LISP Working Group
Internet-Draft
Intended status: Informational
Expires: January 16, 2014

J. N. Chiappa
Yorktown Museum of Asian Art

July 15, 2013

An Architecural Introduction to the LISP Location-Identity Separation System draft-ietf-lisp-introduction-01

Abstract

LISP is an upgrade to the architecture of the IPvN internetworking system, one which separates location and identity (currently intermingled in IPvN addresses). This is a change which has been identified by the IRTF as a critically necessary evolutionary architectural step for the Internet. In LISP, nodes have both a 'locator' (a name which says _where_ in the network's connectivity structure the node is) and an 'identifier' (a name which serves only to provide a persistent handle for the node). A node may have more than one locator, or its locator may change over time (e.g. if the node is mobile), but it keeps the same identifier.

One of the chief novelties of LISP, compared to other proposals for the separation of location and identity, is its approach to deploying this upgrade. (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.) LISP aims to achieve the near-ubiquitous deployment necessary for maximum exploitation of an architectural upgrade by i) minimizing the amount of change needed (existing hosts and routers can operate unmodified); and ii) by providing significant benefits to early adopters.

This document is an introduction to the entire LISP system, for those who are unfamiliar with it.

material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 16, 2014.

Table of Contents

1. Prefaratory 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.

This 'Architectural Introduction' document is primarily intended for those who don't know anything about LISP, and want to start learning about it. It 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. Reading only the first part(s) 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.

This goal explains why the document has a somewhat unusual structure. It is not a reference document, where all the content on a particular topic is grouped in one place. (That role is filled by the various protocol specifications.) It starts with a very high-level view of the entire system, to provide readers with a mental framework to help understand the more detailed material which follows. A second pass over the whole system then goes into more detail; finally, individual sub-systems are covered in still deeper detail.

The intent is two-fold: first, the multiple passes over the entire system, each one going into more detail, are intended to ease understanding; second, people can simply stop reading when they have a detailed-enough understanding for their purposes. People who just want to get an idea of how LISP works might only read the first part(s), whereas people who are going to go on and read all the protocol specifications (perhaps to implement LISP) would need/want to 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. 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 a node (basically, 'who' a node is, and 'where' it is). At the moment, in both IPv4 and IPv6, addresses indicate both where the named device is, as well as identify it for purposes of end-end communication.

The distinction was more than a little hazy at first: the early Internet [RFC791], like the ARPANET before it [Heart] [NIC8246], co-mingled the two, 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. However, it has taken 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. [RFC6115]

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 in 2012 (although work to expand and improve it continues, and undoubtly will for a long time to come).

3. Deployment Philosophy

It may seem odd to cover 'deployment philosophy' at this point in such a document. However the deployment philosophy was a major driver for much of the design (to some degree the architecture, and to a very large measure, the engineering). So, as such an important motivator, it is very desirable for readers to have this material in hand as they examine the design, so that design choices that may seem questionable at first glance can be better understood.

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. 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 a viable deployment model is what has lead to some painful compromises in the design.

The extreme focus on a viable deployment scheme is one of the novelties of LISP.

3.1. Economics

The key factor in successful adoption, as shown by recent experience in the Internet - and little appreciated to begin with, some decades back - 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. One might have the world's best 'clean-slate' design, but if it does not have a deployment plan which is economically feasible, it's not good for much.

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. We believe LISP has met both of

these goals.

3.2. 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 is needed in only a few places (see Section 11.1 for more).

LISP also initially reuses, where-ever possible, existing protocols (IPv4 [RFC791] and IPv6 [RFC2460]). 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.

3.3. 'Self-Deployment'

LISP has deliberately employed a rather different deployment model, which we might call 'self-deployment' (for want of a better term); it does not require a huge push to get it deployed, rather, it is hoped that once people see it and realize they can easily make good use of it _on their own_ (i.e. without requiring adoption by others), it will 'deploy itself' (hence the name of the approach).

One can liken the problem of deploying new systems in this way to rolling a snowball down a hill: unless one starts with a big enough snowball, and finds a hill of the right steepness (i.e. the right path for it to travel), one's snowball is not going to go anywhere on its own. However, if one has picked one's spot correctly, once started, little additional work is needed.

4. LISP Overview

LISP is an incrementally deployable architectural upgrade to the existing Internet infrastructure, one which provides separation of location and identity. The separation is usually not perfect, for reasons which are driven by the deployment philosophy (above), and explored in a little more detail elsewhere (in [Perspective], Section "Namespaces-EIDs-Residual").

LISP separates the functions of location and identity of nodes (a

nebulous term, deliberately chosen for use in this document precisely because its definition is not fixed - you will not go far wrong if you think of a node as being something like a host), which are currently intermingled in IPvN addresses. (This document uses the meaning for 'address' proposed in [Atkinson], i.e. a name with mixed location and identity semantics.)

4.1. Basic Approach

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

A node may have more than one RLOC, or its RLOC may change over time (e.g. if the node is mobile), but it would normally always keep the same EID.

Technically, one should probably say that ideally, the EID names the node (or rather, its end-end communication stack, if one wants to be as forward-looking as possible), and the RLOC(s) name interface(s). (At the moment, in reality, the situation is somewhat more complex, as will be explained elsewhere (in [Perspective], Section "Namespaces-EIDs-Residual".)

This second distinction, of _what_ is named by the two classes of name, is necessary both to enable some of the capabilities that LISP provides (e.g the ability to seamlessly support multiple interfaces, to different networks), and is also a further enhancement to the architecture. Faailing to clearly recognize both interfaces and communication stacks as distinctly separate classes of things is another failing of the existing Internet architecture (again, one inherited from the previous generation of networking).

A novelty in LISP is that it uses existing IPvN addresses (initially, at least) for both of these kinds of names, thereby minimizing the deployment cost, as well as providing the ability to easily interact with unmodified hosts and routers.

4.2. Basic Functionality

The basic operation of LISP, as it currently stands, is that LISP augmented packet switches near the source and destination of packets intercept traffic, and 'enhance' the packets.

The LISP device near the source looks up additional information about the destination, and then wraps the packet in an outer header, one which contains some of that additional information. The LISP device near the destination removes that header, leaving the original, unmodified, packet to be processed by the destination node.

The LISP device near the original source (the Ingress Tunnel Router, or 'ITR') uses the information originally in the packet about the identity of its ultimate destination, i.e. the destination address, which in LISP 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 will be (initially at least) the address of the LISP device near the ultimate destination (the Egress Tunnel Router, or 'ETR').

{{Is it worth distinguishing between 'mapping' and 'binding'? Should the document pick one term, and stick with it?}}

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

When the packet gets to 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 device; these are normally denominated as 'xTRs'.

4.3. 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.

Mappings are requested on need, not (generally) pre-loaded; in other words, mapping are distributed via a 'pull' mechanism. Once obtained by an ITR, they are cached by the ITR, to limit the amount of control traffic to a practicable level. (The mapping system will be discussed in more detail below, in Section 6.2 and Section 10)

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 6.1.1, below.)

4.4. Interworking With Non-LISP-Capable Endpoints

The capability for 'easy' interoperation between nodes using LISP,

and existing non-LISP-using hosts (often called 'legacy' hosts) or sites (where 'site' is usually taken to mean a collection of hosts, routers and networks under a single administrative control), is clearly crucial.

To allow such interoperation, a number of mechanisms have been designed. This multiplicity is in part because different mechanisms have different advantages and disadvantages (so that no single mechanism is optimal for all cases), but also because with limited field experience, it is not clear which (if any) approach will be preferable.

One approach uses proxy LISP devices, called PITRs (proxy ITRs) and PETRs (proxy ETRs), to provide LISP functionality during interaction with legacy hosts. Another approach uses a device with combined LISP and NAT ([RFC1631]) functionality, named a LISP-NAT.

4.5. Security in LISP

LISP has a subtle security philosophy; see [Perspective], Section "Security", where it is laid out in some detail.

To provide a brief overview, 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. That means two things: First, with the limited manpower currently available, we cannot expect to create the complete security apparatus that we might see in the long term (which requires not just design, but also implementation, etc). 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 thorougly analyze what LISP needs for security, and then design, in detail, a scheme for providing that security. 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. Later on, the design can be fully specified, implemented, and deployed.

5. Initial Applications

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 unmodified hosts, use of LISP can still provide benefits for communications with 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 deploying LISP, thereby increasing the cost/benefit ratio needed to drive deployment, and increasing the 'self-deployment' aspect of LISP.

Note also that this section only lists likely _early_ applications and benefits - if and once deployment becomes more widespread, other aspects will come into play (as described in [Perspective], in the "Goals of LISP" section).

5.1. Provider Independence

Provider independence (i.e. the ability to easily change one's Internet Service Provider) was probably the first place where the Internet engineering community finally really felt 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), the so-called "provider aggregated" 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 renumber, a painful undertaking. [RFC5887]

In theory, it ought to be possible to update the DNS entries, and have everyone switch to the new addresses, but in practise, addresses are embedded in many places, such as firewall configurations at other sites.

Having separate namespaces for location and identity greatly reduces the problems involved with renumbering; 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.

5.2. 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 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, 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.

Multi-homing was once somewhat esoteric, but a number of trends are

driving an increased desirability, e.g. the wish to have multiple ISP links to a site for robustness; the desire to have mobile handsets connect up to multiple wireless systems; etc.

Again, separation of location and identity, and the existince of a binding layer which can be updated fairly quickly, as provided by LISP, is a very useful tool for all variants of this issue.

5.3. Traffic Engineering

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 computationa), 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 binding 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 binding 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 6.2 below.)

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.

5.4. 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 so-called 'Default-Free-Zone' (DFZ) - the core of the Internet, the part where routes to _all_ ultimate destinations must be available. LISP is expected to help with this; for more detail, see Section 11.6, below.

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

5.5. 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.

5.6. IP Version Reciprocal Traversal

Note that LISP 'automagically' allows 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 would allow 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 heavyweight 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'.

5.7. Local Uses

LISP has a number of use cases which are within purely 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, IP version traversal, and 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).

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.

6. Major Functional Subsystems

LISP has only two major functional sub-systems - the collection of LISP packet switches (the xTRs), and the mapping system, 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 9 and Section 10, respectively).

6.1. xTRs

xTRs are fairly normal packet switches, enhanced with a little extra functionality in both the data and control planes, to perform LISP data and control functionality.

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 'vanilla'); i.e. looking up the mapping; encapsulating the packet; and sending it to the ETR. This encapsulation is done using UDP [RFC768] (for reasons to be explained below, in Section 9.2), along with an additional 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 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 Map-Request control messages; handling the returning Map-Replies which contain the requested information; managing the local 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 neighbour reachability and liveness function (see Section 12.4); interacting with the mapping sub-system (next section); and answering requests for mappings (ditto).

6.1.1. 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.

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.

The first, [Iannone], was performed in the very early stages of the LISP effort, to verify that that approach was feasible. First, packet traces of all traffic over the external connection of a large university (around 10,000 users) 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. The simulations set no limit on the total cache size, but used a range of cache retention

times (i.e. an entry that remained unused longer than a fixed retention time was discarded), from 3 minutes, up to 300 minutes.

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 (at night, with the shortest retention time) up to about 100,000. Using some estimations as to i) how many RLOCs the average mapping would have (since this will affect its size), and ii) how much memory it would take to store a mapping, 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 (again, depending on the time of day, cache settings, etc) from 30 per minute, up to 3,000 per minute (i.e. 150 per second; with the shortest timeout, and thus the smallest cache). 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 second, [Kim], was in general terms similar, except that it used data from a large ISP (taken over two days, at different times of the year), 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: 60 seconds, 180 seconds, and 1,800 seconds (30 minutes), since the previous study had indicated that extremely long times (hours) had little additional benefit.

The results were similar: cache sizes ranges from 20,000 entries with the shortest timeout, to roughly 60,000 with the longest; the miss rate ranged from very roughly 400 per minute (with the longest timeout) to very roughly 7,000 per minute (with the shortest), 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 (some from an earlier study, [Jakab]). 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.

6.2. Mapping System

The mapping database is a distributed, and potentially replicated, database which holds mappings between EIDs (identity) and RLOCs (location). 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 often is) as small as a single EID. 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 routing table came to be dominated by such small entries.

A particular node may have more than one RLOC, or may change its RLOC(s), while keeping its singlar identity.

The mapping contains not just the RLOC(s), but also (for each RLOC for any given EID) priority and weight (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.

6.2.1. 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 covers the actual mappings themselves; 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 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 user interface sub-system, and indexing sub-system (which holds and provides the actual information).

6.2.2. 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 provied by devices called Map Resolvers (MRs).

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

Similarly, the client interface to the indexing system from an ETR's point of view is through devices called Map Servers (MSs - admittedly a poorly chosen term, since their primary function is not to respond to queries, but it's too late to change it now).

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 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 system are thus likewise hidden from the 'ordinary' ETRs.

6.2.3. Indexing Sub-system

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

DDT replaced an earlier indexing sub-system, ALT ([Perspective], section "Appendices-ALT"); this swap validated the concept of having a separate client-interface sub-system, which would allow the actual indexing sub-system to be replaced without needing to modify the clients.

6.2.3.1. 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 tree of DDT 'nodes', starting with the 'root' DDT node. Each node is responsible (authoritative?) for one or more blocks of the EID namespace.

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

Just as with DNS, for reasons of performance, reliability and robustness, any particular node in the DDT delegation tree may be

instantiated in more than one redundant physical server machines. Obviously, all the servers which instantiate a particular node in the tree have to have identical data about that node.

Also, although the delegation hierarchy is a strict tree {{check - do all servers for the delegation of block X have to return the same list of servers for that block?}}, a single DDT server could be responsible (authoritative?) for more than one block of the EID namespace.

Eventually, leaf nodes in the DDT tree assign ({{delegate? - it's all static configured, nothing is dynamic}}) EID namespace blocks to MS's, which are DDT terminal nodes; i.e. a leaf of the tree is reached when the delegation points to an MS instead of to another DDT node.

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 that block of EID namespace.

6.2.3.2. Use of DDT by MRs

An MR which wants to find a mapping for a particular EID first interacts with the nodes of the DDT tree, discovering (by querying DDT nodes) the chain of delegations which cover that EID. Eventually it is directed to an MS, and then to an ETR which is responsible {{authoritative?}} for that EID.

Also, again like DNS, MRs cache information about the delegations in the DDT 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 do a complete tree-walk of the DDT tree 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.

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

7. 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, 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.

7.1. 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) 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 is also a border router for the site), the border router looks up the desination address (an EID) in its local mapping cache. 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). The destination address in the outer header is set by the ITR to the RLOC of the destination ETR.

The packet is then sent off through the Internet, using normal Internet routing tables, etc.

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.

7.2. A Mapping Cache Miss

If a host sends a packet, and it gets to the ITR, and the ITR both i) determines that it needs to perform LISP processing on the user data packet, but ii) does not yet have a mapping cache entry which covers that destination EID, then more complex processing ensues.

It 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 see if it has the RLOC for the ETR for that destination EID. If not, it will query the DDT system to find the RLOC of the ETR. When it has the RLOC, it will send the Map-Request on to the 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. (Typically, like many ARP

implementations, the original user packet will have been discarded, not cached waiting for the mapping to be found. When the host retransmits the packet, the mapping will be there, and the packet will be forwarded.)

8. 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.

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.

9. xTRs

As mentioned above (in Section 6.1), xTRs are the basic data-handling devices in LISP. 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.

9.1. When to Encapsulate

An ITR knows that an ultimate destination is 'running' LISP (remember that the 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 normally.

Note that the ITR cannot simply depend on the appearance, or non-appearance, of the destination in the routing tables in the DFZ, as a way to tell if an ultimate destination is a LISP node or not, because mechanisms to allow interoperation of LISP sites and 'legacy' sites necessarily involve advertising LISP sites' EIDs into the DFZ.

9.2. UDP Encapsulation Details

The UDP encapsulation used by LISP for carrying traffic from ITR to ETR, and many of the details of how it works, were all chosen for very practical reasons.

Use of UDP (instead of, say, a LISP-specific protocol number) was driven by the fact that many devices filter out 'unknown' protocols, so adopting a non-UDP encapsulation would have made the initial deployment of LISP harder - and our goal (see Section 3.1) was to make the deployment as easy as possible.

The UDP source port in the encapsulated packet is a hash of the original source and ultimate destination; 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 an intolerable load, for no benefit.

9.3. 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 8-bit 'flags' field, and it also includes two data fields (one 24 bits, one 32), 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.

9.3.1. 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. [Versioning] 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 multiplex 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. Some felt that this was non-optimal, and that a better approach would have been to have 'fingerprints' which were computed from the current mapping data (i.e. a hash). It is not clear that the ordering buys much (if anything), and the potential for mishaps with manually configured version numbers is self-evident.

9.3.2. 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 correspondent 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 lack of a response is not necessarily _proof_ that something has gone wrong - but it stronly 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 12.5 for more about Echo Nonces.)

9.3.3. 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).

9.4. Probing

RLOC-Probing (see [LISP], 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.

9.5. 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.)

9.6. 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 what what have been actually delegated to it).

Very briefly, the ITR provided a One-Time Key with its query; this key is used by both the MS (to verify the EID block that it has delegated to the ETR), and indirectly by the ETR (to verify 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).

9.7. 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. {{Is this still there? Check the latest version of the spec.}} 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.

9.8. Fragmentation

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.

One, called the 'stateful' approach, 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.

It is not clear at this point which approach is preferable.

10. The Mapping System

RFC 1034 ("DNS Concepts and Facilities") has this to say about the DNS name to IP address 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 system.

To recap, the mapping system is split into an indexing sub-system, which keeps track of where all the mappings are kept, and the mappings themselves, the authoritative copies of which are always held by ETRs.

10.1. The Mapping System Interface

As mentioned in Section 6.2.2, 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 the details, see [MapInterface].

10.1.1. 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. [AFI] The inclusion of the AFI allows the mapping system interface (as embodied in these control packets) a great deal of flexibility. (See [Perspective], Section "Namespaces" for more on this.)

Other important fields are the source EID (and its AFI), and one or more RLOCs for the source EID, along with their AFIs. 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 9.7).

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.

10.1.2. 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.)

For each RLOC in the entry, there is the RLOC, its AFI (of course), priority and weight fields (see Section 6.2), and multicast priority and weight fields.

10.1.2.1. Solicit-Map-Request Messages

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

SMR's, like most other control traffic, is rate-limited. {{Need to say more about rate limiting, probably in security section? Ref to that from here.}}

10.1.3. 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 9.3.1), 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. [Mobility]

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

10.2. The DDT Indexing Sub-system

As previously mentioned Section 6.2.3, the indexing sub-system in LISP is currently the DDT system.

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

The interaction between MRs and DDT servers is not complex; the MR sends the DDT server a Map-Request control message (which looks almost exactly like the Map-Request which an ITR sends to an MR). 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 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. The Map-Referral also gives the RLOCs of all the machines which are DDT servers for that block. and the MR can then send Map-Requests to any one (or all) of them.

Control flags in the Map-Referral indicate to the querying MR whether the referral is to another DDT node, an MS, or an ETR. If the

former, the MR then sends the Map-Request to the child DDT node, repeating the process.

If the latter, 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, so in such cases the Map-Reply will come from the MS, not the ETR. {{I think this case has been mentioned already; check.}})

Delegations are cached in the MRs, so that once an MR has received information about a delegation, it will not need to look that up again. Once it has been in operation for a short while, it will only need to ask for 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 10.6), significant amounts of modeling and performance measurement have been performed, to verify that DDT has (and will continue to have) acceptable performance.

10.2.1. Map-Referral Messages

Map-Referral messages look almost identical to Map-Reply messages (which is felt to be an advantage by some people, although having a more generic record-based format would probably be better in the long run, as ample experience with DNS has shown), except that the RLOCs potentially name either i) other DDT nodes (children in the delegation tree), or ii) terminal MSs.

10.3. 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.

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

There are generally no mechanisms specified yet to ensure coherence between multiple copies of any particular data item, etc - 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.

10.4. Security of the DDT Indexing Sub-system

LISP provides an advanced model for securing the mapping indexing system, in line with the overall LISP security philosophy.

Briefly, securing the mapping indexing system is broken 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 nodes/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. This requires very little configuration in the clients, and is fairly secure.

The interface between the DDT nodes/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 DNS-SEC ([RFC4033]), where configuration complexity in the servers has been a major stumbling block to deployment.

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

10.5. Extended Tools

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 'source-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.

10.6. 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. [Jakab] The basic structure and concepts of DDT are identical to those of TREE, so the performance simulation work done for that design applies aequally 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 lower nodes 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.

11. Deployment 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.

11.1. LISP Deployment Needs

As mentioned earlier (Section 3.2), LISP requires no change to almost all existing hosts and routers. Obviously, however, one must deploy _something_ to run LISP! Exactly what that has to be will depend greatly on the details of the site's existing networking gear.

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 small amount of LISP-specific support infrastructure, such as MRs, MSs, the DDT hierarchy, etc but much of this will either i) already be deployed, and if the new site can make arrangements to use it, it need do nothing else, or ii) those functions it must provide may be co-located in other LISP devices (again, either new devices, or new software on existing ones).

11.2. Internetworking Mechanism

One aspect which has received a lot of attention are the mechanisms previously referred to (in Section 4.4) to allow interoperation of LISP sites with so-called 'legacy' sites which are not running LISP (yet).

To briefly refresh what was said there, there are two main approaches to such interworking: proxy nodes (PITRs and PETRs), and an alternative mechanism using device with combined NAT and LISP functionality; these are described in more detail here.

11.3. Proxy Devices

PITRs (proxy ITRs) serve as ITRs for traffic _from_ legacy hosts to nodes using LISP. PETRs (proxy ETRs) serve as ETRs for LISP traffic _to_ legacy hosts (for cases where a LISP device 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.

11.3.1. PITRs

PITRs (proxy ITRs) serve as ITRs for traffic _from_ legacy hosts to nodes using LISP. To do that, 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.

As mentioned previously (Section 9.1), an ITR at another LISP site can avoid using a PITR (i.e. it can detect that a given ultimate destination is not a legacy host, if a PITR is advertising it into the DFZ) by checking to see if a LISP mapping exists for that ultimate destination.

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

A PITR may cover a group of EID blocks with a single EID advertisement, 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.)

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.

11.3.2. PETRs

PETRs (proxy ETRs) serve as ETRs for LISP traffic _to_ legacy hosts, for cases where a LISP device 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 paired with 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.

11.4. LISP-NAT

A LISP-NAT device, 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). [Interworking]

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

11.5. Use Through NAT Devices

Like them or not (and NAT devices have many egregious issues - some inherent in the nature of the process of mapping addresses; others, such as the brittleness due to non-replicated critical state, caused by the way NATs were introduced, as stand-alone 'invisible' boxes), NATs are both ubiquitous, and here to stay for a long time to come.

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.

11.5.1. First-Phase NAT Support

The first mechanism used by LISP to operate through a NAT device only

worked with some NATs, those which were configurable to allow inbound packet traffic to reach a configured host.

A pair of new LISP control messages, LISP Echo-Request and Echo-Reply, 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.

The fact that this mechanism did not support all NATs, and also required manual configuration of the NAT, meant that this was not a good solution; in addition, since LISP expects all incoming data traffic to be on a specific port, it was not possible to have multiple ETRs behind a single NAT (which normally would have only one global address to share, meaning port mapping would have to be used, except that...)

11.5.2. Second-Phase NAT Support

For a more comprehensive approach to support of LISP xTR deployment behind NAT devices, a fairly extensive supplement to LISP, LISP NAT Traversal, has been designed. [LISP-NAT]

A new class of LISP device, the LISP Re-encapsulating Tunnel Router (RTR), passes traffic through the NAT, both to and from the xTR. (Inbound traffic has to go through the RTR as well, since otherwise multiple xTRs could not operate behind a single NAT, for the 'specified port' reason in the section above.)

(Had the Map-Reply included a port number, this could have been avoided - although of course it would be possible to define a new RLOC type which included protocol and port, to allow other encapsulation techniques.)

Two new LISP control messages (Info-Request and Info-Reply) allow an xTR to detect if it is behind a NAT device, and also discover the global IP address and UDP port assigned by the NAT to the xTR. A modification to LISP Map-Register control messages allows the xTR to initialize mapping state in the NAT, in order to use the RTR.

This mechanism addresses cases where the xTR is behind a NAT, but the xTR's associated MS is on the public side of the NAT; this limitation, that MS's must be in the 'public' part of the Internet, seems reasonable.

11.6. LISP and DFZ Routing

One of LISP's original motivations was to try and control the growth of the size of the so-called 'Default-Free-Zone' (DFZ), the core of the Internet, the part where routes to _all_ destinations must be

available. As LISP becomes more widely deployed, it can help with this issue, in a variety of ways.

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

11.6.1. Long-term Possibilities

At a later stage of the deployment, a more aggressive approach becomes available: taking part of the DFZ, one for which all 'stub' sites connected to it have deployed LISP, and removing all 'EID routes' (used for backwards compatability with 'legacy' sites); only RLOC routes would remain in the routing table in that part of the Internet backbone.

Obviously there would be a boundary between the two parts of the DFZ, and the routers on the border would have to (effectively) become PITRs, and inject routes to all of the LISP sites 'behind' them into the 'legacy' DFZ (to coin a name for the part of the DFZ which, for reasons of interoperability with legacy sites, still carries EID routes).

Note that it is likely not feasible to have the 'RLOC only' part of the DFZ in the 'middle' of the DFZ; that would require (effectively) EID routes to be removed from BGP on crossing the boundary _into_ the RLOC DFZ, but re-created on crossing the boundary _out_ of the RLOC DFZ. This is likely to be impractical, leading to the suggestion of a simpler boundary between the RLOC-only part of the DFZ, and the 'legacy' DFZ.

The mechanism for detecting which routes are 'EID routes' and which are 'RLOC routes' (required for the boundary routers to be able to filter out the 'EID routes') would also need to be worked out; the

most likely appears to be something involving BGP attributes.

12. Fault Discovery/Handling

LISP is, in terms of its functionality, a fairly simple system: the list of failure modes is thus not extensive.

12.1. Handling Missing Mappings

Handling of missing mappings is fairly simple: 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 'default PITR'.

A number of PITRs advertise all EID blocks into the backbone routing, so that any ITRs which are temporarily missing a mapping can forward the traffic to these default PITRs via normal transmission methods, where they are encapsulated and passed on.

12.2. 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 traffic is being sent to 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

12.2.1. 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. [Versioning]

12.2.2. Outdated Mappings - Wrong ETR

If an ITR is holding a seriously 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).

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.

12.2.3. 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 device at all. In such cases, one might get an ICMP Destination Unreachable 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 [LISP], 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 9.3.2) 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.

12.3. 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.

12.4. Neighbour Liveness

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

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

Note that the two terms (liveness and reachability) are _not_ synonmous (although a lot of LISP documentation confuses them). Liveness is a property of a node - it is either up and functioning, or it is not. Reachability is only a property of a particular _pair_ of nodes.

If packets sent from a first node to a second are successfully received at the second, it is 'reachable' from the first. However, the second node may at the very same time _not_ be reachable from some other node. Reachability is _always_ a ordered pairwise property, and of a specified ordered pair.

12.5. Neighbour Reachability

A more significant issue than whether a particular ETR E is up or not is, as mentioned above, that although ETR E may be up, attached to the network, etc, an issue in the network between a source ITR I and E may prevent traffic from I from getting to E. (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 devices, reliance on it as a sole mechanism would produce a fair amount of overhead.)

13. Current Improvements

In line with the philosophies laid out in Section 8, LISP is something of a moving target. This section discusses some of the contemporaneous improvements being made to LISP.

13.1. Improved NAT Support

13.2. Mobile Device Support

Mobility is an obvious capability to provide with LISP. Doing so is relatively simple, if the mobile host is prepared to act as its own ETR. It obtains a local 'temporary use' address, and registers that address as its RLOC. Packets to the mobile host are sent to its temporary address, wherever that may be, and the mobile host first unwraps them (acting as an ETR), and the processes them normally (acting as a host).

(Doing mobility without having the mobile host act as its ETR is difficult, even if ETRs are quite common. The reason is that if the ETR and mobile host are not integrated, during the step from the ETR to the mobile host, the packets must contain the mobile host's EID,

and this may not be workable. If there is a local router between the ETR and mobile host, for instance, it is unlikely to know how to get the packets to the mobile host.)

If the mobile host migrates to a site which is itself a LISP site, things get a little more complicated. The 'temporary address' it gets is itself an EID, requiring mapping, and wrapping for transit across the rest of the Internet. A 'double encapsulation' is thus required at the other end; the packets are first encapsulated with the mobile node's temporary address as their RLOC, and then this has to be looked up in a second lookup cycle (see Section 9.1), and then wrapped again, with the site's RLOC as their destination.

This results in slight loss in maximum packet size, due to the duplicated headers, but on the whole it is considerably simpler than the alternative, which would be to re-wrap the packet at the site's ETR, when it is discovered that the ultimate destination's EID was not 'native' to the site. This would require that the mobile node's EID effectively have two different mappings, depending on whether the lookup was being performed outside the LISP site, or inside.

{{Also probably need to mention briefly how the other end is notified when mappings are updated, and about proxy-Map-Replies.}} [Mobility]

13.3. Multicast Support

Multicast may seem an odd thing to support with LISP, since LISP is all about separating identity from location, but although a multicast group in some sense has an identity, it certainly does not have _a_ location.

However, multicast is important to some users of the network, for a number of reasons: doing multiple unicast streams is inefficient; it is easy to use up all the upstream bandwidth, and without multicast a server can also be saturated fairly easily in doing the unicast replication. So it is important for LISP to 'play nicely' with multicast; work on multicast support in LISP is fairly advanced, although not far-ranging.

Briefly, 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, LISP's mapping capability is used: it is just applied to the source, not the destination (as with most LISP activity); the inner source is the EID, and the outer source is the EID's RLOC.

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 approach currently used by LISP requires no packet format changes to existing multicast protocols. See [Multicast] for more; additional LISP multicast issues are discussed in [LISP], Section 12.

13.4. {{Any others?}}

14. 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) Vina Ermagan, Vince Fuller and Vasileios Lakafosis for their careful review of, and helpful suggestions for, this document. (If I have missed anyone in this list, I apologize most profusely.) A very special thank you goes to Joel Halpern, who, 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 Vince Fuller and Dino Farinacci 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.

15. IANA Considerations

This document makes no request of the IANA.

16. Security Considerations

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

17. References

17.1. Normative References

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

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

[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.

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

[AFI] IANA, "Address Family Indicators (AFIs)", Address Family Numbers, January 2011, <http://www.iana.org/assignments/address-family-numbers>.

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

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

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

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

[DDT] V. Fuller, D. Lewis, and D. Farinacci, "LISP Delegated Database Tree", draft-ietf-lisp-ddt-00 (work in progress), October 2012.

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

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

[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.

[LISP-NAT] 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-07 (work in progress), April 2012.

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

[Deployment] L. Jakab, A. Cabellos-Aparicio, F. Coras, J. Domingo-Pascual, and D. Lewis, "LISP Network Element Deployment Considerations", draft-ietf-lisp-deployment-08 (work in progress), June 2013.

17.2. 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.

[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.

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

[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.

[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.

[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.

[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'.

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

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

[NSAP] International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", ISO Standard 7489.1984, 1984.

[Atkinson] R. Atkinson, "Revised draft proposed definitions", RRG list message, Message-Id: 808E6500-97B4-4107-8A2F-36BC913BE196@extremenetworks.com, 11 June 2007, <http://www.ietf.org/mail-archive/web/ram/current/msg01470.html>.

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

[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.

[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.

[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>.

[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.

[Jakab] 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.

[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.

Appendix A. Glossary/Definition of Terms

- Address
- Locator
- EID
- RLOC
- ITR
- ETR
- xTR
- PITR
- PETR
- MR
- MS
- DFZ

Appendix B. Other Appendices

B.1. 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.2. Possible Other Appendices

-- Location/Identity Separation Brief History
-- LISP History

Author's Address

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

EMail: jnc@mit.edu