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)", Address Family Numbers, January 2011, <http://www.iana.org/assignments/address-family-numbers>.

[RFC768] J. Postel, "User Datagram Protocol", RFC 768, August 1980.

[RFC791] J. Postel, "Internet Protocol", RFC 791, September 1981.

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

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

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

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

[RFC6833] V. Fuller and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, January 2013.

[RFC6834] L. Iannone, D. Saucez, and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning", RFC 6834, January 2013.

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

[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), September 2013.

[DDT] V. Fuller, D. Lewis, and D. Farinacci, "LISP Delegated Database Tree", draft-ietf-lisp-ddt-01 (work in progress), March 2013.

[LISP-SEC] F. Maino, V. Ermagan, A. Cabellos-Aparicio, D. Saucez, and O. Bonaventure, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-04 (work in progress), October 2012.

[NAT-Traversal] V. Ermagan, D. Farinacci, D. Lewis, J. Skriver, F. Maino, and C. White, "NAT traversal for LISP", draft-ermagan-lisp-nat-traversal-03 (work in progress), March 2013.

[Mobility] D. Farinacci, V. Fuller, D. Lewis, and D. Meyer, "LISP Mobility Architecture", draft-meyer-lisp-mn-08 (work in progress), April 2012.

[Deployment] L. Jakab, A. Cabellos-Aparicio, F. Coras, J. Domingo-Pascual, and D. Lewis, "LISP Network Element Deployment Considerations", draft-ietf-lisp-deployment-09 (work in progress), July 2013.

[Threats] D. Saucez, L. Iannone, and O. Bonaventure, "LISP Threats Analysis", draft-ietf-lisp-threats-08 (work in progress), October 2013.

[LCAF] D. Farinacci, D. Meyer, and J. Snijders, "LISP Canonical Address Format (LCAF)", draft-ietf-lisp-lcaf-03 (work in progress), September 2013.

[LISP-TE] D. Farinacci, P. Lahiri, and M. Kowal, "LISP Traffic

Engineering Use-Cases", draft-farinacci-lisp-te-03 (work in progress), July 2013.

20.2. Informative References

[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.

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

[RFC826] D. Plummer, "Ethernet Address Resolution Protocol", RFC 826, November 1982.

[RFC1034] P. V. Mockapetris, "Domain Names - Concepts and Facilities", RFC 1034, November 1987.

[RFC1498] J. H. Saltzer, "On the Naming and Binding of Network Destinations", RFC 1498, (Originally published in: 'Local Computer Networks', edited by P. Ravasio et al., North-Holland Publishing Company, Amsterdam, 1982, pp. 311-317.), August 1993.

[RFC1631] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)", RFC 1631, May 1994.

[RFC1812] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC1918] Y. Rekhter, R. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address Allocation for Private Internets", RFC 1918, February 1996.

[RFC1992] I. Castineyra, J. N. Chiappa, and M. Steenstrup, "The Nimrod Routing Architecture", RFC 1992, August 1996.

[RFC3168] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3170] B. Quinn and K. Almeroth, "IP Multicast Applications: Challenges and Solutions", RFC 3170, September 2001.

[RFC3272] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and Principles of Internet Traffic Engineering", RFC 3272, May 2002.

[RFC4026] L. Andersson and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, March 2005.

[RFC4033] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4107] S. Bellovin and R. Housley, "Guidelines for Cryptographic Key Management", RFC 4107, June 2005.

[RFC4116] J. Abley, K. Lindqvist, E. Davies, B. Black, and V. Gill, "IPv4 Multihoming Practices and Limitations", RFC 4116, July 2005.

[RFC4786] J. Abley and K. Lindqvist, "Operation of Anycast Services", RFC 4786, December 2006.

[RFC4984] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing", RFC 4984, September 2007.

[RFC5110] P. Savola, "Overview of the Internet Multicast Routing Architecture", RFC 5110, January 2008.

[RFC5887] B. Carpenter, R. Atkinson, and H. Flinck, "Renumbering Still Needs Work", RFC 5887, May 2010.

[RFC6115] T. Li, Ed., "Recommendation for a Routing Architecture", RFC 6115, February 2011.

(Perhaps the most ill-named RFC of all time; it contains nothing that could truly be called a 'routing architecture'.)

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

[LISP0] D. Farinacci, V. Fuller, and D. Oran, "Locator/ID Separation Protocol (LISP)", draft-farinacci-lisp-00 (work in progress), January 2007.

[Future] J. N. Chiappa, "Potential Long-Term Developments With the LISP System", draft-chiappa-lisp-evolution-00 (work in progress), October 2012.

[Baran] P. Baran, "On Distributed Communications Networks", IEEE Transactions on Communications Systems Vol. CS-12 No. 1, pp. 1-9, March 1964.

[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>.

[Clark] D. D. Clark, "The Design Philosophy of the DARPA Internet Protocols", in 'Proceedings of the Symposium on Communications Architectures and Protocols SIGCOMM '88', pp. 106-114, 1988.

[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.

[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 1970 SJCC, Vol. 36, pp. 551-567.

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

[Kim] J. Kim, L. Iannone, and A. Feldmann, "A Deep Dive Into the LISP Cache and What ISPs Should Know About It", in 'Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I (NETWORKING '11)', IFIP, pp. 367-378, May 2011.

[CorasCache] F. Coras, A. Cabellos-Aparicio, and J. Domingo-Pascual, "An Analytical Model for the LISP Cache Size", in 'Proceedings of the 11th International IFIP TC 6 Networking Conference: Part I', IFIP, pp. 409-420, May 2012.

[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", in 'IEEE Journal on Selected Areas in Communications', Vol. 28, No. 8, pp. 1332-1343, October 2010.

[Saucez] D. Saucez, L. Iannone, and B. Donnet, "A First Measurement Look at the Deployment and Evolution of the Locator/ID Separation Protocol", in 'ACM SIGCOMM Computer Communication Review', Vol. 43 No. 2, pp.

37-43, April 2013.

[CorasBGP] F. Coras, D. Saucez, L. Jakab, A. Cabellos-Aparicio, and J. Domingo-Pascual, "Implementing a BGP-free ISP Core with LISP", in 'Proceedings of the Global Communications Conference (GlobeCom)', IEEE, pp. 2772-2778, December 2012.

[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>.

[Bibliography] J. N. Chiappa (editor), "LISP (Location/Identity Separation Protocol) Bibliography", Personal site (work in progress), July 2013, <http://www.chiappa.net/~jnc/tech/lisp/LISPbiblio.html>.

Appendix A. Glossary/Definition of Terms

- 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. [Chiappa] 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 "IPvN" "address". The source and destination addresses of the _innermost_ header in a LISP packet are usually EIDs.
- 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 "IPvN" address, and of an "ETR". The source and adestination addresses of the _outermost_ header in a LISP packet are usually RLOCs.
- 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.
- 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.
- 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.
- 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.

- 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.
- 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.
- MR, Map Resolver: A LISP "node" to which "ITRs' send requests for "mappings". See Section 8.2.2, "Interface to the Mapping System", for more.
- 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. See Section 8.2.2, "Interface to the Mapping System" for more.
- 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. See Section 8.2.1, "Mapping System Organization" for more.
- Mapping Database: The term 'mapping database' refers to the entire collection of {EID->RLOC} "mappings" spread throughout the entire LISP system. It is a subset of the "mapping system". See Section 8.2, "Control Plane - Mapping System Overview", for more.
- Mapping Cache: A collection of copies of {EID->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.
- Indexing Sub-system: the entire ensemble of data and mechanisms which allows "MRs" to find out which "ETR(s)" hold the mappping 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. See Section 8.2.1, "Mapping System Organization" for more.
- DDT Vertex; Vertex: a node (in the graph theory sense of the term) in the (abstract) LISP namespace "delegation hierarchy".
- 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".
- 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. See Section 15.2.1.1, "PITRs", for more.
- 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. See Section 15.2.1.2, "PETRs" for more.

- RTR: Re-encapsulating Tunnel Router; a data plane 'anchor point' used by a LISP-speaking node to perform functions that can only be 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 [Improvements], Section "Improved NAT Support" for more.
- 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 ([RFC791]), 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

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. [LISP0]

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 undoubtly will for a long time to come).

B.3. Old LISP 'Models'

LISP, as initilly 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.

- 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.
- LISP 1.5: Similar to LISP 1, but the routability of EIDs happens on a separate network.
- LISP 2: EIDs are not routable; EID->RLOC mappings are available from the DNS.
- LISP 3: EIDs are not routable; and have to be looked up in 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. [ALT] 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 [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 sort 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}}.

Author's Address

J. Noel Chiappa
Yorktown Museum of Asian Art
Yorktown, Virginia
USA

EMail: jnc@mit.edu