LISP Working Group
Internet-Draft
Intended status: Informational
Expires: January 17, 2013

J. N. Chiappa
Yorktown Museum of Asian Art
July 16, 2012

An Introduction to the LISP Location-Identity Separation System
draft-chiappa-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. It is intended to be both easy to follow, and also give a fairly detailed understanding of the entire system.

Status of This Memo

Copyright Notice

Table of Contents

1. 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.

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. [LISP]

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

LISP transitioned from an IRTF activity to an IETF WG in March 2009, and after numerous revisions, the basic specifications moved to becoming RFCs in 2012 (although work to expand and improve it continues, and undoubtly will for a long time to come).

2. 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.

2.1. 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 just a mildly interesting piece of paper.

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.

2.2. 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 devices in the Internet (both hosts and routers), and also initially reuses, whereever 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 succeeds.

2.3. Self-Deployment

LISP has deliberately employed a rather different deployment model, which we might call 'self-deployment'; 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 initial snowball, and finds a hill of the right steepness (i.e. the right path for it to travel, once it starts moving), one's snowball is not going to go anywhere on its own. However, if one has picked one's spot correctly, little additional work is needed - just stand back and watch it go.

3. 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 [Architecture], Section "Namespaces-EIDs-Residual").

LISP separates the functions of location and identity, current intermingled in IPvN addresses. (This document uses the meaning for 'address' proposed in [Atkinson], i.e. a name with mixed location and identity semantics.)

3.1. 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', and an 'identifier' (a name which serves only to provide a persistent handle for the node), called an 'EID'. A node may have more than one RLOC, or its RLOC may change over time (e.g. if the node is mobile), but it keeps 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 [Architecture], 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.

3.2. 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 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 one can view as 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 bindings from EIDs to RLOCS. The destination RLOC will be (initially at least) the address of the LISP device near the destination (the Egress Tunnel Router, or 'ETR').

The ITR then generates a new outer header for the original packet, with that header containing the 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'.

3.3. 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, to limit the amount of control traffic to a practicable level. (The mapping system will be discussed in more detail below, in Section 5.2 and Section 9)

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. [Iannone] (This subject will be discussed in more detail in Section 9.5, below.)

3.4. Interworking With Non-LISP-Capable Endpoints

The capability for 'easy' interoperation between nodes using LISP, and existing non-LISP-using hosts or sites (often called 'legacy' hosts), 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 sites. Another approach uses a device with combined LISP and NAT ([RFC1631]) functionality, named a LISP-NAT.

4. 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 importan 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 [Architecture], in the "Goals of LISP" section).

4.1. 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 binding from the EIDs to the new RLOCs.

4.2. 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.

4.3. 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 visualized when the Internet was designed, so support for it is somewhat in the 'when the only tool you have is a hammer, everything looks like nail' category.

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 5.2 below.)

4.4. 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.

4.5. 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'.

4.6. 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.

5. Major Functional Subsystems

LISP has only two major functional subsystems - 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 8 and Section 9, respectively).

5.1. 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 sites may be sent 'vanilla'); 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 8.2), along with an additional IPvN header (to hold the asource 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 'vanilla' 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 indexing system (next section); and answering requests for mappings (ditto).

5.2. Mapping System

The mapping database is a distributed, and potentially replicated, database which holds bindings between EIDs (identity) and RLOCs (location). To be exact, it contains bindings 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 binding 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.

5.2.1. Mapping System Organization

The mapping system is actually split into two major functional sub-systems. The actual bindings themselves are held by the ETRs, and an ITR which needs a binding effectively gets it 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 subsystem, an 'indexing system', itself also a distributed, potentially replicated database, provides information on which ETR(s) are authoritative sources of information about the bindings which are available.

5.2.2. Interface to the Mapping System

The client interface to the mapping system from an ITR's point of view is not with the indexing system directly; rather, it is through 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 uses the indexing system to eventually forward the Map-Request to the appropriate ETR. The ETR formulates reply control messages (Map-Reply packets), which is conveyed to the ITR. The details of the indexing system, etc, are thus hidden from the 'ordinary' 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, 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.

5.2.3. Indexing Subsystem

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

Again, extensive large-scale simulations driven by lengthy recordings of actual traffic at several major sites, have been performed to verify the effectiveness of this particular indexing system. [Jakab]

6. 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 destination (at a particular ITR) is to be processed by LISP.

6.1. 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 open HTTP connection) as it makes its way from the source host to the destination.

{{Rest to be written.}}

6.2. 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.

{{Rest to be written.}}

7. Design Approach

Before describing LISP's components in more detail below, it may be worth saying a few words about the design philosophy used in creating them - this may make clearer the reasons for some engineering choices in the mechanisms given there.

7.1. Quick Implement-Test Loop

LISP uses a philosophy similar to that used in the early days of the Internet, which is to just build it, then try it and see what happens, and move forward from there based on what actually happens. The concept has been to get something up and running, and then modify it based on testing and experience.

7.1.1. No Desk Fixes

Don't try and forsee all issues from desk analysis. (Which is not to say that one should not spend _some_ time on trying to forsee problems, but be aware that it is a 'diminishing returns' process.)

The performance of very large, complex, physically distributed systems is hard to predict, so rather than try (which would necessarily be an incomplete exercise anyway, testing would inevitably be required eventually), at a certain point it's better just to get on with it - and you will learn a host of other lessons in the process, too.

7.1.2. Code Before Documentation

This is often a corollary to the kind of style described above. While it probably would not have been possible in a large, inhomogenous group, the small, close nature of the LISP implementation group did allow this approach.

7.2. Only Fix Real Problems

Don't worry about anything unless experience show it's a real problem. For instance, in the early stages, much was made out of the problem of 'what does an ITR do if it gets a packet, but does not (yet) have a mapping for the destination?'

In practise, simply dropping such packets has just not proved to be a problem; the higher level protocol will retransmit them after a timeout, and the mapping is usually in place by then. So spending a lot of time (and its companion, energy) and mechanism (and _its_ extremely undesirable companion, complexity) on solving this 'problem' would not have been the most efficient approach, overall.

7.3. No Theoretical Perfection

Attack hard problems with a number of cheap and simple mechanisms that co-operate and overlap. Trying to find a single mechanism that is all of:

- Robust
- Efficient
- Fast

is often (usually?) a fool's errand. (The analogy to the aphorism 'Fast, Cheap, Good - Pick Any Two' should be obvious.) However, a collection of simple and cheap mechanisms may effectively be able to meet all of these goals (see, for example, ETR Liveness/Reachability, Section 12.4).

Yes, this results in a system which is not provably correct in all circumstances. The world, however, is full of such systems - and in the real world, effective robustness is more likely to result from having multiple, overlapping mechanisms than one single high-powered (and inevitably complex) one. In the world of civil engineering, redundancy is now accepted as a key design principle; the same should be true of information systems. [Salvadori]

7.3.1. No Ocean Boiling

Don't boil the ocean to kill a single fish. This is a combination of 7.2 (Only Fix Real Problems) and 7.3 (No Theoretical Perfection); it just means that spending a lot of complexity and/or overhead to deal with a problem that's not really a problem is not good engineering.

7.4. Just Enough Security

How much security to have is a complex issue. It's relatively easy for designers to add good security, but much harder to get the users to jump over all the hoops necessary to use it. LISP has therefore adopted a position where we add 'just enough' security.

The overall approach to security in LISP is fairly subtle, though, and is covered in more detail elsewhere (in [Architecture], Section "Security").

8. xTRs

As mentioned above (in Section 5.1), 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.

8.1. When to Encapsulate

An ITR knows that a destination is running 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 destination is not running LISP, and the packet can be forwarded normally.

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

8.2. UDP Encapsulation Details

The UDP encapsulation used by LISP for carrying traffic from ITR to ETR, and many of the details of how the 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 2.1) 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 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.

8.3. Header Control Channel

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

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.

8.3.1. Echo Nonces

One important use 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.)

8.3.2. 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).

8.4. 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.

8.5. Mapping Gleaning in ETRs

As an optimization to the mapping acquisition process, ETRs are allowed to 'glean' mappings from incoming user data packets, and also from incoming Map-Request control messages. This is not secure, and so any such mapping must be 'verified' by sending a Map-Request to get an authoritative mapping. (See further discussion of the security implications of this in [Architecture], 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. 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.

As previously mentioned, the mapping system is split into an indexing subsystem, which keeps track of where all the mappings are kept, and the mappings themsleves, the authoritative copies of which are always held by ETRs.

9.1. The Indexing Subsystem

The indexing system in LISP is currently implemented by the DDT system. LISP initially used (for ease of getting something operational without having to write a lot of code) an indexing system called ALT, which used BGP running over virtual tunnels. [ALT] This proved to have a number of issues, and has now been superseded by DDT.

In DDT, the EID namespace(s) are instantiated as a tree of DDT nodes. Starting with the root node(s), which have 'reponsibility' for the entire namespace, portions of the namespace are delegated to child nodes, in a recursive process extending through as many levels as are needed. Eventually, leaf nodes in the DDT tree delegate namespace blocks to ETRs.

MRs obtain information about delegations by interrogating DDT nodes, and caching the results. aThis allows them, when passed a request for a mapping by an ITR, to forward the mapping request to the appropriate ETR (perhaps after loading some missing delegation entries into their delegation cache).

9.2. The Mapping System Interface

As mentioned in Section 5.2.2, 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 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.

9.2.1. 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 [Architecture], 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 8.5).

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

9.2.2. Map-Reply Maessages

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 5.2), and multicast priority and weight fields.

9.2.3. 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 11.1), 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.

9.2.4. 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.

There are also optional authentication fields; see [Architecture], Section "Security-Mappings" for more.

9.3. 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.

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.

9.4. 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 are '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.

9.5. Expected Performance

{{To be written.}}

10. 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.

10.1. Internetworking Mechanism

One aspect which has received a lot of attention are the mechanisms previously referred to (in Section 3.4) 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.

10.2. 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 sites, without encapsulation).

Note that return traffic _to_ a legacy site 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 destination. However, for some kinds of LISP operation (e.g. mobile nodes), this is not possible; in these situations, the PETR is needed.

10.2.1. 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 8.1), an ITR at another LISP site can avoid using a PITR (i.e. it can detect that a given destination is not a legacy site, if a PITR is advertising it into the DFZ) by checking to see if a LISP mapping exists for that 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, it 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.

10.2.2. PETRs

PETRs (proxy ETRs) serve as ETRs for LISP traffic _to_ legacy hosts, for cases where a LISP device cannot send packets to sites 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 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 source along its entir 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 destination. However, this is not advised: as mentioned, it is easy to tell if something is a legacy destination.

10.3. 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]

10.4. LISP and DFZ Routing

{{To be written.}}

10.5. 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.

10.5.1. 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...)

10.5.2. 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. [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. Current Improvements

In line with the philosophies laid out in Section 7, LISP is something of a moving target. This section discusses some of the contemporaneous improvements being made to LISP.

11.1. Mapping Versioning

As mentioned, LISP has been under development for a considerable time. One early addition to LISP (it is already part of the base specification) is mapping versioning; i.e. the application of identifying sequence numbers 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 (see Section 8.3).

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.

11.2. Replacement of ALT with DDT

As mentioned in Section 9.2, an interface is provided to allow replacement of the indexing subsystem. LISP initially used an indexing system called ALT. [ALT] ALT was relatively easy to construct from existing tools (GRE, BGP, etc), but it had a number of issues that made it unsuitable for large-scale use. ALT is now being superseded by DDT.

As indicated previously (Section 9.5), the basic structure and operation of DDT is identical to that of TREE, so the extensive simulation work done for TREE applies equally to DDT, as do the conclusions drawn about TREE's superiority to ALT. [Jakab]

{{Briefly synopsize results}}

11.2.1. Why Not Use DNS

One obvious question is 'Since DDT is so similar to DNS, why not simply use DNS?' In particular, people are familiar with the DNS, how to configure it, etc - would it not thus be preferable to use it? To completely answer this would take more space that available here, but, briefly, there were two main reasons, and one lesser one.

First, the syntax of DNS names did not lend itself to looking up names in other syntaxes (e.g. bit fields). This is a problem which has been previously encountered, e.g. in reverse address lookups. [RFC5855]

Second, as an existing system, the interfaces between DNS (should it have been used as an indexing subsystem for LISP) would not be 'tuneable' to be optimal for LISP. For instance, if it were desired to have the leaf node in an indexing lookup directly contact the ETR on behalf of the node doing the lookup (thereby avoiding a round-trip delay), that would not be easy without modifications to the DNS code. Obviously, with a 'custom' system, this issue does not arise.

Finally, DNS security, while robust, is fairly complex. Doing DDT offered an opportunity to provide a more nuanced security model. (See [Architecture], Section "Security" for more about this.)

11.3. 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, whereever 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 8.1), 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 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]

11.4. 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 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 isa 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.

11.5. {{Any others?}}

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 the 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

{{To be written.}}

12.2.3. Outdated Mappings - No Longer an ETR

If the destination of traffic from an ITR is no longer an ETR, one might get an ICMP Destination Unreachable error message. However, one cannot depend on that. The following mechanism will work, though.

Since the destination is not an ETR, the echoing reachability detection mechanism (see Section 8.3.1) 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 re-map may be requested.

12.3. Erroneous mappings

{{To be written.}}

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 (Section 7.3), 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), 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. Acknowledgments

The author would like thank 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 especially Joel Halpern for their careful review of, and helpful suggestions for, this document. Grateful thanks also to Darrel Lewis for his help with material on non-Internet uses of LISP, and to Vince Fuller for help with XML.

A final thanks is due to John Wrocklawski for the author's organizational affiliation. This memo was created using the xml2rfc tool

14. IANA Considerations

This document makes no request of the IANA.

15. Security Considerations

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

16. References

16.1. 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.

[Architecture] J.N. Chiappa, "The Architecture of the LISP Location-Identity Separation System", draft-chiappa-lisp-architecture-00 (work in progress), July 2012.

[DDT] V. Fuller, D. Lewis, and D. Farinacci, "LISP Delegated Database Tree", draft-fuller-lisp-ddt-01 (work in progress), March 2012.

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

[Interworking] D. Lewis, D. Meyer, D. Farinacci, and V. Fuller, "Interworking LISP with IPv4 and IPv6", draft-ietf-lisp-interworking-06 (work in progress), March 2012.

[LISP] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-23 (work in progress), May 2012.

[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, "LISP for Multicast Environments", draft-ietf-lisp-multicast-14 (work in progress), February 2012.

[NAT] V. Ermagan, D. Farinacci, D. Lewis, J. Skriver, F. Maino, and C. White, "NAT traversal for LISP", draft-ermagan-lisp-nat-traversal-01 (work in progress), March 2012.

[Versioning] L. Iannone, D. Saucez, and O. Bonaventure, "LISP Mapping Versioning", draft-ietf-lisp-map-versioning-09 (work in progress), March 2012.

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

16.2. 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.

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

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

[RFC5855] J. Abley and T. Manderson, "Nameservers for IPv4 and IPv6 Reverse Zones", RFC 5855, May 2010.

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

[ALT] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "LISP Alternative Topology (LISP-ALT)", draft-ietf-lisp-alt-10 (work in progress), December 2011.

[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.

[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.

[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.

[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.

[Salvadori] M. Salvadori and M. Levy, "Why Buildings Fall Down", W. W. Norton, New York, pg. 81, 1992.

Appendix A. Glossary/Definition of Terms

- Address
- Locator
- EID
- RLOC
- ITR
- ETR
- xTR

- PITR
- PETR
- MR
- MS
- DFZ

Appendix B. Other Appendices

Possible appendices:

-- Location/Identity Separation Brief History
-- LISP History
-- Old models (LISP 1, LISP 1.5, etc)

Author's Address

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

EMail: jnc@mit.edu