LISP Working Group J. N. Chiappa
Internet-Draft Yorktown Museum of Asian Art
Intended status: Informational October 15, 2012
Expires: April 18, 2013

An Architectural Perspective on the LISP Location-Identity Separation System
draft-ietf-lisp-architecture-00

Abstract

LISP upgrades the architecture of the IPvN internetworking system by separating location and identity, current 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.

This document gives additional architectural insight into LISP, and considers a number of aspects of LISP from a high-level standpoint.

Status of This Memo

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Table of Contents

1. Introduction

This document begins by introducing some high-level architectural perspectives which have proven useful for thinking about the LISP location-identity separation system. It then discusses some architectural aspects of LISP (e.g. its namespaces). The balance (and bulk) of the document contains architectural analysis of the LISP system; that is, it reviews from a high-level standpoint various aspects of that system; e.g. its scalability, security, robustness, etc.

NOTE: This document assumes a fair degree of familiarity with LISP; in particular, the reader should have a good ’high-level’ understanding of the overall LISP system architecture, such as is provided by [Introduction], "An Introduction to the LISP System".

By "system architecture" above, the restricted meaning used there is: ’How the system is broken up into subsystems, and how those subsystems interact; when does information flows from one to another, and what that information is.’ There is obviously somewhat more to architecture (e.g. the namespaces of a system, in particular their syntax and semantics), and that remaining architectural content is covered here.

2. Goals of LISP

As previously stated in the abstract, broadly, the goal of LISP is to be a practically deployable architectural upgrade to IPvN which performs separation of location and identity. But what is the value of that? What will it allow us to do?

The answer to that obviously starts with the things mentioned in the "Initial Applications" section of [Introduction], but there are other, longer-range (and broader) goals as well.

2.1. Reduce DFZ Routing Table Size

One of the main design drivers for LISP, as well as other location-identity separation proposals, is to decrease the overhead of running global routing system. In fact, it was this aspect that led the IRTF Routing RG to conclude that separation of location and identity was a key architectural underpinning needed to control the growth of the global routing system. [RFC6115]

As noted in [Introduction], many of the practical needs of Internet users are today met with techniques that increase the load on the global routing system (Provider Independent addresses for the provision of provider independence, multihoming, etc; more-specific routes for TE; etc.) Provision of these capabilities by a mechanism which does not involve extra load on the global routing system is therefore very desirable.

A number of factors, including the use of these techniques, has led to a great increase in the fragmentation of the address space, at least in terms of routing table entries. In particular, the growth in demand for multi-homing has been forseen as driving a large increase in the size of the global routing tables.

In addition, as the IPv4 address space becomes fuller and fuller, there will be an inevitable tendency to find use in smaller and smaller ’chunks’ of that space. [RFC6127] This too would tend to increase the size of the global routing table.

LISP, if successful and widely deployed, offers an opportunity to use separation of location and identity to control the growth of the size of the global routing table. (A full examination of this topic is beyond the scope of this document - see {{find reference}}.)

2.2. Deployment of New Namespaces

Once the mapping system is widely deployed and available, it should make deployment of new namespaces (in the sense of new syntax, if not new semantics) easier. E.g. if someone wishes in the future to devise a system which uses native MPLS [RFC3031] for a data carriage system joining together a large number of xTRs, it would easy enough to arrange to have the mappings for destinations attached to those xTRs abe some sort of MPLS-specific name.

More broadly, the existence of a binding layer, with support for multiple namespace built into the interface on both sides (see Section 5) is a tremendously powerful evolutionary tool; one can introduce a new namespace (on one side) more easily, if it is mapped to something which is already deployed (on the other). Then, having taken that step, one can invert the process, and deploy yet another new namespace, but this time on the other.

2.3. Future Development of LISP

Speculation about long-term future developments which are enabled by the deployment of LISP is not really proper for this document. However, interested readers may wish to consult [Future] for one person’s thoughts on this topic.

3. Architectual Perspectives

This section contains some high-level architectural perspectives which have proven useful in a number of ways for thinking about LISP. For one, when trying to think of LISP as a complete system, they provide a conceptual structure which can aid analysis of LISP. For another, they can allow the application of past analysis of, and experience with, similar designs.

3.1. Another Packet-Switching Layer

When considering the overall structure of the LISP system at a high level, it has proven most useful to think of it as another packet-switching layer, run on top of the original internet layer - much as the Internet first ran on top of the ARPANET.

All the functions that a normal packet switch has to undertake - such as ensuring that it can reach its neighbours, and they they are still up - the devices that make up the LISP overlay also have to do, along the ’tunnels’ which connect them to other LISP devices.

There is, however, one big difference: the fanout of a typical LISP ITR will be much larger than most classic physical packet switches. (ITRs only need to be considered, as the LISP tunnels are all effectively unidirectional, from ITR to ETR - an ETR needs to keep no per-tunnel state, etc.)

LISP is, fundamentally, a ’tunnel’ based system. Tunnel system designs do have their issues (e.g. the high inter-’switch’ fan-out), but it’s important to realize that they also can have advantages, some of which are listed below.

3.2. ’Double-Ended’ Approach

LISP may be thought of as a ’double-ended’ approach to enhancing the architecture, in that it uses pairs of devices, one at each end of a communication stream. In particular, to interact with the population of ’legacy’ hosts (which will be, inevitably, the vast majority, in the early stages of deployment) it requires a LISP device at both ends of the ’tunnel’.

This is in distinction to, say, NAT systems ([RFC1631]), which only need a device deployed at one end: the host at the other end doesn’t need a matching device at its end to massage the packets, but can simply consume them on its own, as any packets it receives are fully normal packets. This allows any site which deploys such a ’single-ended’ device to get the full benefit, whilst acting entirely on its own. [Wasserman]

The issue is not that LISP uses tunnels. Designs like HIP ([RFC4423]) and ILNP ([ILNP]), which do not involve tunnels, inhabit a similar space to tunnel-based designs like LISP, in that unless both ends are upgraded - or there is a proxy at the un-upgraded end - one doen’t get any benefits. So it’s really not the tunnel which is the key aspect, it’s the ’all at one end’ part which is key. Whether the system is tunnel, versus non-tunnel, is not that important.

However, the double-ended approach of LISP does have advantages, as well as costs. To put it simply, the ’feature’ of the alternative approach, that there’s only a box at one end, has a ’bug’: there’s only a box at one end. There are things which such a design cannot accomplish, because of that.

To put it another way, does the fact that the packet thus necessarily has only a single ’name’ in it for the entities at each end (i.e. the IPvN source and destination addresses), because it is a ’normal’ packet, present a limitation? Put that way, it would seem natural that it should cause certain limits.

To compile a complete list of the things that can be done, when two separate ’names’ are in the packet, is beyond the scope of this document. However, one example of the kind of thing that can be done is mobility with open connections, without needing to ’triangle route’ the packets through some sort of ’base station’ at the original location. Another is that is possible to automatically tunnel IPv6 traffic over IPv4 infrastructure, or vice versa, invisibly to the hosts on both ends.

In the longer term, having having tunnel boxes will allow (and is allowing) us to explore other kinds of wrappings. For example, we can transport ’raw’ local-network packets (such as Ethernet MAC frames) across an IPvN infrastructure.

One could also wrap packets in non-IPvN formats: perhaps to take direct advantage of the capabilities of underlying switching fabrics (e.g. MPLS [RFC3031]); perhaps to deploy new carriage protocols, etc, where non-standard packet formats will allow extended semantics.

4. Architectual Aspects

LISP does take some novel architectural approaches in a number of ways: e.g. its use of a separate mapping system, etc, etc. This section contains some commentary on some of the high-level architectural aspects of LISP.

4.1. Critical State

LISP does have ’critical state’ in the network (i.e. state which, if if lost, causes the communication to fail). However, because LISP is designed as an overall system, ’designing it in’ allows for a ’systems’ approach to its state issues. In LISP, this state has been designed to be maintained in an ’architected’ way, so it does not produce systemic brittleness in the way that the state in NATs does.

For instance, throughout the system, provisions have been made to have redundant copies of state, in multiple devices, so that the loss of any one device does not necessarily cause a failure of an ongoing connection.

4.2. Need for a Mapping System

LISP does need to have a mapping system, which brings design, implementation, configuration and operational costs. Surely all these costs are a bad thing? However, having a mapping system have advantages, especially when there is a mapping layer which has global visibility (i.e. other entities know that it is there, and have an interface designed to be able to interact with it). This is unlike, say, the mappings in NAT, which are ’invisible’ to the rest of the network.

In fact, one could argue that the mapping layer is LISP’s greatest strength. Wheeler’s Axiom* (’Any problem in computer science can be solved with another level of indirection’) indicates that the binding layer available with the LISP mapping system will be of great value. Again, it is not the job of this document to list them all - and in any event, there is no way to forsee them all.

The author of this document has often opined that the hallmark of great architecture is not how well it does the things it was designed to do, but how well it does things it was never expected to have to handle. Providing such a powerful and generic binding layer is one sure way to achieve the sort of lasting flexibility and power that leads to that outcome.

[Footnote *: This Axiom is often mis-attributed to Butler Lampson, but Lampson himself indicated that it came from David Wheeler.]

4.3. Piggybacking of Control on User Data

LISP piggybacks control transactions on top of user data packets. This is a technique that has a long history in data networking, going back to the early ARPANET. [McQuillan] It is now apparently regarded as a somewhat dubious technique, the feeling seemingly being that control and user data should be strictly segregated.

It should be noted that _none_ of the piggybacking of control functionality in LISP is _architecturally fundamental_ to LISP. All of the functions in LISP which are performed with piggybacking could be performed almost equally well with separate control packets.

The "almost" is solely because it would cause more overhead (i.e. control packets); neither the response time, robustness, etc would necessarily be affected - although for some functions, to match the response time observed using piggybacking on user data would need as much control traffic as user data traffic.

This technique is particularly important, however, because of the issue identified at the start of this section - the very large fanout of the typical LISP switch. Unlike a typical router, which will have control interactions with only a few neighbours, a LISP switch could eventually have control interactions with hundreds, or perhaps even thousands (for a large site) of neighbours.

Explicit control traffic, especially if good response times are desired, could amount to a very great deal of overhead in such a case.

5. Namespaces

One of the key elements in any architecture, or architectural analysis, are the namespaces involved: what are their semantics and syntax, what are the kinds of things they name, etc.

LISP has two key namespace, EIDs and RLOCs, but it must be emphasized that on an architectural level, neither the syntax, or, to a lesser degree, the semantics, of either are absolutely fixed. There are certain core semantics which are generaly unchanging (such as the notion that EIDs provide only identity, whereas RLOCs provide location), but as we will see, there is a certain amount of flexibility available for the long-term.

In particular, all of LISP's key interfaces always include an Address Family Identifier (AFI) [AFI] for all names, so that new forms can be introduced at any time the need is felt. Of course, in practise such an introduction would not be a trivial exercise - but neither is is impossibly painful, as is the case with IPv4's 32-bit addresses, which are effectively impossible to upgrade.

5.1. LISP EIDs

A 'classic' EID is defined as a subset of the possible namespaces for endpoints. [Chiappa] Like most 'proper' endpoint names, as proposed there, they contain contain no information about the location of the endpoint. EIDs are the subset of possible endpoint names which are: fixed length, 'reasonably' short', binary (i.e. not intended for direct human use), globally unique (in theory), and allocated in a top-down fashion (to achieve the former).

LISP EIDs are, in line with the general LISP deployment philosophy, a reuse of something already existing - i.e. IPvN addresses. For those used as in LISP as EIDs, LISP removes much (or, in some cases, all) of the location-naming function of IPvN addresses.

In addition, the goal is to have EIDs name hosts (or, more properly, their end-end communication stacks), whereas the other LISP namespace group (RLOCs) names interfaces. The idea is not just to have two namespaces (with different semantics), but also to use them to name _different classes of things_ - classes which currently do not have clearly differentiated names. This should produce even more functionality.

5.1.1. Residual Location Functionality in EIDs

LISP retains, especially in the early stages of the deployment, in many cases some residual location-naming functionality in EIDs, This is to allow the packet to be correctly routed/forwarded to the destination node, once it has been unwrapped by the ETR - and this is a direct result of LISP's deployment philosophy (see [Introduction], Section "Deployment").

Clearly, if there are one or more unmodified routers between the ETR and the desination node, those routers will have to perform a routing step on the packet, for which it will need _some_ information as to the location of the destination.

One can thus view such LISP EIDs, which retain 'stub' location information, as 'addresses' (in the definition of the generic sense of this term, as used here), but with the location information restricted to a limited, local scope.

This retention of some location functionality in LISP EIDs, in some cases, has led some people to argue that use of the name 'EID' is improper. In response, it was suggested that LISP use the term 'LEID', to distinguish LISP's 'bastardized' EIDs from 'true' EIDs, but this usage has never caught on.

It has also been suggested that one usage mode for LISP EIDs, in existing software loads, is to assign them as the address on an internal virtual interface; all the real interfaces would have RLOCs only. [Templin] This would make such LISP EIDs functionally equivalent to 'real' EIDs - they are names which are purely identity, have no location information of any kind in them, and cannot be used to make any routing decisions anywhere outside the host.

It is true that even in such cases, the EID is still not a 'pure' EID, as it names an interface, not the end-end stack directly. However, to do a perfect job here (or on separation of location and identity) is impossible without modifying existing hosts (which are, inevitably, almost always one end of an end-end communication) - and that has been ruled out, for reasons of viable deployment.

The need for interoperation with existing unmodified hosts limits the semantic changes one can impose, much as one might like to provide a cleaner separation. (Future evolution can bring us toward that state, however: see [Future].)

5.2. RLOCs

RLOCs are basically pure 'locators' [RFC1992], although their syntax and semantics is restricted at the moment, because in practise the only forms of RLOCs supported are IPv4 and IPv6.

5.3. Overlapping Uses of Existing Namespaces

It is in theory possible to have a block of IPvN namespace used as both EIDs and RLOCs. In other words, EIDs from that block might map to some other RLOCs, and that block might also appear in the DFZ as the locators of some other ETRs.

This is obviously potentially confusing - when a 'bare' IPvN address from one of these blocks, is it the RLOC, or the EID? Sometimes it it obvious from the context, but in general one could not simply have a (hypothetical) table which assigns all of the address space to either 'EID' or 'RLOC'.

In addition, such usage will not allow interoperation of the sites named by those EIDs with legacy sites, using the PITR mechanism ([Introduction], Section "Proxy Devices"), since that mechanisms depends on advertizing the EIDs into the DFZ, although the LISP-NAT mechanism should still work ([Introduction], Section "LISP-NAT").

Nevertheless, as the IPv4 namespace becomes increasingly used up, this may be an increasingly attractive way of getting the 'absolute last drop' out of that space.

5.4. LCAFs

{{To be written.}}

--- Key-ID
--- Instance-IDs

6. Scalability

As with robustness, any global communication system must be scalable, and scalable up to almost any size. As previously mentioned (xref target="Perspectives-Packet"/), the large fanouts to be seen with LISP, due to its 'overlay' nature, present a special challenge.

One likely saving grace is that as the Internet grows, most sites will likely only interact with a limited subset of the Internet; if nothing else, the separation of the world into language blocks means that content in, say, Chinese, will not be of interest to most of the rest of the world. This tendency will help with a lot of things which could be problematic if constant, full, N^2 connectivity were likely on all nodes; for example the caching of mappings.

6.1. Demand Loading of Mappings

One question that many will have about LISP's design is 'why demand-load mappings - why not just load them all'? It is certainly true that with the growth of memory sizes, the size of the complete database is such that one could reasonably propose keeping the entire thing in each LISP device. (In fact, one proposed mapping system for LISP, named NERD, did just that. [NERD])

A 'pull'-based system was chosen over 'push' for several reasons; the main one being that the issue is not just the pure _size_ of the mapping database, but its _dynamicity_. Depending on how often mappings change, the update rate of a complete database could be relatively large.

It is especially important to realize that, depending on what (probably unforseeable) uses eventually evolve for the identity->location mapping capability LISP provides, the update rate could be very high indeed. E.g. if LISP is used for mobility, that will greatly increase the update rate. Such a powerful and flexible tool is likely be used in unforseen ways (Section 4.2), so it's unwise to make a choice that would preclude any which raise the update rate significantly.

Push as a mechanism is also fundamentally less desirable than pull, since the control plane overhead consumed to load and maintain information about unused destinations is entirely wasted. The only potential downside to the pull option is the delay required for the demand-loading of information.

(It's also probably worth noting that many issues that some people have with the mapping approach of LISP, such as the total mapping database size, etc are the same - if not worse - for push as they are for pull.)

Finally, for IPv4, as the address space becomes more highly used, it will become more fragmented - i.e. there will tend to be more, smaller, entries. For a routing table, which every router has to hold, this is problematic. For a demand-loaded mapping table, it is not bad. Indeed, this was the original motivation for LISP ([RFC4984]) - although many other useful and desirable uses for it have since been enumerated (see [Introduction], Section "Applications").

For all of these reasons, as long as there is locality of reference (i.e. most ITRs will use only a subset of the entire set), it makes much more sense to use the a pull model, than the classic push one heretofore seen widely at the internetwork layer (with a pull approach thus being somewhat novel - and thus unsettling to many - to people who work at that layer).

It may well be that some sites (e.g. large content providers) may need non-standard mechanisms - perhaps something more of a ’push’ model. This remains to be determined, but it is certainly feasible.

6.2. Caching of Mappings

It should be noted that the caching spoken of here is likely not classic caching, where there is a fixed/limited size cache, and entries have to be discarded to make room for newly needed entries. The economics of memory being what they are, there is no reason to discard mappings once they have been loaded (although of course implementations are free to chose to do so, if they wish to).

This leads to another point about the caching of mappings: the algorithms for management of the cache are purely a local issue. The algorithm in any particular ITR can be changed at will, with no need for any coordination. A change might be for purposes of experimentation, or for upgrade, or even because of environmental variations - different environments might call for different cache management strategies.

The local, unsynchronized replacability of the cache management scheme is the architectural aspect of the design; the exact algorithm, which is engineering, is not.

6.3. Amount of State

{{To be written.}} [Iannone]

-- Mapping cache size
--- Mention studies
-- Delegation cache size (in MRs)
--- Mention studies
-- Any others?

6.4. Scalability of The Indexing Subsystem

LISP initially used an indexing subsystem 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. [DDT]

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]

From an architectural point of view, the main advantage of DDT is that it enables client side caching of information about intermediate nodes in the resolution hierarchy, and also enables direct communication with them. As a result, DDT has much better scaling properties than ALT.

The most important result of this change is that it avoids a concentration of resolution request traffic at the root of the indexing tree, a problem which by itself made ALT unsuitable for a global-scale system. The problem of root concentration (and thus overload) is almost unavoidable in ALT (even if masses of ’bypass’ links are created).

ALT’s scalability also depends on enforcing an intelligent organization that aincreases aggregation. Unfortunately, the current backbone routing BGP system shows that there is a risk of an organic growth of ALT, one which does not achieve aggregation. DDT does not display this weakness, since its organization is inherently hierarchical (and thus inherently aggregable).

The hierarchical organization of DDT also reduces the possibility for a configuration error which interferes with the operation of the network (unlike the situation with the current BGP DFZ). DDT security mechanisms can also help produce a high degree of robustness, both against misconfiguration, and deliberate attack. The direct communication with intermediate nodes in DDT also helps to quickly locate problems when they occur, resulting in better operational characteristics.

Next, since in ALT mapping requests must be transmitted through an overlay network, a significant share of requests can see substantially increased latencies. Simulation results in the TREE work clearly showed, and quantified, this effect.

The simulations also showed that the nodes composing the ALT and DDT networks for a mapping database of full Internet size could have thousands of neighbours. This is not an issue for DDT, but would almost certainly have been problematic for ALT nodes, since handling that number of simultaneous BGP sessions would likely to be difficult.

7. Security

LISP does not yet have an overarching security architecture. Many parts of the system have been hardened, but more on a case-by case basis, rather than from an overall perspective. (This is in part due to the ’just enough’ approach to security initially taken in LISP; see [Introduction], Section "Just Enough Security".)

This section represents an attempt to produce a more broadly-based view of security in LISP; it mostly resulted from an attempt to add security to the DDT indexing system ([DDT]), but the analysis is is general enough to apply to LISP broadly.

The _good_ thing about the Internet is that it brings the world to your doorstep - masses of information from all around the world are instantly available on your computing device. The _bad_ thing about the Internet is that it brings the world to your doorstep - including legions of crackers, thieves, and general scum and villainy. Thus, any node may be the target of fairly sophisticated attack - often automated (thereby reducing the effort required of the attacker to spread their attack as broadly as possible).

Security in LISP faces many of the same challenges as security for other parts of the Internet: good security usually means work for the users, but without good security, things are vulnerable.

The Internet has seen many very secure systems devised, only to see them fail to reach wide adoption; the reasons for that are complex, and vary, but being too much work to use is a common thread. It is for this reason that LISP attempts to provide ’just enough’ security (see [Introduction], Section "Just Enough Security").

7.1. Basic Philosophy

To square this circle, of needing to have very good security, but of it being too difficult to use very good security, the general concept is for LISP to have a series of ’graded’ security measures available, with the ’ultimate’ security mechanisms being very high-grade indeed.

The concept is to devise a plan in which LISP can simultaneously attempt to have not just ’ultimate’ security, but also one or more ’easier’ modes, ones which will be easier to configure and use. This ’easier’ mode can be both an interim system (with the full powered system available for when it it needed), as well as the system used in sections of the network where security is less critical (following the general rule that the level of any security should generally be matched to what is being protected).

The challenge is to do this in a way that does not make the design more complex, since it has to include both the ’full strength’ mechanism(s), and the ’easier to configure’ mechanism(s). This is one of the fundamental tradeoffs to struggle with: it is easy to provide ’easier to configure’ options, but that may make the overall design more complex.

As far as making it hard to implement to begin with (also something of a concern initially, although obviously not for the long term): we can make it ’easy’ to deploy initially by simply not implementing/ configuring the heavy-duty security early on. (Provided, of course, that the packet formats, etc, needed to support such security are all included in the design to begin with.)

7.2. Design Guidance

In designing the security, there are a small number of key points that will guide the design:

- Design lifetime
- Threat level

How long is the design intended to last? If LISP is successful, a minimum of a 50-year lifetime is quite possible. (For comparison, IPv4 is now 34 at the time of writing this, and will be around for at least several decades yet, if not longer; DNS is 28, and will probably last indefinitely.)

How serious are the threats it needs to meet? As mentioned above, the Internet can bring the worst crackers from anywhere to any location, in a flash. Their sophistication level is rising all the time: as the easier holes are plugged, they go after others. This will inevitably eventually require the most powerful security mechanisms available to counteract their attacks.

Which is not to say that LISP needs to be that secure _right away_. The threat will develop and grow over a long time period. However, the basic design has to be capable of being _securable_ to the expanded degree that will eventually be necessary. However, _eventually_ it will need to be as securable as, say, DNS - i.e. it _can_ be secured to the same level, although people may chose not to secure their LISP infrastructure as well as DNSSEC potentially does. [RFC4033]

In particular, it should be noted that historically many systems have been broken into, not through a weakness in the algorithms, etc, but because of poor operational mechanics. (The well-known 'Ultra' breakins of the Allies were mostly due to failures in operational procedure. [Welchman]) So operational capabilities intended to reduce the chance of human operational failure are just as important as strong algorithms; making things operationally robust is a key part of 'real' security.

7.2.1. Security Mechanism Complexity

Complexity is bad for several reasons, and should always be reduced to a minimum. There are three kinds of complexity cost: protocol complexity, implementation complexity, and configuration complexity. We can further subdivide protocol complexity into packet format complexity, and algorithm complexity. (There is some overlap of algorithm complexity, and implementation complexity.)

We can, within some limits, trade off one kind of complexity for others: e.g. we can provide configuration _options_ which are simpler for the users to operate, at the cost of making the protocol and implementation complexity greater. And we can make initial (less capable) implementations simpler if we make the protocols slightly more complex (so that early implementations don't have to implement all the features of the full-blown protocol).

It's more of a question of some operational convenience/etc issues - e.g. 'How easy will it be to recover from a cryptosystem compromise'. If we have two ways to recover from a security compromise, one which is mostly manual and a lot of work, and another which is more automated but makes the protocol more complicated, if compromises really are very rare, maybe the smart call _is_ to go with the manual thing - as long as we have looked carefully at both options, and understood in some detail the costs and benefits of each.

7.3. Security Overview

First, there are two different classes of attack to be considered: denial of service (DoS, i.e. the ability of an intruder to simply cause traffic not to successfully flow) versus exploitation (i.e. the ability to cause traffic to be 'highjacked', i.e. traffic to be sent to the wrong location).

Second, one needs to look at all the places that may be attacked. Again, LISP is a relatively simple system, so there are not that many parts to examine. The following are the things we need to secure:

- Lookups
- Indexing
- Mappings

7.3.1. Securing Lookups

{{To be written.}} Nonces, [SecurityReq]

7.3.2. Securing The Indexing Subsystem

It is envisioned that DDT will be highly securable, with all the delegations cryptographiclly secured via public-private signatures, very similar to the way DNS is ([RFC4033]).

The detailed mechanisms will be based on DNS's; this has the obvious benefit that all the lessons of DNS's years of practical experience with deployment, operations, etc, as well as the improvements to the basic design of DNS Security to provide a secure but usable system can be taken into account. However, DDT's security will also apply the thinking above, about making a 'versio' which is easier to use available.

{{To be written.}}

7.3.3. Securing Mappings

There are two approaches to securing the provision of mappings. The first, which is of course not completely satisfactory, is to only secure the channel between the ITR and the entities involved in providing mappings for it. (See above, Section 7.3.1)

The second is to secure the mappings themselves, by signing them 'at birth' (much the same way in which DNS Security operates). [RFC4033]. There was an attempt early on to suggest such a system for LISP ([SecurityAuth]), but it was not adopted (although the particular proposal was rather complex).

In the long run, the latter approach would obviously be superior, since it would be almost immune to any compromises of the mapping distribution system. {{Tie-in to space allocation security}}

7.4. Securing the xTRs

```
--- Cache management
--- Unsoliticed Map-Replies are _very bad_ - must go through
    mapping system to verify that the sender is authoritative for
    that range of EIDs
```

8. Robustness

```
-- Depends on deployment as well as design
-- Architected, visible replication of state/data
-- Overlapping mechanisms (ref redundancy as key for robustness)
```

9. Fault Discovery/Handling

Any global communication system must be robust, and to be robust, it must be able to discover and handle problems. LISP's general philosophy of robustness is usually to have overlapping, simple mechanisms to discover and repair problems.

10. Optimization

```
-- Philosophy
-- Piggybacking
-- 'Wiretapping' return mappings
--- Security is an issue on that
```

11. Open Issues

Although much work has been done on LISP, and it operates satisfactorily in a reasonably large initial deployment, there are a few potentially problematic issues which remain. It is not clear if they will be issues which need to be dealt, since they have not proven to be obstacles so far, but it is worth listing them.

We can divide them in _local_ issues, i.e. ones which can be solved on a node-by-node basis, without requiring co-ordinated change, and systemic issues, which are obviously more problematic, since they could require co-ordinated changes to the protocols.

11.1. Local Open Issues

11.1.1. Missing Mapping Packet Queueing

Currently, some (all?) ITRs discard packets when they need a mapping, but have not loaded one yet, thereby causing the applicaton to have to retransmit their opening packet. True, many ARP implementations use the same strategy, but the average APR cache will only ever contain a few mappings, so it will not be so noticeable as with the mapping cache in an ITR, which will likely contain thousands.

Obviously, they could queue the packets while waiting to load the mapping, but this presents a number of subtle implementation issues: the ITR must make sure that it does not queue too many packets, etc.

In particular, if such packets are queued, this presents a potential DoS attack vector, unless the code is carefully written with that possibility in mind.

11.1.2. Mapping Cache Management Algorithm

Relatively little work has been done on sophisticated mapping cache management algorithms; in particular, the issue of which mapping(s) to drop if the cache reaches some maximum allowed size.

This particular issue has also been identified as another potential DoS attack vector.

11.2. Systemic Open Issues

11.2.1. Mapping Database Provider Lock-in

This refers to the fact that if one does not like the entity which is providing the indexing for the part of the address space which one's EIDs are allocated out of, there isn't probably isn't any way to switch to an alternative provider.

It is not clear that this is a real probem, though - the fact that all DNS top-level zones only have a single registry has not been a problem, nor has the fact that if one doesn't like the service the registry offers, one can't take one's DNS name to another registry.

Doing anything about it would also be difficult. Although it is _technically_ possible to duplicate any node in the delegation tree, and in theory such duplicates could be provided by different providers, it is not clear that such an arrangement would make _business_ sense.

For instance, if the holder of 10.1.1/24 decides they do not like the entity providing indexing for 10.1/16 (call them E1), and ask another entity (E2) to provide alternative service for 10.1/16, two problems arise. First, E1 is _still_ going to have to maintain the correct data for 10.1.1/24, and response to queries asking about them. Second, E2 will similarly have to maintain data for, and reply to queries about, all the other space-holders in 10.1/16 - even though they will likely not have any business relationship with them.

11.2.2. Automated ETR Synchronization

LISP requires that all the ETRs which are authoritative for the mappings for a particular address block return the same mapping data. In particular, their idea of the ’liveness’ of all the ETRs should be identical, and correct.

At the moment, this is mostly a manual process, although liveness information can be currently be gathered from some IGPs.

11.2.3. EID Reachability

At the moment, LISP assumes that if an ETR is reachable from a given ITR, all destination EIDs behind that ETR are reachable from that ETR. There is no way to detect if any are not, nor to switch to an alternate ETR.

It is not clear that this is a problem that needs attention. The same has been true for all border routers for many years now, and there does not seem to be any general mechanism to deal with it (Although some BGP implementations may advertize changes in reachability status if what they are seeing from their IGP changes.)

11.2.4. Detect and Avoid Broken ETRs

{{To be written}}

12. 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 Joel Halpern for their careful review of, and helpful suggestions for, this document. Grateful thanks also 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

13. IANA Considerations

This document makes no request of the IANA.

14. Security Considerations

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

15. References

15.1. Normative References

[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.

[Introduction] J. N. Chiappa, "An Introduction to the LISP Location-Identity Separation System", draft-ietf-lisp-introduction-00 (work in progress), October 2012.

[SecurityAuth] R. Gagliano, "A Profile for Endpoint Identifier Origin Authorizations (IOA)", draft-rgaglian-lisp-iao-00 (work in progress), March 2009.

[SecurityReq] F. Maino, V. Ermagan, A. Cabellos, D. Saucez, and O. Bonaventure, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-02 (work in progress), March 2012.

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

15.2. Informative References

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

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

[RFC3031] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

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

[RFC4423] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture", RFC 4423, May 2006.

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

[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’.

[RFC6127] J. Arkko and M. Townsley, "IPv4 Run-Out and IPv4-IPv6 Co-Existence Scenarios", RFC 6127, May 2011.

[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.

[NERD] E. Lear, "NERD: A Not-so-novel EID to RLOC Database", draft-lear-lisp-nerd-09 (work in progress), April 2012.

[ILNP] R.J. Atkinson and S.N. Bhatti, "ILNP Architectural Description", draft-irtf-rrg-ilnp-arch-05 (work in progress), May 2012.

[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>.

[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.

[McQuillan] J. M. McQuillan, W. R. Crowther, B. P. Cosell, D. C. Walden, and F. E. Heart, "Improvements in the Design and Performance of the ARPA Network", Proceedings AFIPS 1972 FJCC, Vol. 40, pp. 741-754.

[Templin] F. Templin, "LISP WG", LISP WG list message, Message-ID: 39C363776A4E8C4A94691D2BD9D1C9A105B0AC71@XCH-NW-7V2.nw.nos.boeing.com, 13 March 2009,, <http://www.ietf.org/mail-archive/web/lisp/current/msg00269.html>.

[Wasserman] M. Wasserman, "IPv6 networking: Bad news for small biz", IETF list message, Message-Id: D11C4A34-7362-423E-A60E-476FC5D61D37@lilacglade.org, 5 April 2012, <https://www.ietf.org/ibin/c5i?mid=6&rid=49&gid=0&k1=933&k2=62733&tid=1340933524>.

[Welchman] G. Welchman, "The Hut Six Story", Allen Lane, London, pg. 3, 1982.

A truly monumental book; the ground it covers ranges from his work helping break German codes in World War II to his experience with securing data packet networks!

Appendix A. Glossary/Definition of Terms

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

- xTR
- PITR
- PETR
- MR
- MS
- DFZ

Appendix B. Other Appendices

- -- Location/Identity Separation Brief History
- -- LISP History
- -- Old models (LISP 1, LISP 1.5, etc)
- -- Different mapping distribution models (e.g. LISP-NERD)
- -- Different mapping indexing models (LISP-ALT forwarding/overlay model), LISP-TREE DNS-based, LISP-CONS)

Author's Address

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

EMail: jnc@mit.edu