Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 22, 2013

J. Arango
S. Venaas
I. Kouvelas
Cisco Systems

February 18, 2013

PIM Join Attributes for LISP Environments
draft-arango-pim-join-attributes-for-lisp-00.txt

Abstract

This document defines two PIM Join/Prune attributes that support the construction of multicast distribution trees where the root and receivers are located in different LISP sites. These attributes allow the receiver site to select between unicast and multicast transport and to convey the receiver RLOC address to the control plane of the root xTR.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The construction of multicast distribution trees where the root and receivers are located in different LISP sites [RFC6830] is defined in [RFC6831]. Creation of (root-EID,G) state in the root site requires that unicast LISP-encapsulated Join/Prune messages be sent from an xTR on the receiver site to an xTR on the root site.

[RFC6831] specifies that (root-EID,G) data packets are to be LISP-encapsulated into (root-RLOC,G) multicast packets. However, a wide deployment of multicast connectivity between LISP sites is unlikely to happen any time soon. In fact, some implementations are initially focusing on unicast transport with head-end replication between root and receiver sites.

The unicast LISP-encapsulated Join/Prune message specifies the (root-EID,G) state that needs to be established in the root site, but conveys nothing about the receivers capability or desire to use multicast as the underlying transport. This document specifies a Join/Prune attribute that allows the receiver to select the desired transport.

Knowledge of the receiver RLOC is also essential to the control plane of the root xTR. It determines the downstream destination for unicast head-end replication and identifies the receiver xTR that needs to be notified should the root of the distribution tree move to another site.

The outer source address field of the encapsulated Join/Prune message contains an RLOC address of the receiver xTR. This source address is message to the root xTR RLOC destination. Due to policy and load balancing considerations, the selected source address may not be the RLOC on which the receiver site wishes to receive a particular flow. This document specifies a Join/Prune attribute that conveys the appropriate receiver RLOC address to the control plane of the root xTR.

2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. PIM Join/Prune Attributes

PIM Join/Prune attributes are defined in [RFC5384] by introducing a new Encoded-Source type that, in addition to the Join/Prune source, can carry multiple type-length-value (TLV) attributes. These attributes apply to the individual Join/Prune sources on which they are stored.

The attributes defined in this document conform to the format of the encoding type defined in [RFC5384]. The attributes would typically be the same for all the sources in the Join/Prune message. Hence we RECOMMEND using the hierarchical Join/Prune attribute scheme defined in [I-D.venaas-pim-hierarchicaljoinattr]. This hirarchichal system allows attributes to be conveyed on the Upstream Neighbor Address field, thus enabling the efficient application of a single attribute instance to all the sources in the Join/Prune message.

LISP xTRs do not exchange PIM Hello Messages and hence no Hello option is defined to negotiate support for these attributes. Systems that support unicast head-end replication are assumed to support these attributes.

4. The Transport Attribute

It is essential that a mechanism be provided by which the desired transport can be conveyed by receiver sites. Root sites with multicast connectivity will want to leverage multicast replication. However, not all receiver sites can be expected to have multicast connectivity. It is thus desirable that root sites be prepared to support (root-EID,G) state with a mixture of multicast and unicast output state. This document specifies a Join/Prune attribute that allows the receiver to select the desired underlying transport.

4.1. Transport Attribute Format

```
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|E| Type = 5  | Length = 1    |  Transport    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

F-bit: The Transitive bit. Specifies whether the attribute is transitive or non-transitive. MUST be set to zero. This attribute is ALWAYS non-transitive.

E-bit: End-of-Attributes bit. Specifies whether this attribute is the last. Set to zero if there are more attributes. Set to 1 if this is the last attribute.

Type: The Transport Attribute type is 5.

Length: The length of the Transport Attribute value. MUST be set to 1.

Transport: The type of transport being requested. Set to 0 for multicast. Set to 1 for unicast.

4.2. Using the Transport Attribute

Hierarchical Join/Prune attribute instances [I-D.venaas-pim-hierarchicaljoinattr] SHOULD be used when the same Transport Attribute is to be applied to all the sources within the Join/Prune message or all the sources within a group set. The root xTR MUST accept Transport Attributes in the Upstream Neighbor Encoded-Unicast address, Encoded-Group addresses, and Encoded-Source addresses.

There MUST NOT be more than one Transport Attribute within the same encoded address. If an encoded address has more than one instance of the attribute, the root xTR MUST discard all affected Join/Prune sources.

5. Receiver RLOC Attribute

The root xTR must know the receiver RLOC addresses of all receiver sites for a given (root-EID,G) so that it can perform unicast LISP-encapsulation of multicast data packets to each and every receiver site that has requested unicast head-end replication.

To support mobility of EIDs, the root xTR must keep track of ALL receiver RLOCs even when the corresponding downstream site has not requested unicast replication. The root xTR may detect that a local multicast source "root-EID" has moved to a remote LISP site. Under such circumstances LISP sends a SMR message to all receiver xTRs, prompting them to update their map cache. This is only possible if LISP can obtain from PIM the set of all receiver RLOCS that have active Join state for the root-EID.

The outer source address field of the encapsulated Join/Prune message contains an RLOC address of the receiver xTR. LISP xTRs, as edge devices, are commonly subject to URPF checks by the network providers on each core-facing interface. The source address for the encapsulation header must therefore be the RLOC of the core-facing interface used to physically transmit the encapsulated Join/Prune message. Due to policy and load balancing considerations, that may not be the RLOC on which the receiver site wishes to receive a particular flow. This document specifies a Join/Prune attribute that conveys the appropriate receiver RLOC address to the control plane of the root xTR.

5.1. Receiver RLOC Attribute Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|E| Type = 6  |    Length     |  Addr Family  |  Receiver RLOC
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...
```

F-bit: The Transitive bit. Specifies whether this attribute is transitive or non-transitive. MUST be set to zero. This attribute is ALWAYS non-transitive.

E-bit: End-of-Attributes bit. Specifies whether this attribute is the last. Set to zero if there are more attributes. Set to 1 if this is the last attribute.

Type: The Receiver RLOC Attribute type is 6.

Length: The length in octets of the attribute value. MUST be set to the length in octets of the receiver RLOC address plus one octet to account for the Address Family field.

Addr Family: The PIM Address Family of the receiver RLOC as defined in [RFC4601].

Receiver RLOC: The RLOC address on which the receiver xTR wishes to receiver the unicast-encapsulated flow.">

5.2. Using the Receiver RLOC Attribute

Hierarchical Join/Prune attribute instances [I-D.venaas-pim-hierarchicaljoinattr] SHOULD be used when the same Receiver RLOC attribute is to be applied to all the sources within the message or all the sources within a group set. The root xTR MUST accept Transport Attributes in the Upstream Neighbor Encoded-Unicast address, Encoded-Group addresses, and Encoded-Source addresses.

There MUST NOT be more than one Receiver RLOC Attribute within the same encoded address. If an encoded address has more than one instance of the attribute, the root xTR MUST discard all affected Join/Prune sources.

6. Security Considerations

Security of the Join Attribute is only guaranteed by the security of the PIM packet. The attributes specified herein do not enhance or diminish the privacy or authenticity of a Join/Prune message. A site that legitimately or maliciously sends and delivers a Join/Prune message to another site will equally be able to append these and any other attributes it wishes.

7. IANA Considerations

Two new PIM Join/Prune attribute types need to be assigned. Type 5 is being requested for the Transport Attribute. Type 6 is being requested for the Receiver RLOC Attribute.

8. Normative References

[AFI] IANA, "Address Family Numbers", http://www.iana.org/assignments/address-family-numbers.

[I-D.venaas-pim-hierarchicaljoinattr] Venaas, S., Kouvelas, I., and J. Arango, "Hierarchical Join/Prune Attributes", draft-venaas-pim-hierarchicaljoinattr-00 (work in progress), February 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

[RFC5384] Boers, A., Wijnands, I., and E. Rosen, "The Protocol Independent Multicast (PIM) Join Attribute Format", RFC 5384, November 2008.

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

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

Authors' Addresses

Jesus Arango
Cisco Systems
170 Tasman Drive
San Jose, CA 95134
USA

Email: jearango@cisco.com

Stig Venaas
Cisco Systems
170 Tasman Drive
San Jose, CA 95134
USA

Email: stig@cisco.com

Isidor Kouvelas
Cisco Systems
170 Tasman Drive
San Jose, CA 95134
USA

Email: kouvelas@cisco.com

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 12, 2013

S. Barkai
ConteXtream Inc.
D. Farinacci
.
D. Meyer
Brocade
F. Maino
V. Ermagan
Cisco Systems
March 11, 2013

LISP Based FlowMapping for Scaling NFV
draft-barkai-lisp-nfv-00

Abstract

This draft describes distributed flow-mapping applied according to RFC 6830 Locator ID Separation Protocol (LISP) for dynamic scaling of virtualized network functions (NFV) . Network functions such as subscriber management-mobility-security-quality, are typically delivered using proprietary appliances topologically embedded into the network as service-nodes or service-blades. Next generation virtualized network functions are pure software instances running on standard servers - unbundled building blocks of processing capacity and modular functionality. LISP based flow-mapping dynamically wires VNF instances into the data-path, and scales virtualized functions by steering the right traffic in the right sequence to the right process.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2013.

Copyright Notice

Table of Contents

1. Introduction

This draft describes distributed flow-mapping applied according to RFC 6830 Locator ID Separation Protocol (LISP) for dynamic scaling of virtualized network functions (NFV) .[RFC6830]Network functions such as subscriber management-mobility-security-quality are typically delivered using proprietary appliances topologically embedded into the network as service-nodes or service-blades.

This monolithic service delivery method increases the complexity of roll-out and capacity planning, limits functional choices, and inhibits service innovation. Next generation virtualized network

functions are pure software instances running on standard servers - unbundled building blocks of compute capacity and modular functionality. This component based model opens up service provider networks to the savings of elasticity and the innovation of an open architectures. However this model also presents the network (or the virtual network rather) with the brand new challenges of assembling components into whole solutions by forwarding the right traffic to the right process at the right sequence and the right time.

While it is possible, to some extent, to use traditional virtual networking mechanisms such as virtual-LANs (VLAN) and virtual-private-networks (VPN) in-order to map traffic to functions-processes, these mechanisms are relatively static and require complex and intense configuration of physical network interfaces with vbridge vrf configuration. NGN software-defined flow based models on the other hand are much more programmable and dynamic, and if orchestrated properly offer a better fit to next generation service-provider data-centers. LISP based flow-mapping enables such a dynamic global orchestration of flows. LISP xTRs wire virtual function instances into the data-path, based on dynamic identity-function and identity-location mappings, perform these actions in a dynamically programmable and elastic manner, and operate based on subscriber-profiles and function-demand global information.

2. Connectivity Models Used

The basic connectivity model used to map flows to function is an identity-matrix forwarding. Unlike topological forwarding models which are based on source-subnet >> routed hop by hop >> to destination-subnet, identity-matrices are based on flow-identity "patched" to function-identity. This model is implemented using the LISP distributed overlay and LISP distributed database mechanisms. These mechanisms are applied over in-place physical networks in a manner described bellow:

- o The topological network basis or the "location" network is assumed to be implemented using standard bridging and routing. Basic design principles are applied in order to achieve both physical capacity and physical availability of connectivity. Typical examples of these practices include spine-leafs switching for cluster many server racks that are used as the compute and virtual compute foundations to functions. These practices also include core-edge routing for inter-connecting server clusters across points of presence, as well as for inter-connecting these points of presence to the access networks and to the public Internet.

- o The functional network or the "identity" matrix is there to map identified subscriber-flows carrying an application thread to the

right function task or instance. This identity-matrix enables scaling and massive concurrency of the logical compute functions. By mapping each subscriber-application flow to the correct processes based on global definitions of the service and application, the system can engage as many functional components as there are available within and across data canters. Applied recursively flow-function matrix mapping can chain as many distinct functional components that make up a service as defined globally by the operator.

o The overlay network or the location-identity fabric enables the implementation of the functional network on the physical in-place bridge-routed network. The overlay forms a virtualization ring around the core-spine physical networks. All subscriber flows and function flows are aggregated at the outer interfaces of this virtualization ring, and are encapsulated over the inner interfaces of this ring in order to reach each of the locations. Ingress-Aggregation, Flow-Seperation, Matrix-Mapping, Encapsulation-Decapsulation, Egress-Delivry are all based on LISP xTRs and LISP mmap architecture and services.

```
                    POP3    POP4
                    \ /     \ /
                   EdgeR -- EdgeRouter
                      |      |
          Access ...  | Core |    ... Internet
                      |      |
                   EdgeR -- EdgeR
                     / \
                    /   \
           ^      Spine1 Spine2 ... Spine5
           |       /  \  /  \    __/ / .. |
           |       |   \/   | __/   /     |
           P       |   /\   ||     /      |
           O      Leaf1   Leaf2  ... Leaf300
           P       |-PC1   |-PC1
           1       |-PC2   |-PC2
           |       |..     |..
           |       |-PC40  |-PC40
           v
```

Topological Location Network

```
     v <<  FunctionA   FunctionB ..  FunctionN
     v
Recursion Instance1..i Instance1..j Instance1..k
     v      | | | |      | | | |      | | | |
     v      | | | |      | | | |      | | | |
Subscriber1 o o o o - - -+ o o o - - -o o o o
            | | | |      | | | |      | | | |
Subscriber2 o + o o - - -o o o o - - -o o o o
            | | | |      | | | |      | | | |
    .         ...          ...          ...
    .         ...          ...          ...
    .         ...          ...          ...
            | | | |      | | | |      | | | |
SubscriberM o o o o - - -o o o o - - -+ o o o
            | | | |      | | | |      | | | |
```

Functional Identity Matrix

```
   AoF AoF AoF Access or Functions AoF AoF AoF
      \ | /          \ | /           \ | /
       BoR            BoR             BoR
        |              |               |
       XTR            XTR             XTR
        ||             ||              ||
         ================================
        ||                              ||
 B     _||                              ||_     B
 o -XTR_ |                              | _XTR- o
 R      ||     Bridges or Routers       ||      R
       _||                              ||_
 B -XTR_ |                              | _XTR- B
 o      ||                              ||      o
 R      ||                              ||      R
         ================================
        ||             ||              ||
       XTR             XTR             XTR
        |              |               |
       BoR             BoR             BoR
      / | \           / | \           / | \
```

Identity-Location Overlay

3. XTR FlowMapping Reference Architecture

In order to map subscriber flows to virtualized function instances and essentially to overlay identity grid onto topology based bridge-routed network we use the following XTR 3-tier reference architecture:

1. Flow-Switching: is a set of n-tuple LOCALLY defined masks, BEST matched against each packet in-order to separate traffic to LOCALLY significant sequences representing application threads. Flows are either Encapsulated in-to the overlay, Decapsulated out-of the overlay, Forwarded up-to xTR internally registered Flow-Handlers ..

2. Flow-Handlers: are a set of control processes invoked for each flow where a specific identity-location mapping has not been defined and provisioned into the Flow-Switching. The default "catch-all" Flow-Handler maps IP flows to locations and gateways based on RFC 6830. In addition protocol-specific handlers can be loaded into the xTR for dealing with specific mapping and AFFINITY requirements of network functions such as SIP, GTP, S1X etc.

3. Global-Mappers: is how GLOBALLY significant key-value mappings is translated by Flow-Handlers to LOCALLY defines masks and encapsulation headers. Examples of such mappings include functional VIP to actual function processes EIDs, application specific SubscriberIDs to function EIDs, public IP-Port to SubscriberID, and EIDs to RLOCs.

```
          Orchestration    Authorization    OSS/BSS
             Mappings        Mappings       Mappings
                 v               v              v
           (NFVMs etc.)     (3A etc.)     (Subs. etc)
                 v               v              v
                 _____________________________________
                |                                    |
                |             LISP-MMAP              |
                |____________________________________|

                    ^             ^              ^
        Runtime Mappings(location, affinity, load, etc.)
                    ^             ^              ^
                    ^             ^              ^
           ^     -------       -------        -------
           |    |MMapper|     |MMapper|      |MMapper|
           |    |-------|     |-------|      |-------|
```

```
X    |F F F F|    |F F F F|     |F F F F|
T    |H H H H|    |H H H H|     |H H H H|
R    |n n n n|    |n n n n|     |n n n n|
|    |d d d d|    |d d d d|     |d d d d|
|    |-------|    |-------|     |-------|
v    | FlowX |    | FlowX |     | FlowX |
      -------      -------       -------
```

Identity-Location Overlay Ring

3.1.

4. Intra-Provider Mappings

5. LCAF Mapping Subscription

6. Inter-Provider Mappings

7. QOS and Echo Measurements

8. Security Considerations

There are no security considerations related with this memo.

9. IANA Considerations

There are no IANA considerations related with this memo.

10. Acknowledgements

11. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

Authors' Addresses

Sharon Barkai
ConteXtream Inc.
California
USA

Email: Sharon@Contextream.c

Dino Farinacci
.
California
USA

Email: farinacci@gmail.com

David Meyer
Brocade
California
USA

Email: dmm@1-4-5.net

Fabio Maino
Cisco Systems
California
USA

Email: fmaino@cisco.com

Vina Ermagan
Cisco Systems
California
USA

Email: vermagan@cisco.com

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

L. Cheng
M. Sun
ZTE Corporation

July 15, 2013

draft-cheng-lisp-nat-traversal-extension-01
Extension to LISP NAT Traversal Proposal

Abstract

This draft specifies several special NAT traversal scenarios when two or more LISP Sites/MNs which locate behind the same NAT equipment communicate with each other. When these LISP Sites/MNs communicate with each other, it may cause routing latency and will increase re-encapsulation load on Re-encapsulating Tunnel Routers(RTRs) based on existing LISP-NAT strategy.

In this draft, we give detail descriptions of these scenarios. Also we propose some suggestions to solve the problems. According to our strategy, a new kind of message is used for RTRs to send relative information of Corresponding Sites/MNs to xTRs.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Locator/ID Separation Protocol [RFC6830] defines a set of functions for encapsulating routers to exchange information used to map from Endpoint Identifiers (EIDs) to routable Routing Locators (RLOCs). When a LISP site locates behind a NAT, LISP Tunnel Routers (xTRs) are only reachable through the NAT's public address which broke the assumption that xTRs are reachable at their RLOCs.

To make sure LISP devices locate behind a NAT reachable, [LISP-NAT] proposes a NAT traversal mechanism for LISP. To achieve this, an ETR of a LISP site will use its Map-Server to discover whether it is behind a NAT and to get its translated global RLOC and port via two LISP messages: Info-Request and Info-Reply. Once an ETR detects it locates behind a NAT, it use a LISP Re-encapsulating Tunnel Router (RTR) to act as a data plane 'anchor point' to send and receive traffic through the NAT device.

According to RTR proxy strategy introduced in [LISP-NAT], when an ITR behind a NAT needs to encapsulate outbound LISP traffic, it does not send Map-Request for destination EIDs, but just use RTR RLOC as locator for all destination EIDs that it wishes to send data to.

However, in some special scenarios such like two or more LISP Sites/ MNs locate behind the same NAT, when these LISP Sites/MNs communicate with each other, this RTR proxy strategy will cause routing latency and extra decapsulation/re-encapsulation cost on RTR. In this draft, we list and describe several special cases. A new kind of message "Info-Notify" is adopted. This message is used by RTR to notify the LISP Site with relative information of its Corresponding Site which locates behind the same NAT with it. Based on this "Info-Notify" message, some feasible solutions are proposed.

2. Definition of Terms

This draft uses terms defined in [RFC6830] and [LISP-NAT]. This section introduces some new terms used in the document.

Corresponding Site/MN When two LISP Sites/MNs communicate with each other, each LISP Sites/MN is considered as Corresponding Site/MN of the other one, regardless of Site’s/MN’s status, i.e. it is a sender or a receiver.

Info-Notify A message used by RTR to notify LISP Site with relative information of its Corresponding Site.

3. Issue Statement

In this section, several scenarios are specified.

3.1. Basic Scenario-Single Level NAT

As shown in Fig.1, there are two LISP Sites (Site1 and Site2) locate behind the same Nat equipment (NAT1). Furthermore, xTRs of the two sites choose the same RTR as proxy to register mapping information and transfer data traffic.

```
                                  +-------+
                                  |  RTR  |
                                  +---+---+
                                      |
                                  +---+---+
                                  |  NAT1 |
                                  +-++-++-+
                                   / | | \
   +----------------------------+ /  | |  \ +----------------------------+
   |  +------+_____+------+_____|/   | |   \|_____+------+   __+------+  |
   |  |Host 1|     | ITR1 |     |    | |    |     | ITR2 |  /  |Host 2|  |
   |  +------+     +------+     |    | |    |     +------+ /   +------+  |
   |               +------+     |    | |    |     +------+/              |
   |     Site1     | ETR1 |_____|____| |____|_____| ETR2 |     Site2     |
```

```
|              +------+     |           |     +------+              |
+-----------------------------+           +-----------------------------+
```

Fig.1 Case 1: Single Level Nat Traversal

Based on RTR proxy strategy proposed in [LISP-NAT], when Host1 in Site1 want to send a packet to Host2 in Site2, following steps will be performed.

1. Host1 sends a packet to ITR1 of Site1, destination of the packet is EID address of Host2.

2. When ITR1 receives the packet, it encapsulates it in a LISP data header with outer header destination set to RTR RLOC and outer header destination port set to 4341. Based on RTR proxy strategy, ITR1 will not send Map-Request for Host2. As a result, ITR1 is not able to get RLOCs of Site2, i.e. ETR2’s RLOC.

3. When encapsulated packet passes through NAT, NAT transform the source address and source port in outer header to be global address and port. This may create a state in the NAT device.

4. When RTR receive the encapsulated packet destination to its RLOC, it decapsulates the packet, and look for if there are local mapping information match the destination EID. As Site2 also registers through the RTR, RTR will find mapping information of Site2, as well as global state of ETR2 including the global address and global port in its local cache.

5. RTR uses ETR2’s global state information to re-encapsulate the packet, and transfer it to ETR2.

Seen from steps enumerated above, based on existing RTR Proxy strategy, even though Site1 and Site2 locate behind the same NAT, traffic between these Sites need to route to the RTR which locates outside the NAT.

However, as Site1 and Site2 locate behind the same NAT, that’s to say, ITR1 and ETR2 locate in the same Intranet, RLOC of ETR2 is reachable to ITR1. As a result, if ITR1 could get mapping information of Site2, it could encapsulate the packet directly to RLOC of ETR2 which could avoid routing latency of packet and could lighten re-encapsulation load on RTR.

3.2. Extended Scenario I-Multiple Levels NAT

In some topologies, there may include multiple NAT devices. Consider the scenario depicted in Fig.2. Suppose NAT1 is a large industrial NAT deployed by an internet service provider (ISP) to multiplex many customers onto a few public IP addresses, and NAT2 is a small consumer NAT router deployed by one of the ISP's customers to multiplex its private home networks onto its ISP-provided IP addresses. Only RTR and NAT1 have globally routable IP addresses; Site1's RLOC addresses and the "public" IP addresses used by NAT2 are actually private to the ISP's address realm, while Site2's addresses are private to the addressing realms of NAT2.

```
                                +-------+
                                |  RTR  |
                                +---+---+
                                    |
                                +---+---+
                                |  NAT1 |
                                +++---+-+
                                 ||    \
                                 || +------+
                                 || | NAT2 |
                                 || +-+--+-+
+-------------------------+   ||   |  |  +-------------------------+
|  +------+_____+------+__|___||   |  |__|__+------+   __+------+  |
|  |Host 1|     | ITR1 |  |    |   |     |  | ITR2 |  /  |Host 2|  |
|  +------+     +------+  |    |   |     |  +------+ /   +------+  |
|               +------+  |    |   |     |  +------+/              |
|     Site1     | ETR1 |__|____|   |_____|__| ETR2 |     Site2     |
|               +------+  |              |  +------+               |
+-------------------------+              +-------------------------+
```

When Site1 register through RTR, it follows the same steps described in [LISP-NAT]. NAT1 establishes states for sessions between RTR and xTR1s. RTR stores relative information of Site1 in its cache, including mapping information, global IP addresses and global ports of xTR1s, etc..

According to Site2, when Site2 register through RTR, both NAT1 and NAT2 will establish states for sessions between RTR and xTRs. When Site2 sends messages no matter control messages or data packets to RTR, both NAT2 and NAT1 translates IP address and port of packet's out header.

According to RTR, when RTR receives an encapsulated packet from Site2, out header address of packet will be a routable address assigned by NAT1, and inner header address will be EID of source host. As a result, addresses which assigned by NAT2 and used for

routing between NAT1 and NAT2 are invisible to RTR. In RTR's local cache, it will store relative information of Site2, including mapping information, global IP addresses and global ports of xTR2s which are assigned by NAT1, etc.

In this scenario, every packet from Host1 to Host2 will follow the path:

Host1 -> ITR1 -> NAT1 -> RTR -> NAT2 -> ETR2 ->Host2

RTR will perform decapsulation and re-encapsulation process.

Based on existing RTR Proxy strategy, traffic between Site1 and Site2 still needs to route to the RTR which locates outside the NAT.

According to scenario depicted in Fig.2, even if ITR1 could get mapping information of Site2, ETR2's RLOC is not reachable to ITR1.

However, if ITR2 gets global information of Site2, i.e. global addresses and global ports of xTR2s which are used to route outside NAT1, it could encapsulate packets directly to ETR2's global address. If NAT1 supports Hairpin function, the packets destination to ETR2's global address will be route to Site2, and need not to be re-encapsulated by RTR.

3.3. Extended Scenario II-Multiple Levels NAT

As shown in Fig.3, Site1 and Site2 both locate behind two levels NAT. Site1 and Site2 register through RTR as we described in Section 3.2.

In this scenario, each packet from Host1 to Host2 follows the path:

Host1 -> ITR1 -> NAT2 -> NAT1 -> RTR -> NAT3 -> ETR2 ->Host2

In this situation, if Site1 gets global information of the Corresponding Site, ITR1 could probe ETR2's global IP address. If NAT1 support Hairpin function, probe message will be transfer to Site2, and Site1 need not to encapsulate packet to RTR anymore.

```
                          +-------+
                          |  RTR  |
                          +---+---+
                              |
                          +---+---+
                          |  NAT1 |
                          +-+---+-+
                           /     \
                     +------+  +------+
```

```
                            | NAT2 |  | NAT3 |
                            +-+--+-+  +-+--+-+
   +-------------------------+ |  |      |  | +-------------------------+
   |  +------+_____+------+__|_|  |      |  |_|__+------+   __+------+  |
   |  |Host 1|     | ITR1 |  |    |      |    |  | ITR2 |  /  |Host 2|  |
   |  +------+     +------+  |    |      |    |  +------+ /   +------+  |
   |               +------+  |    |      |    |  +------+/              |
   |     Site1     | ETR1 |__|____|      |____|__| ETR2 |     Site2     |
   |               +------+  |               |  +------+               |
   +-------------------------+               +-------------------------+
```

4. Solutions

According analysis we described above, in these section we propose our solution strategy. In this

This mechanism enables RTR to send an "Info-Notify" message to the LISP Site with relative information of its Corresponding Site.

4.1. RTR Processing

When RTR receives an encapsulated packet from ITR1 of Site1, following steps will be performed.

1. RTR strips the outer header and lookup in local cache for mapping information of destination EID.

2. In our extension, RTR will judge if Source Site and Destination Site locate behind the same NAT. For example, based on whether the global address of Source Site ITR1 and global address of Destination Site ETR2 are the same, or whether the two global addresses may locate in the same address prefix. Furthermore, if ISP would like to build a relationship between NAT and corresponding RTR, RTR could be informed with public address information of the NAT.

3. If RTR judge that source site and destination site both locate behind the same NAT, it could send "Info-Notify" messages which contain relative information of Corresponding Site to Sender ITR and Receiver ETR respectively. Relative information mentioned above may include mapping information of Site, global addresses and global ports information of Site xTRs.

Note: After RTR sending sage to ITR, if RTR receive packet from ITR to the particular ETR, RTR continue to transfer the packet to destination ETR.

4.2. ITR Processing

When ITR receives "Info-Notify" message from RTR,

1. Based on mapping information of Destination Site, ITR could send a RLOC probe message to ETR’s RLOC. After receiving Mapping Reply from ETR, ITR could encapsulate packet directly to ETR’s RLOC.

2. If Destination Site locate behind multilevel NATs, such like scenarios described in Fig.2 and Fig.3, ETR’s RLOC is not reachable to ITR, and ITR will not receive Map Reply form Destination Site when it send RLOC Probe message to ETR’s RLOC. In this situation, ITR could then send Probe message to ETR’s global address. If ITR could get map reply message, then it could use ETR’s global address as destination address of outer header to encapsulate packet.

Note In order to avoid traffic affection, during RLOC Probe phase, ITR could continue to encapsulate data packet to RTR.

4.3. ETR Processing

When ETR receives "Info-Notify" message from RTR, ETR could choose to send a message such like a SMR (Solicit-Map-Request)[RFC6830] to ITR to trigger a mapping request operation. Furthermore, ETR could use ITR’s RLOC or global address as destination when it send the message.

5. Security Considerations

TBD

6. IANA Considerations

This document makes no requests to IANA.

7. Normative References

[LISP-NAT] Ermagan, V., Farinacci, D., Lewis, D., Skriver, J., Maino, F., and C. White, "NAT traversal for LISP", March 2013.

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

Authors’ Addresses

Li Cheng
ZTE Corporation
R&D Building 1, Zijinghua Road No.68
Nanjing, Yuhuatai District 210012
P.R.China

Email: cheng.li2@zte.com.cn

Mo Sun
ZTE Corporation
R&D Building 1, Zijinghua Road No.68
Nanjing, Yuhuatai District 210012
P.R.China

Email: sun.mo@zte.com.cn

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

L. Cheng
M. Sun
ZTE Corporation

July 15, 2013

draft-cheng-lisp-shdht-04
LISP Single-Hop DHT Mapping Overlay

Abstract

This draft specifies the LISP Single-Hop Distributed Hash Table Mapping Database (LISP-SHDHT), a distributed mapping database which consists of a set of SHDHT Nodes to provide mappings from LISP Endpoint Identifiers (EIDs) to Routing Locators (RLOCs). EID namespace is dynamically distributed among SHDHT Nodes based on DHT Hash algorithm. Each SHDHT Node is configured with one or more hash spaces which contain multiple EID-prefixes along with RLOCs of corresponding Map Servers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Locator/ID Separation Protocol (LISP) [RFC6830] specifies an architecture and mechanism for replacing the address currently used by IP with two separate name spaces: Endpoint IDs (EIDs), used within LISP sites, and Routing Locators (RLOCs), used on transit networks that make up the Internet infrastructure. To achieve this separation, LISP defines protocol mechanisms for mapping from EIDs to RLOCs. As a result, an efficient database is needed to store and propagate those mappings globally. Several such mapping databases have been proposed, among them: LISP-NERD [I-D.lear-lisp-nerd], LISP-ALT[RFC6836], LISP-DDT[I-ietf-lisp-ddt], and LISP-DHT [LISP-DHT].

According to databases such like LISP-ALT [RFC6836] and LISP-DDT [I-ietf-lisp-ddt], architectures of these mapping databases are based on announcement/delegation of hierarchically-delegated segments of EID namespace (i.e., prefixes). Therefore, based on these architectures, when a roaming event occurs and a LISP site or a LISP MN receives new RLOCs, the site or MN has to anchor pre-configured map-server to register its new mapping information no matter where the site or MN currently locates, just in order to protect EID prefixes announced aggregately in the database [I-D.meyer-lisp-mn].

As a DHT strategy based mapping database, LISP-DHT [LISP-DHT] exhibits several interesting properties, such as self-configuration, self-maintenance, scalability and robustness that are clearly desirable for a EID-to-RLOC resolution service. However, this database is based on multi-hop Chord DHT. On one hand, inquiries of mapping information in this case need to pass through iterative multi-hop lookup steps which will cause relatively large delay time. On the other hand, load balance between Chord nodes is another essential problem need to be solved.

This draft specifies a LISP Single-Hop Distributed Hash Table Mapping Overlay (LISP-SHDHT) which provides mapping information lookup service for sites running LISP. Main characters of this strategy is that,

1. Each SHDHT Node maintains routing information for all other SHDHT Nodes. Thus, messages interaction between SHDHT Nodes in the same SHDHT overlay just need one hop.

2. Traditionally, Node IDs are used to identify DHT nodes and represent hash space arrangement on DHT nodes. In SHDHT strategy, the two roles are separated. Partition IDs are adopted for hash space arrangement and a build-in load balancing solution is designed.

This draft specifies the outline of SHDHT and the basic application of LISP SHDHT. In actual deployment of LISP SHDHT, mapping database could be maintained by multiple service providers and could be deployed as collaborative combination of multiple Domain LISP SHDHTs. Details about Domain LISP SHDHT Deployment are introduced in Section 5.

In main context of this draft, SHDHT Mapping database is proposed according to structure requirements of LISP-MS [RFC6833]. This SHDHT strategy provides services for LISP Sites mapping lookup. In Appendix A, a special SHDHT strategy for LISP-MN [I-D.meyer-lisp-mn] scenario is proposed. This SHDHT strategy is not completely match structure requirements of LISP-MS. However, it could provide better

service for LISP-MN scenario, where LISP-MN need not to anchor pre-configured Map Server and could update its mapping information as soon as possible when it roams to a new location.

2. Definition of Terms

This draft uses terms defined in [RFC6830]. This section defines some new terms used in this document.

SHDHT: Single-Hop Distributed Hash Table Mapping Overlay.

SHDHT Node: Physical nodes which compose SHDHT overlay's topology. Each SHDHT Node has a unique Node ID and maintains multiple hash space segments which labeled by Partition IDs. Each SHDHT Node maintains a Node Routing Table of local SHDHT Mapping Overlay. SHDHT Nodes locates in the same Mapping Overlay implement hash operation based on the same hash algorithm. SHDHT Nodes hash data object to be a unique Resource ID, and perform put/get/move operations based on the Resource IDs.

Node ID: Node identifier, which is used for maintenance. Each SHDHT Node has a unique Node ID. The ring containing Node IDs indicates overlay's topology.

Partition ID: Partition identifier, which is used for hash space assignment. Partition IDs and Resource IDs share the same hash space. All Partition IDs in overlay are unique. Each SHDHT Node could have multiple Partition IDs. The ring containing Partition IDs determines how the hash space is partitioned into segments and how these segments are assigned to nodes.

Resource ID: Each data object stored in DHT overlay could be hashed to be a unique Resource ID. In LISP-SHDHT strategy, data objects correspond to the EIDs. Resource IDs share the same hash space with Partition IDs. As a result, SHDHT Nodes perform data objects put/get/remove operations based on these IDs.

Node Routing Table: Routing table of a SHDHT Mapping Overlay which contains all SHDHT Nodes information of this overly, including Node IDs, Partition IDs and Node IP addresses, etc. Each SHDHT Node of this overly will maintain the Routing Table.

SHDHT Map Resolver: A SHDHT Node that also implements Map Resolver functionality (accept Map-Requests, and forward them to corresponding SHDHT Map Servers based on information get from SHDHT mapping database or forward them to corresponding SHDHT Map Servers through SHDHT Nodes, which corresponds to different operation modes of the SHDHT Mapping Database.).

Report Message: Kind of message sent by Map Server to SHDHT Nodes to publish the EIDs/EID prefix information in charge of Map Server onto the LISP-SHDHT mapping overlay.

SHDHT Border Node: SHDHT Border Node locates on the border of a Domain SHDHT Overlay. Each SHDHT Border Node maintains an Inter-Domain Routing Table. SHDHT Border Nodes are used to flood cross domain routing and forward cross domain packets.

Inter-Domain Routing Table: A routing table maintained on a SHDHT Border Node. This routing table contains information of other Domain SHDHT Overlays, such like EID prefixes other domain overlays maintain, IP addresses and ports information of other overlay’s Border Nodes.

3. SHDHT Overview

3.1. Node ID and Partition ID

Most of existing DHTs use node IDs for both maintenance and hash space arrangement. For example, in LISP-DHT[LISP-DHT], each chord node of the DHT ring has a unique k-bits identifier (ChordID). Nodes perform operations such like put/get/remove based on ChordIDs. Furthermore, ChordIDs are also used to associate nodes with hash space segments that the nodes responsible for.

In SHDHT, two roles of maintenance and hash space arrangement are separated and a new kind identifier called Partition ID is adopted. Each SHDHT node has a unique Node ID which identifies the physical node and multiple Partition IDs which represent hash space segments. All Partition IDs in the overlay are also unique. Node IDs and Partition IDs are mapped into two ring-shaped spaces respectively. The ring containing Node IDs indicates the overlay’s topology. The ring containing Partition IDs determines how the hash space is partitioned into segments and how these segments are assigned to nodes. It is noteworthy that SHDHT Nodes could determine number of Partition IDs on them separately and could generate Partition IDs randomly just need to make sure that the generated Partition IDs will not conflict with existing Partition IDs on the SHDHT plane.

```
+--------------------+                        +--------------------+
|Node ID:      0x0123|                        |Node ID:      0x4444|
|Partition ID: 0x1234| +-----+         +-----+ |Partition ID: 0x9000|
|              0x7000| |Node1+----------+Node2| |              0x3234|
+--------------------+ +--+--+         +--+--+ +--------------------+
                          |               |
                          |               |
                          |               |
                          |               |
+--------------------+ +--+--+         +--+--+ +--------------------+
|Node ID:      0xe000| |Node3+----------+Node4| |Node ID:      0xc000|
|Partition ID: 0x5000| +-----+         +-----+ |Partition ID: 0xaaaa|
|              0xeeee|                         |              0xcccc|
+--------------------+                        +--------------------+
```

Fig.1 SHDHT Deployment Example

As shown in Fig.1 is an example of SHDHT. This SHDHT overlay is consist of four SHDHT NODEs each has a unique Node ID and maintains two Partition IDs. According to this deployment, hash space is partitioned to be eight segments each is indexed by a Partition ID. From Fig. 1, it could be observed that hash space segments are not required to be partitioned equally. As SHDHT Nodes could generate

Partition IDs separately, when a SHDHT Node gets all hash segments assignment information of other SHDHT Nodes, it will be able to implement the load balance of SHDHT overlay by generate proper Partition IDs.

In SHDHT, each SHDHT Node stores and maintains data objects. Data objects are indexed by Resource IDs which share the same hash space with Partition IDs and will locate in the hash space segments whose Partition IDs are closest to their Resource IDs.

For example, for a data object whose Resource ID is 0x8213, the Resource ID locates between Partition ID 0x7000 and Partition ID 0x9000. As Partition ID 0x9000 is closer to Resource ID 0x8213, the data object will be stored and maintained on Node2 who is assigned with the hash space segment indexed by Partition ID 0x9000.

3.2. Data Storage and Hash Assignment

In traditional DHTs, hash space is partitioned into segments based on node IDs. As a result, data objects are always stored in their root nodes, whose node IDs are "closest" to data objects' Resource IDs.

What does "closest" means? Suppose we have three consecutive Partition IDs a, b and c which are the only Partition IDs in SHDHT for our example, then the range of each hash space segment is defined as follow:

Partition ID a: [id(a)-0.5*d(c,a); id(a)+0.5*d(a,b))

Partition ID b: [id(b)-0.5*d(a,b); id(b)+0.5*d(b,c))

Partition ID c: [id(c)-0.5*d(b,c); id(c)+0.5*d(c,a))

with functions

id(x): value of Partition ID x in hash space

d(x,y): distance between Partition ID x and y in hash space

Replications of data objects in a particular node are always stored in the preceding node or successor node of the root node. The backup preceding node or successor node will automatically become the new closest node if the root node leaves the overlay.

In SHDHT, the whole hash space is partitioned into segments based on partition IDs. The root node of a data object is the node, which has the closest partition ID to the data object's Resource ID. In SHDHT, each node can maintain multiple hash space segments with respective

Partition IDs. As the preceding Partition ID or successor Partition ID may be owned by the same root node. Replication of data objects could still be stored in preceding node or successor node of root node.

3.3. Node Routing Table

In SHDHT, each node maintains a Node Routing Table containing routing information of all other SHDHT Nodes locate in the same SHDHT overlay. Table I shows the Node Routing Table on SHDHT Nodes of Fig.1. A Node Routing Table contains all Partition IDs and their associated Node IDs and node addresses. For simplification, Node IDs and Partition IDs shown in the draft are only 16-bit numbers.

When SHDHT Node receives a message points to a particular Resource ID, it could look up Node Routing Table and find out the Partition ID which is closest to the Resource ID. Furthermore, message could be transferred to the corresponding SHDHT Node.

Partition ID	Node ID	Address:Port
0x1234	0x0123	10.0.0.2:2000
0x3234	0x4444	10.0.0.3:2000
0x5000	0xe000	10.0.0.4:2000
0x7000	0x0123	10.0.0.2:2000
0x9000	0x4444	10.0.0.3:2000
0xaaaa	0xc000	10.0.0.5:2000
0xcccc	0xc000	10.0.0.5:2000
0xeeee	0xe000	10.0.0.4:2000

TABLE I SHDHT Node Routing Table

For example, if Node 1 (ID: 0x1234) in Fig.1 needs to implement put/get/remove operations for a data object with Resource ID 0x8213. Node 1 first looks up its Node Routing Table and finds out that the closest Partition ID corresponding to this Resource ID is 0x9000. Then Node 1 will send put/get/remove request to the node owns the Partition ID, in Fig.1 is Node2, whose Node ID is 0x4444 and address is 10.0.0.3:2000.

4. LISP SHDHT

LISP SHDHT is proposed to provide "EID-to-RLOC(s)" mapping information lookup service for sites running the Locator/ID Separation Protocol (LISP).

As shown in Fig.2, LISP SHDHT is consists of SHDHT Nodes in which some nodes play roles of Map Resolver. And in the draft, term "SHDHT-MR" is adopted to identify these nodes.

```
                                   +--------------------+
                                   |Node ID:      0x4444|
             +-----+         +-----+ |Partition ID: 0x9000|
             |Node1+---------+Node2| |              0x3234|
             +--+--+         +--+--+ +--------------------+
                |               |
                |     SHDHT     |
                |               |
             +--+--+         +--+--+       +-----+
      +-------+ M R +---------+Node4+-------+ M S |
      |      +-----+         +-----+       +--+--+
      |                                       |
   +--+--+                                  +--+--+
   | ITR |                                  | ETR |
   +-----+                                  +-----+
```

Fig.2 LISP-SHDHT Deployment Example

Map Server publishes EIDs/EID prefixes information it responsible to onto SHDHT Mapping overlay. All EIDs/EID prefixes entries are stored in SHDHT Nodes as data objects. EIDs/EID prefixes in mapping entries can be hashed as Resource IDs of data objects. All SHDHT Nodes in the same SHDHT overlay perform hash operation based on the same hash algorithm.

In this draft, LISP-SHDHT can run in two distinct modes: i) SHDHT Forward Mode and ii) Recursive Lookup Mode.

4.1. ITR Operation

According to LISP-MS [RFC6833], LISP ITRs use Map Resolvers as proxy to send control messages, such like encapsulated Map-Requests and Map-Replies.

In Scenario of LISP SHDHT, an ITR send Map-Requests directly to the SHDHT Node which is selected to play roles of SHDHT Map Resolver for that ITR.

4.2. ETR Operation

LISP ETR plays the same role as defined in LISP-MS [RFC6833]; it registers mapping information onto the Map Server by sending Map-Register messages.

4.3. SHDHT Map Server Operation

When Map Server receives Map Request messages, it forwards the messages to ETRs or send Map-Reply messages directly based on M-bits of ETRs' Map Registers. That's to say, Map Server performs the same operation as defined in LISP-MS[RFC6833] .

Extended in LISP SHDHT, Map Server needs to publish EIDs/EID prefixes information onto the LISP-SHDHT mapping overlay.

For example, as shown in figure 2, when a Map Server needs to publish EIDs information such like EID "1.1.1.1" onto LISP-SHDHT, following operations will be performed.

1. Map Server sends a report message to the nearest SHDHT Node, i.e. Node 4. Map Server contains the information of EID "1.1.1.1" in the report message, along with Map Server's routable RLOCs. Report message may not be a kind of new defined message. For example, Map Server could advertise EID information onto SHDHT mapping database based on BGP protocol.

2. When Node 4 receives the report message, it extracts EID information from the message, i.e. EID "1.1.1.1". Node 4 then hashes the report EID to be a Resource ID.

3. Suppose Node 4 hash EID "1.1.1.1" to be Resource ID 0x8956. Then it checks the Nodes Routing Table to find out which Node maintains the hash space whose Partition ID matches the Resource ID. In this example, the match Partition ID is 0x9000, and the corresponding hash space is maintained by Node 2.

4. Node 4 forwards the report message to Node 2. Node 2 stores the (key, value) pair, where key is the Resource ID 0x8956, and value contains reported EID "1.1.1.1" along with corresponding Map Server's RLOCs.

Other SHDHT Nodes now could contact Node2 to get which Map Server is responsible to the EID "1.1.1.1".

Note: In previous example, we suppose that Map-Server reports an EID address onto LISP-SHDHT. In practical deployment, Map Server reports EID prefixes at most time. Details about EID prefix report will be illustrated in Section 4.6.

4.4. SHDHT Map Resolver Operation

As previous mentioned, LISP-SHDHT can run in two distinct modes: i) SHDHT Forward Mode and ii) Recursive Lookup Mode.

As shown in Fig.2, suppose SHDHT Map Resolver receives a Map-Request message target at EID 1.1.1.1. SHDHT Map Resolver operations under two different modes are illustrated in following sections.

4.4.1. SHDHT Map Resolver Operation under SHDHT Forward Mode

Under SHDHT Forward Mode, SHDHT Map Resolver performs the following operaion.

1. SHDHT Map Resolver extracts destination EID address "1.1.1.1" from the Map-Request message.

2. SHDHT Map Resolver hashes the EID address to be Resource ID 0x8956 based on the shared hash algorithm.

3. SHDHT Map Resolver looks up Node Routing Table and finds out the Partition ID 0x9000 which matches the Resource ID.

4. SHDHT Map Resolver forwards Map-Request message to the corresponding SHDHT Node 2 who maintains the hash space labeled by matched Partition ID 0x9000.

4.4.2. SHDHT Map Resolver Operation under Recursive Lookup Mode

Under Recursive Lookup Mode, SHDHT Map Resolver performs the following operation.

1. SHDHT Map Resolver receives the Map-Request message and stores it in local catch.

2. SHDHT Map Resolver hash destination EID of Map-Request to be Resource ID 0x8956.

3. SHDHT Map Resolver looks up Node Routing Table and finds out that Node 2 maintains the corresponding hash space and stores the data object indexed by 0x8956.

4. SHDHT Map Resolver query Node 2 to get data object indexed by 0x8956, i.e. get EID information and RLOCs of the Map Server who maintains the destination EID.

5. SHDHT Map Resolver forwards Map-Request message to corresponding Map Server based on data object.

Under Recursive Lookup Mode, SHDHT Map Resolve could catch information of the Map Server, including EID prefix Map Server responsible for and Map Server’s RLOCs. When receives other Map Requests whose destination EIDs covered by Map Server’s EID prefix, Map Resolver could forward Map Requests directly to Map Server.

4.5. SHDHT Nodes Operation

As specified in Section 4.3, when SHDHT Nodes receive a report message, it will hash the report EID/EID prefix to be Resource ID, and check which Node should store the data object. If itself is the responsible Node, it will store the (key, value) pair, otherwise, it forward report message to corresponding Node.

As specified in Section 4.4.1, under SHDHT Forward Mode, when a SHDHT Node receives a Map-Request message from a Map Resolver, it hashes the requested EID to be a Resource ID to get the data object stored in its hash space. Then SHDHT Node forwards the Map-Request to Map Server based on data object information.

As specified in Section 4.4.2, under Recursive Lookup Mode, when a SHDHT Node receives a query message from Map Resolver, it replies with the data object Map Resolver requested.

4.6. EID prefixes Report onto LISP-SHDHT

In LISP-SHDHT, Map Server always report EID prefixes onto the SHDHT Mapping overlay. However, Map-Request message always targets at a specific EID address. How to hash the requested EID and the EID prefix covered this EID to be the same Resource ID? Each LISP-SHDHT Mapping overlay could configure a "Hash Bit" to solve this problem.

As shown in Fig.3, suppose the LISP-SHDHT Mapping overlay configures the "Hash Bit" to be 16 bits.

```
   +--------------------+                        +--------------------+
   |Node ID:      0x0123|                        |Node ID:      0x4444|
   |Partition ID: 0x1234| +-----+         +-----+ |Partition ID: 0x9000|
   |              0x7000| |Node1+---------+Node2| |              0x3234|
   +--------------------+ +--+--+         +--+--+ +--------------------+
                             |               |
                             |     SHDHT     |
                             |  Hash Bit=16  |
                             |               |       1.1.1/24
                          +--+--+         +--+--+    +-----+    +------+
     Map-Request  +-------+ M R +---------+Node4+----+ MS1 +----+ ETR1 |
       1.1.1.1    |       +-----+         +--+--+    +-----+    +------+
       2.0.0.1    |                          |
                +--+--+                       |       +-----+    +------+
                | ITR |                       +-------+ MS2 +----+ ETR2 |
                +-----+                               +-----+    +------+
                                                      2.0/15
```

Fig.3 EID Prefix Report Example

Example 1: MS1 reports EID prefix 1.1.1/24 onto the LISP-SHDHT.

1. When Node4 receives the report message from MS1, it extracts the EID prefix 1.1.1/24.

2. Node4 hashes the EID prefix to be Resource ID based on Hash Bit. In this case, Hash Bit is 16 bits, as a result Node4 hashes the /16 prefix of reported EID prefix. That’s to say, Node4 hashes 1.1/16 to be a Resource ID, suppose to be 0x8560.

3. Node 4 checks Node Routing Table and forwards the report message to Node 2 who maintains the corresponding hash space with Partition ID 0x9000.

In this example, when another Map Server advertises EID prefix such like 1.1.2/24, this prefix will also be hashed to be Resource ID 0x8560 and the report message will be forwarded to Node 2.

Node 2 maintains (key, value) pair, where key is 0x8560 and value contains all EIDs/EID prefixes information covered by 1.1/16.

Example 2: MS2 reports EID prefix 2.0/15 onto the LISP-SHDHT.

1. When Node4 receives the report message from MS1, it extracts the EID prefix 2.0/15.

2. Node4 hashes the EID prefix to be Resource ID based on Hash Bit. In this case, Hash Bit is 16 bits, Node4 first splits the EID

prefix 2.0/15 to be two 16 bits sub-prefixes, 2.0/16 and 2.1/16. Suppose Node 4 hashes prefix 2.0/16 to be Resource ID 0x1210 and hashes prefix 2.1/16 to be Resource ID 0x3200.

3. As Shown in Fig. 3, data objects with Resource ID 0x1210 and 0x3200 should be stored on Node 1 and Node 2 separately. Node 4 will copy the report message and forward report message both to Node 1 and Node 2.

In this example, Node 1 and Node 2 maintains (key, value) pairs with different keys (0x1210 and 0x3200), but the value both contain the same EID prefix 2.0/15.

In practical deployment, SHDHT service providers could configure proper Hash Bits, in order to avoid the scenario which needs to split a shorter EID prefix to be multiple longer prefixes.

Example 3: ITR sends Map Requests onto LISP-SHDHT.

1. When ITR sends a Map-Request target at EID 1.1.1.1 as shown in Fig.3, SHDHT Map Resolver hashes the EID based on Hash Bit, i.e. SHDHT Map Resolver hashes EID prefix 1.1/16 to get the Resource ID. SHDHT Map Resolver judges the corresponding data object is stored on Node 2 (according to Example 1), then SHDHT Map Resolver could forward the Map-Request to Node 2 (based on SHDHT Forward Mode) or get information about the best matched EID prefix 1.1.1/24 from Node 2 (based on Recursive Lookup Mode).

2. When ITR sends a Map-Request target at EID 2.0.0.1, SHDHT Map Resolver hashes EID prefix 2.0/16 to get the Resource ID. SHDHT Map Resolver judges the corresponding data object is stored on Node 1 (according to Example 2), then SHDHT Map Resolver could forward the Map-Request to Node 1 (based on SHDHT Forward Mode) or get information about the best matched EID prefix 2.0/15 from Node 1 (based on Recursive Lookup Mode).

5. Domain LISP SHDHT Deployment

LISP is a global architecture. In order to make LISP SHDHT meets requirements of LISP mapping database better, LISP SHDHT should perform better scalability and distribution attributes. Especially in practical deployment, LISP mapping database may be operated by different ISPs, when a new mapping service provider join or leave the mapping database, all other providers should not be influenced to be re-assigned.

In practical deployment, LISP SHDHT mapping overlay could be consist of multiple Domain SHDHT overlays which are operated by different mapping service providers. These Domain SHDHT overlays communicate through SHDHT Border Nodes of each other.

As shown in Fig.4, there are two Domain LISP SHDHT Overlays which communicate through BN1 (Border Node1) and BN2.

In domain LISP SHDHT deployment, different domain overlays maintain EID-to-RLOC mapping information covered by different EID prefixes. As in example of Fig.4, Domain 1 maintains mapping information according to EID prefix 12.0.0.0/8, and Domain 2 maintains mapping information covered by EID prefix 16.0.0.0/8. Furthermore, different Domain Overlay could configure their Hash Bits separately.

```
 +------+  +-----+       +-----+     +-----+       +-----+   +------+
 | ITR1 +--+ MR1 +-------+Node2|     |Node4+-------+ MR2 +---+ ITR2 |
 +------+  +--+--+       +--+--+     +--+--+       +--+--+   +------+
             |   Domain    |           |   Domain    |
             |  Overlay 1  |           |  Overlay 2  |
             | Hash Bit 16 |           | Hash Bit 14 |
             | 12.0.0.0\8  |           | 16.0.0.0\8  |
  +-----+  +--+--+       +--+--+     +--+--+       +--+--+   +-----+
  | MS1 +--+Node3+-------+ BN1 |<--->| BN2 +-------+Node6+---+ MS2 |
  +-----+  +-----+       +-----+     +-----+       +-----+   +-----+
```

* BN: SHDHT Border Node
* MR: SHDHT Map Resolver
* MS: Map Server

Fig.4 Domain SHDHT Deployment Example

5.1. SHDHT Border Node

Each Domain SHDHT Overlay has one or more Border Nodes which are not only perform like normal SHDHT Nodes, but also be used to flood cross domain routing and forward the cross domain packets.

Each SHDHT Border Node maintains an Inter-Domain Routing Table, which contains information of all other domain overlays, such like EID prefixes other domain overlays maintain, IP addresses and ports information of other overlays' Border Nodes.

At the beginning, Inter-Domain Routing Table could be configured on SHDHT Border Nodes. Then, a SHDHT Border Node will flood cross domain routing periodically to trigger other Border Nodes update their Inter-Domain Routing Tables.

5.2. EIDs/EID Prefixes Report onto Domain SHDHT Overlay

All SHDHT Nodes of a Domain SHDHT Overlay must be noticed the EID prefixes that local domain overlay responsible for. When a SHDHT Node of a domain overlay receives a report message, it checks if the registered EIDs/EID prefixes are covered by local domain overlay's EID prefixes.

If local domain overlay is responsible for reported EIDs/EID prefixes, SHDHT Node who receives report message will process the message as procedures listed in Section 4.3 and 4.6

Otherwise, if local domain overlay is not responsible for reported EIDs/EID prefixes, SHDHT Node who receives report message will forward it directly to local domain overlay's Border Nodes. Then, Border Nodes will forward the message to corresponding domain overlay based on the Inter-Domain Routing Table.

Suppose in Fig.4, MS2 reports EID prefix 12.2.0/24 to Node 6 of Domain 2.

1. Node6 extracts the EID prefix from report message and finds that the reported EID prefix is 12.2.0/24.

2. Node6 determines that EID prefix 12.2.0/24 is not covered by Domain 2's prefix 16.0.0.0/8.

3. Node6 forwards the report message to BN2.

4. BN2 looks up Inter-Domain Routing Table to find that Domain 1 is responsible for EID prefix 12.2.0/24. BN2 forwards report message to Domain 1's Border Node (BN1).

5. BN1 processes the report message based on procedures introduced in Section 4.3 and 4.6.

5.3. Mapping Request Lookup onto Domain SHDHT Overlay

When SHDHT Map Resolver receives a Map-Request message, it checks if the requested EID is covered by local domain overlay's EID prefixes, i.e. if the requested mapping entry is stored on local domain overlay.

If local domain overlay is responsible for requested EID, SHDHT Map Resolver processes the message based on procedures introduced in Section 4.4 and 4.6.

Otherwise, if the requested EID entry is not stored on local domain overlay, under SHDHT Forward Mode, SHDHT Map Resolver directly forwards the Map-Request to Border Nodes. Border Nodes of local domain overlay then forwards it to corresponding domain overlay based on Inter-Domain Routing Table.

Suppose in Fig.4, ITR2 sends a Map-Request message to SHDHT MR 2 of Domain 2 to get mapping information of EID 12.2.0.1.

1. SHDHT MR2 extracts requested EID from the Map-Request message.

2. SHDHT MR2 determines that requested EID 12.2.0.1 is not covered by Domain 2's prefix 16.0.0.0/8.

3. SHDHT MR2 forwards the Map-Request message to BN2.

4. BN2 extracts requested EID and looks for Inter-Domain Routing Table to find corresponding domain overlay of EID 12.2.0.1.

5. BN2 finds out Domain 1 is responsible for EID 12.2.0.1. BN2 forwards Map-Request message to Domain 1's Border Node (BN1).

6. BN1 processes the Map-Request message based on procedures introduced in Section 4.4 and 4.6.

If the requested EID entry is not stored on local domain overlay, under Recursive Lookup Mode, SHDHT Map Resolver catch the Map-Request message, and send query message to Border Nodes. Border Nodes of local domain overlay query Border Nodes of the corresponding domain overlay responsible for requested EID entry to get related information.

Suppose in Fig.4, ITR2 sends Map-Request message to SHDHT MR 2 to get mapping information of 12.2.0.1.

1. SHDHT MR2 determines the requested EID 12.2.0.1 is not covered by Domain 2's prefix 16.0.0.0/8.

2. SHDHT MR2 catches the Map-Request message and sends a query message for EID 12.2.0.1 to BN2.

3. BN2 finds out Domain 1 is responsible for the requested EID. BN2 sends query message to BN1.

4. BN1 hashes 12.2/16 to be Resource ID to get which SHDHT Node in Domain 1 now maintains information of EIDs covered by EID prefix 12.2/16.

5. Suppose the responsible Node in Domain 1 is Node 2. Node 2 maintains information of EID prefix 12.2.0/24 along with MS2's RLOC information. BN2 queries Node 2 to get the information and sends them to BN1.

6. BN1 sends relative information to SHDHT MR 2.

7. After the recursive lookup procedures, SHDHT MR 2 sends the Map-Request directly to MS2.

6. Security Considerations

TBD

7. IANA Considerations

This document makes no requests to IANA.

8. References

8.1. Normative References

[I-D.meyer-lisp-mn]
Farinacci, D., Lewis, D., Meyer, D., and C. White, "LISP Mobile Node", October 2012.

[I-ietf-lisp-ddt]
Fuller, V., Lewis, D., and V. Ermagan, "LISP Delegated Database Tree", October 2012.

[LISP-DHT]
Mathy, L. and L. Iannone, "LISP-DHT: Towards a DHT to map identifiers onto locators, http://dl.acm.org/citation.cfm?id=1544073", December 2008.

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

[RFC6833] Fuller, V. and D. Farinacci, "LISP Map Server Interface", January 2013.

[RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "LISP Alternative Topology (LISP+ALT)", January 2013.

8.2. Informational References

[I-D.lear-lisp-nerd]
Lear, E., "NERD: A Not-so-novel EID to RLOC Database", April 2012.

Appendix A. Acknowledgments

The authors with to express their thanks to Michael Hoefling for work on Hash space segment of SHDHT overlay. Thanks also go to Dino Farinacci and Darrel Lewis for their suggestions about database structure deployment.

Authors' Addresses

Li Cheng
ZTE Corporation
R&D Building 1, Zijinghua Road No.68
Nanjing, Yuhuatai District 210012
P.R.China

Email: cheng.li2@zte.com.cn

Mo Sun
ZTE Corporation
R&D Building 1, Zijinghua Road No.68
Nanjing, Yuhuatai District 210012
P.R.China

Email: sun.mo@zte.com.cn

LISP Working Group
Internet-Draft
Intended status: Informational
Expires: April 18, 2013

J. N. Chiappa
Yorktown Museum of Asian Art
October 15, 2012

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

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

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: August 29, 2016

L. Iannone
Telecom ParisTech
D. Lewis
Cisco Systems, Inc.
D. Meyer
Brocade
V. Fuller

February 26, 2016

LISP EID Block
draft-ietf-lisp-eid-block-13.txt

Abstract

This is a direction to IANA to allocate a /32 IPv6 prefix for use with the Locator/ID Separation Protocol (LISP). The prefix will be used for local intra-domain routing and global endpoint identification, by sites deploying LISP as EID (Endpoint IDentifier) addressing space.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on August 29, 2016.

Table of Contents

1. Introduction

This document directs the IANA to allocate a /32 IPv6 prefix for use with the Locator/ID Separation Protocol (LISP - [RFC6830]), LISP Map Server ([RFC6833]), LISP Alternative Topology (LISP+ALT - [RFC6836]) (or other) mapping systems, and LISP Interworking ([RFC6832]).

This block will be used as global Endpoint IDentifier (EID) space.

2. Definition of Terms

The present document does not introduce any new term with respect to the set of LISP Specifications ([RFC6830], [RFC6831], [RFC6832], [RFC6833], [RFC6834], [RFC6835], [RFC6836], [RFC6837]), but assumes that the reader is familiar with the LISP terminology. [I-D.ietf-lisp-introduction] provides an introduction to the LISP technology, including its terminology.

3. Rationale and Intent

Discussion within the LISP Working Group led to identify several scenarios in which the existence of a LISP specific address block brings technical benefits. Hereafter the most relevant scenarios are described:

Early LISP destination detection: With the current specifications, there is no direct way to detect whether or not a certain destination is in a LISP domain or not without performing a LISP mapping lookup. For instance, if an ITR is sending to all types of destinations (i.e., non-LISP destinations, LISP destinations not in the IPv6 EID block, and LISP destinations in the IPv6 EID block) the only way to understand whether or not to encapsulate the traffic is to perform a cache lookup and, in case of a LISP Cache miss, send a Map-Request to the mapping system. In the meanwhile (waiting the Map-Reply), packets may be dropped in order to avoid excessive buffering.

Avoid penalizing non-LISP traffic: In certain circumstances it might be desirable to configure a router using LISP features to natively forward all packets that have not a destination address in the block, hence, no lookup whatsoever is performed and packets destined to non-LISP sites are not penalized in any manner.

Traffic Engineering: In some deployment scenarios it might be desirable to apply different traffic engineering policies for LISP and non-LISP traffic. A LISP specific EID block would allow improved traffic engineering capabilities with respect to LISP vs. non-LISP traffic. In particular, LISP traffic might be identified without having to use DPI techniques in order to parse the encapsulated packet, performing instead a simple inspection of the outer header is sufficient.

Transition Mechanism: The existence of a LISP specific EID block may prove useful in transition scenarios. A non-LISP domain would ask for an allocation in the LISP EID block and use it to deploy LISP in its network. Such allocation will not be announced in the BGP routing infrastructure (cf., Section 4). This approach will allow non-LISP domains to avoid fragmenting their already allocated non-LISP addressing space, which may lead to BGP routing table inflation since it may (rightfully) be announced in the BGP routing infrastructure.

Limit the impact on BGP routing infrastructure: As described in the previous scenario, LISP adopters will avoid fragmenting their addressing space, since fragmentation would negatively impact the BGP routing infrastructure. Adopters will use addressing space from the EID block, which might be announced in large aggregates and in a tightly controlled manner only by proxy xTRs.

Is worth mentioning that new use cases can arise in the future, due to new and unforeseen scenarios.

Furthermore, the use of a dedicated address block will give a tighter control, especially filtering, over the traffic in the initial experimental phase, while facilitating its large-scale deployment.

[RFC3692] considers assigning experimental and testing numbers useful, and the request of a reserved IPv6 prefix is a perfect match of such practice. The present document follows the guidelines provided in [RFC3692], with one exception. [RFC3692] suggests the use of values similar to those called "Private Use" in [RFC5226], which by definition are not unique. One of the purposes of the present request to IANA is to guarantee uniqueness to the EID block. The lack thereof would result in a lack of real utility of a reserved IPv6 prefix.

4. Expected use

Sites planning to deploy LISP may request a prefix in the IPv6 EID

block. Such prefixes will be used for routing and endpoint identification inside the site requesting it. Mappings related to such prefix, or part of it, will be made available through the mapping system in use and registered to one or more Map Server(s).

The EID block must be used for LISP experimentation and must not be advertised in the form of more specific route advertisements in the non-LISP inter-domain routing environment. Interworking between the EID block sub-prefixes and the non-LISP Internet is done according to [RFC6832] and [RFC7215].

As the LISP adoption progresses, the EID block may potentially have a reduced impact on the BGP routing infrastructure, compared to the case of having the same number of adopters using global unicast space allocated by RIRs ([MobiArch2007]). From a short-term perspective, the EID block offers potentially large aggregation capabilities since it is announced by PxTRs possibly concentrating several contiguous prefixes. This trend should continue with even lower impact from a long-term perspective, since more aggressive aggregation can be used, potentially leading at using few PxTRs announcing the whole EID block ([FIABook2010]).

The EID block will be used only at configuration level, it is recommended not to hard-code in any way the IPv6 EID block in the router hardware. This allows avoiding locking out sites that may want to switch to LISP while keeping their own IPv6 prefix, which is not in the IPv6 EID block. Furthermore, in the case of a future permanent allocation, the allocated prefix may differ from the experimental temporary prefix allocated during the experimentation phase.

With the exception of PITR case (described in Section 8) prefixes out of the EID block must not be announced in the BGP routing infrastructure.

5. Block Dimension

The working group reached consensus on an initial allocation of a /32 prefix. The reason of such consensus is manifold:

- o The working group agreed that /32 prefix is sufficiently large to cover initial allocation and requests for prefixes in the EID space in the next few years for very large-scale experimentation and deployment.

- o As a comparison, it is worth mentioning that the current LISP Beta Network ([BETA]) is using a /32 prefix, with more than 250 sites

using a /48 sub prefix. Hence, a /32 prefix appears sufficiently large to allow the current deployment to scale up and be open for interoperation with independent deployments using EIDs in the new /32 prefix.

- o A /32 prefix is sufficiently large to allow deployment of independent (commercial) LISP enabled networks by third parties, but may as well boost LISP experimentation and deployment.

- o The use of a /32 prefix is in line with previous similar prefix allocation for tunneling protocols ([RFC3056]).

6. 3+3 Allocation Plan

This document requests IANA to initially assign a /32 prefix out of the IPv6 addressing space for use as EID in LISP (Locator/ID Separation Protocol).

IANA allocates the requested address space by MMMM/YYYY0 for a duration of 3 (three) initial years (through MMMM/YYYY3), with an option to extend this period by 3 (three) more years (until MMMM/YYYY6). By the end of the first period, the IETF will provide a decision on whether to transform the prefix in a permanent assignment or to put it back in the free pool (see Section 7 for more information).

[RFC Editor: please replace MMMM and all its occurrences in the document with the month of publication as RFC.]

[RFC Editor: please replace YYYY0 and all its occurrences in the document with the year of publication as RFC.]

[RFC Editor: please replace YYYY3 and all its occurrences in the document with the year of publication as RFC plus 3 years, e.g., if published in 2016 then put 2019.]

[RFC Editor: please replace YYYY6 and all its occurrences in the document with the year of publication as RFC plus 6 years, e.g., if published in 2016 then put 2022.]

In the first case, i.e., if the IETF decides to transform the block in a permanent allocation, the EID block allocation period will be extended for three years (until MMMM/YYYY6) so to give time to the IETF to define the final size of the EID block and create a transition plan. The transition of the EID block into a permanent allocation has the potential to pose policy issues (as recognized in [RFC2860], section 4.3) and hence discussion with the IANA, the RIR

communities, and the IETF community will be necessary to determine appropriate policy for permanent EID block allocation and management. Note as well that the final permanent allocation may differ from the initial experimental assignment, hence, it is recommended not to hard-code in any way the experimental EID block on LISP-capable devices.

In the latter case, i.e., if the IETF decides to stop the EID block experimental use, by MMMM/YYYY3 all temporary prefix allocations in such address range must expire and be released, so that the entire /32 is returned to the free pool.

The allocation and management of the EID block for the initial 3 years period (and the optional 3 more years) is detailed in [I-D.ietf-lisp-eid-block-mgmnt].

7. Allocation Lifetime

If no explicit action is carried out by the end of the experiment (by MMMM/YYYY3) it is automatically considered that there was no sufficient interest in having a permanent allocation and the address block will be returned to the free pool.

Otherwise, if the LISP Working Group recognizes that there is value in having a permanent allocation then explicit action is needed.

In order to trigger the process for a permanent allocation a document is required. Such document has to articulate the rationale why a permanent allocation would be beneficial. More specifically, the document has to detail the experience gained during experimentation and all of the technical benefits provided by the use of a LISP specific prefix. Such technical benefits are expected to lay in the scenarios described in Section 3, however, new unforeseen benefits may appear during experimentation. The description should be sufficiently articulate so to allow to provide an estimation of what should be the size of the permanent allocation. Note however that, as explained in Section 6, it is up to IANA to decide which address block will be used as permanent allocation and that such block may be different from the temporary experimental allocation.

8. Routing Considerations

In order to provide connectivity between the Legacy Internet and LISP sites, PITRs announcing large aggregates (ideally one single large aggregate) of the IPv6 EID block could be deployed. By doing so, PITRs will attract traffic destined to LISP sites in order to

encapsulate and forward it toward the specific destination LISP site. Routers in the Legacy Internet must treat announcements of prefixes from the IPv6 EID block as normal announcements, applying best current practice for traffic engineering and security.

Even in a LISP site, not all routers need to run LISP elements. In particular, routers that are not at the border of the local domain, used only for intra-domain routing, do not need to provide any specific LISP functionality but must be able to route traffic using addresses in the IPv6 EID block.

For the above-mentioned reasons, routers that do not run any LISP element, must not include any special handling code or hardware for addresses in the IPv6 EID block. In particular, it is recommended that the default router configuration does not handle such addresses in any special way. Doing differently could prevent communication between the Legacy Internet and LISP sites or even break local intra-domain connectivity.

9. Security Considerations

This document does not introduce new security threats in the LISP architecture nor in the legacy Internet architecture.

10. IANA Considerations

This document instructs the IANA to assign a /32 IPv6 prefix for use as the global LISP EID space using a hierarchical allocation as outlined in [RFC5226] and summarized in Table 1.

This document does not specify any specific value for the requested address block but suggests that should come from the 2000::/3 Global Unicast Space. IANA is not requested to issue an AS0 ROA (Route Origin Attestation [RFC6491]), since the Global EID Space will be used for routing purposes.

Attribute	Value
Address Block	2001:5::/32
Name	EID Space for LISP
RFC	[This Document]
Allocation Date	2015
Termination Date	MMMM/YYYY3 [1]
Source	True [2]
Destination	True
Forwardable	True
Global	True
Reserved-by-protocol	True [3]

[1] According to the 3+3 Plan outlined in this document termination date can be postponed to MMMM/YYYY6. [2] Can be used as a multicast source as well. [3] To be used as EID space by LISP [RFC6830] enabled routers.

Table 1: Global EID Space

[IANA: Please update the Termination Date and footnote [1] in the Special-Purpose Address Registry when the I-D is published as RFC.]

The reserved address space is requested for a period of time of three initial years starting in MMMM/YYYY0 (until MMMM/YYYY3), with an option to extend it by three years (until MMMM/YYYY6) up on decision of the IETF (see Section 6 and Section 7). Following the policies outlined in [RFC5226], upon IETF Review, by MMMM/YYYY3 decision should be made on whether to have a permanent EID block assignment. If no explicit action is taken or if the IETF review outcome will be that is not worth to have a reserved prefix as global EID space, the whole /32 will be taken out from the IPv6 Special Purpose Address Registry and put back in the free pool managed by IANA.

Allocation and management of the Global EID Space is detailed in a different document. Nevertheless, all prefix allocations out of this space must be temporary and no allocation must go beyond MMMM/YYYY3 unless the IETF Review decides for a permanent Global EID Space assignment.

11. Acknowledgments

Special thanks to Roque Gagliano for his suggestions and pointers. Thanks to Alvaro Retana, Deborah Brungard, Ron Bonica, Damien Saucez, David Conrad, Scott Bradner, John Curran, Paul Wilson, Geoff Huston,

Wes George, Arturo Servin, Sander Steffann, Brian Carpenter, Roger Jorgensen, Terry Manderson, Brian Haberman, Adrian Farrel, Job Snijders, Marla Azinger, Chris Morrow, and Peter Schoenmaker, for their insightful comments. Thanks as well to all participants to the fruitful discussions on the IETF mailing list.

The work of Luigi Iannone has been partially supported by the ANR-13-INFR-0009 LISP-Lab Project (www.lisp-lab.org) and the EIT KIC ICT-Labs SOFNETS Project.

12. References

12.1. Normative References

[I-D.ietf-lisp-eid-block-mgmnt]
Iannone, L., Jorgensen, R., Conrad, D., and G. Huston, "LISP EID Block Management Guidelines", draft-ietf-lisp-eid-block-mgmnt-06 (work in progress), August 2015.

[RFC2860] Carpenter, B., Baker, F., and M. Roberts, "Memorandum of Understanding Concerning the Technical Work of the Internet Assigned Numbers Authority", RFC 2860, DOI 10.17487/RFC2860, June 2000, <http://www.rfc-editor.org/info/rfc2860>.

[RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, DOI 10.17487/RFC3692, January 2004, <http://www.rfc-editor.org/info/rfc3692>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <http://www.rfc-editor.org/info/rfc6830>.

[RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, DOI 10.17487/RFC6831, January 2013, <http://www.rfc-editor.org/info/rfc6831>.

[RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller,

"Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <http://www.rfc-editor.org/info/rfc6832>.

[RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <http://www.rfc-editor.org/info/rfc6833>.

[RFC6834] Iannone, L., Saucez, D., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning", RFC 6834, DOI 10.17487/RFC6834, January 2013, <http://www.rfc-editor.org/info/rfc6834>.

[RFC6835] Farinacci, D. and D. Meyer, "The Locator/ID Separation Protocol Internet Groper (LIG)", RFC 6835, DOI 10.17487/RFC6835, January 2013, <http://www.rfc-editor.org/info/rfc6835>.

[RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <http://www.rfc-editor.org/info/rfc6836>.

[RFC6837] Lear, E., "NERD: A Not-so-novel Endpoint ID (EID) to Routing Locator (RLOC) Database", RFC 6837, DOI 10.17487/RFC6837, January 2013, <http://www.rfc-editor.org/info/rfc6837>.

12.2. Informative References

[BETA] LISP Beta Network, "http://www.lisp4.net".

[FIABook2010]
L. Iannone, T. Leva, "Modeling the economics of Loc/ID Separation for the Future Internet.", Towards the Future Internet - Emerging Trends from the European Research, Pages 11-20, ISBN: 9781607505389, IOS Press , May 2010.

[I-D.ietf-lisp-introduction]
Cabellos-Aparicio, A. and D. Saucez, "An Architectural Introduction to the Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-introduction-13 (work in progress), April 2015.

[MobiArch2007]
B. Quoitin, L. Iannone, C. de Launois, O. Bonaventure,

"Evaluating the Benefits of the Locator/Identifier Separation", The 2nd ACM-SIGCOMM International Workshop on Mobility in the Evolving Internet Architecture (MobiArch'07) , August 2007.

[RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, DOI 10.17487/RFC3056, February 2001, <http://www.rfc-editor.org/info/rfc3056>.

[RFC6491] Manderson, T., Vegoda, L., and S. Kent, "Resource Public Key Infrastructure (RPKI) Objects Issued by IANA", RFC 6491, DOI 10.17487/RFC6491, February 2012, <http://www.rfc-editor.org/info/rfc6491>.

[RFC7215] Jakab, L., Cabellos-Aparicio, A., Coras, F., Domingo-Pascual, J., and D. Lewis, "Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations", RFC 7215, DOI 10.17487/RFC7215, April 2014, <http://www.rfc-editor.org/info/rfc7215>.

Appendix A. Document Change Log

[RFC Editor: Please remove this section on publication as RFC]

Version 13 Posted MMMM 2016.

- o Changed I-D type from "Informational" to "Experimental" as requested by A. Retana during IESG review.

- o Dropped the appendix "LISP Terminology"; replaced by pointer to the LISP Introduction document.

- o Added Section 7 to clarify the process after the 3 years experimental allocation.

- o Modified the dates, introducing variables, so to allow RFC Editor to easily update dates by publication as RFC.

Version 12 Posted May 2015.

- o Fixed typos and references as suggested by the Gen-ART and OPS-DIR review.

Version 11 Posted April 2015.

- o In Section 4, deleted contradictory text on EID prefix advertisement in non-LISP inter-domain routing environments.

- o In Section 3 deleted the "Avoid excessive strech" bullet, because confusing.

- o Deleted last bullet of the list in Section 3 because retundant w.r.t. global content of the document.

Version 10 Posted January 2015.

- o Keep alive version

Version 09 Posted July 2014.

- o Few Editorial modifications as requested by D. Saucez, as shepherd, during the write up of the document.

- o Allocation date postponed to beginning 2015, as suggested by D. Saucez.

Version 08 Posted January 2014.

- o Modified Section 4 as suggested by G. Houston.

Version 07 Posted November 2013.

- o Modified the document so to request a /32 allocation, as for the consensus reached during IETF 88th.

Version 06 Posted October 2013.

- o Clarified the rationale and intent of the EID block request with respect to [RFC3692], as suggested by S. Bradner and J. Curran.

- o Extended Section 3 by adding the transion scenario (as suggested by J. Curran) and the TE scenario. The other scenarios have been also edited.

- o Section 6 has been re-written to introduce the 3+3 allocation plan as suggested by B. Haberman and discussed during 86th IETF.

- o Section 10 has also been updated to the 3+3 years allocation plan.

- o Moved Section 11 at the end of the document.

- o Changed the original Definition of terms to an appendix.

Version 05 Posted September 2013.

- o No changes.

Version 04 Posted February 2013.

- o Added Table 1 as requested by IANA.
- o Transformed the prefix request in a temporary request as suggested by various comments during IETF Last Call.
- o Added discussion about short/long term impact on BGP in Section 4 as requested by B. Carpenter.

Version 03 Posted November 2012.

- o General review of Section 5 as requested by T. Manderson and B. Haberman.
- o Dropped RFC 2119 Notation, as requested by A. Farrel and B. Haberman.
- o Changed "IETF Consensus" to "IETF Review" as pointed out by Roque Gagliano.
- o Changed every occurrence of "Map-Server" and "Map-Resolver" with "Map Server" and "Map Resolver" to make the document consistent with [RFC6833]. Thanks to Job Snijders for pointing out the issue.

Version 02 Posted April 2012.

- o Fixed typos, nits, references.
- o Deleted reference to IANA allocation policies.

Version 01 Posted October 2011.

- o Added Section 5.

Version 00 Posted July 2011.

- o Updated section "IANA Considerations"
- o Added section "Rationale and Intent" explaining why the EID block allocation is useful.
- o Added section "Expected Use" explaining how sites can request and use a prefix in the IPv6 EID Block.

- o Added section "Action Plan" suggesting IANA to avoid allocating address space adjacent the allocated EID block in order to accommodate future EID space requests.

- o Added section "Routing Consideration" describing how routers not running LISP deal with the requested address block.

- o Added the present section to keep track of changes.

- o Rename of draft-meyer-lisp-eid-block-02.txt.

Authors' Addresses

Luigi Iannone
Telecom ParisTech

Email: ggx@gigix.net

Darrel Lewis
Cisco Systems, Inc.

Email: darlewis@cisco.com

David Meyer
Brocade

Email: dmm@1-4-5.net

Vince Fuller

Email: vaf@vaf.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 24 March 2022

A. Cabellos
UPC-BarcelonaTech
D. Saucez (Ed.)
Inria

20 September 2021

An Architectural Introduction to the Locator/ID Separation Protocol (LISP)
draft-ietf-lisp-introduction-15

Abstract

This document describes the architecture of the Locator/ID Separation Protocol (LISP), making it easier to read the rest of the LISP specifications and providing a basis for discussion about the details of the LISP protocols. This document is used for introductory purposes, more details can be found in [I-D.ietf-lisp-rfc6830bis] and [I-D.ietf-lisp-rfc6833bis], the protocol specifications.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document introduces the Locator/ID Separation Protocol (LISP) architecture ([I-D.ietf-lisp-rfc6830bis], [I-D.ietf-lisp-rfc6833bis]), its main operational mechanisms and its design rationale. Fundamentally, LISP is built following a well-known architectural idea: decoupling the IP address overloaded semantics. Indeed and as pointed out by Noel Chiappa [RFC4984], currently IP addresses both identify the topological location of a network attachment point as well as the node's identity. However, nodes and routing have fundamentally different requirements. On the one hand, routing systems require that addresses are aggregatable and have topological meaning, on the other hand, nodes require to be identified independently of their current location [RFC4984].

LISP creates two separate namespaces, EIDs (End-host IDentifiers) and RLOCs (Routing LOCators), both are syntactically identical to the current IPv4 and IPv6 addresses. However, EIDs are used to uniquely identify nodes irrespective of their topological location and are typically routed intra-domain. RLOCs are assigned topologically to network attachment points and are typically routed inter-domain. With LISP, the edge of the Internet (where the nodes are connected) and the core (where inter-domain routing occurs) can be logically separated. LISP-capable routers interconnect the two logical spaces. LISP also introduces a database, called the Mapping System, to store and retrieve mappings between identity and location. LISP-capable routers exchange packets over the Internet core by encapsulating them to the appropriate location.

In summary:

* RLOCs have meaning only in the underlay network, that is the underlying core routing system.

* EIDs have meaning only in the overlay network, which is the encapsulation relationship between LISP-capable routers.

* The LISP edge maps EIDs to RLOCs

* Within the underlay network, RLOCs have both locator and identifier semantics

* An EID within a LISP site carries both identifier and locator semantics to other nodes within that site

* An EID within a LISP site carries identifier and limited locator semantics to nodes at other LISP sites (i.e., enough locator information to tell that the EID is external to the site)

The relationship described above is not unique to LISP and it is common to other overlay technologies.

The initial motivation in the LISP effort is to be found in the routing scalability problem [RFC4984], where, if LISP were to be completely deployed, the Internet core would be populated with RLOCs while Traffic Engineering mechanisms would be pushed to the Mapping System. In such scenario RLOCs are quasi-static (i.e., low churn), hence making the routing system scalable [Quoitin], while EIDs can roam anywhere with no churn to the underlying global routing system. [RFC7215] discusses the impact of LISP on the global routing system during the transition period. However, the separation between location and identity that LISP offers makes it suitable for use in additional scenarios such as Traffic Engineering (TE), multihoming, and mobility among others.

This document describes the LISP architecture and its main operational mechanisms as well as its design rationale. It is important to note that this document does not specify or complement the LISP protocol. The interested reader should refer to the main LISP specifications [I-D.ietf-lisp-rfc6830bis] and [I-D.ietf-lisp-rfc6833bis], as well as the complementary documents [RFC6831], [RFC6832], [I-D.ietf-lisp-6834bis], [RFC6835], [RFC6836], [RFC7052] for the protocol specifications along with the LISP deployment guidelines [RFC7215].

2. Definition of Terms

Endpoint IDentifier (EID): EIDs are addresses used to uniquely identify nodes irrespective of their topological location and are typically routed intra-domain.

Routing LOcator (RLOC): RLOCs are addresses assigned topologically to network attachment points and typically routed inter-domain.

Ingress Tunnel Router (ITR): A LISP-capable router that encapsulates packets from a LISP site towards the core network.

Egress Tunnel Router (ETR): A LISP-capable router that decapsulates packets from the core of the network towards a LISP site.

xTR: A router that implements both ITR and ETR functionalities.

Map-Request: A LISP signaling message used to request an EID-to-RLOC mapping.

Map-Reply: A LISP signaling message sent in response to a Map-Request that contains a resolved EID-to-RLOC mapping.

Map-Register: A LISP signaling message used to register an EID-to-RLOC mapping.

Map-Notify: A LISP signaling message sent in response of a Map-Register to acknowledge the correct reception of an EID-to-RLOC mapping.

This document describes the LISP architecture and does not introduce any new term. The reader is referred to [I-D.ietf-lisp-rfc6830bis] and [I-D.ietf-lisp-rfc6833bis], [RFC6831], [RFC6832], [I-D.ietf-lisp-6834bis], [RFC6835], [RFC6836], [RFC7052], [RFC7215] for the complete definition of terms.

3. LISP Architecture

This section presents the LISP architecture, it first details the design principles of LISP and then it proceeds to describe its main aspects: data-plane, control-plane, and internetworking mechanisms.

3.1. Design Principles

The LISP architecture is built on top of four basic design principles:

* Locator/Identifier split: By decoupling the overloaded semantics of the current IP addresses the Internet core can be assigned identity meaningful addresses and hence, can use aggregation to scale. Devices are assigned with relatively opaque topologically meaningful addresses that are independent of their topological location.

* Overlay architecture: Overlays route packets over the current Internet, allowing deployment of new protocols without changing the current infrastructure hence, resulting into a low deployment cost.

* Decoupled data-plane and control-plane: Separating the data-plane from the control-plane allows them to scale independently and use different architectural approaches. This is important given that they typically have different requirements and allows for other data-planes to be added. Even though the data-plane and the control-plane are decoupled, they are not completely isolated because the LISP data-plane may trigger control-plane activity.

* Incremental deployability: This principle ensures that the protocol interoperates with the legacy Internet while providing some of the targeted benefits to early adopters.

3.2. Overview of the Architecture

LISP splits architecturally the core from the edge of the Internet by creating two separate namespaces: Endpoint Identifiers (EIDs) and Routing LOCators (RLOCs). The edge consists of LISP sites (e.g., an Autonomous System) that use EID addresses. EIDs are IPv4 or IPv6 addresses that uniquely identify communication end-hosts and are assigned and configured by the same mechanisms that exist at the time of this writing. EIDs do not contain inter-domain topological information and because of this, EIDs are usually routable at the edge (within LISP sites) but not in the core; see Section Section 3.5 for discussion of LISP site internetworking with non-LISP sites and domains in the Internet.

LISP sites (at the edge) are connected to the interconnecting core by means of LISP-capable routers (e.g., border routers). LISP sites are connected across the interconnecting core using tunnels between the LISP-capable routers. When packets originated from a LISP site are flowing towards the core network, they ingress into an encapsulated tunnel via an Ingress Tunnel Router (ITR). When packets flow from the core network to a LISP site, they egress from an encapsulated tunnel to an Egress Tunnel Router (ETR). An xTR is a router which can perform both ITR and ETR operations. In this context ITRs encapsulate packets while ETRs decapsulate them, hence LISP operates as an overlay on top of the current Internet core.

```
                        /-----------------\                 ---
                        |     Mapping     |                  |
                        .     System      |                  | Control
                      -|                 |`,                 | Plane
                    ,' \-----------------/  .                |
                   /                         |              ---
   ,..,           _                 _,.....,,  |      ,..,    |
  /     `        ,'       ,-`         `',      |     /     `   |
 /        \ +-----+    ,'              `,  +-----+ /        \ |
 |  EID   |-| xTR |--/        RLOC     ,--| xTR |-|  EID   | | Data
 | Space  |-|     |--|       Space     |--|     |-| Space  | | Plane
 \        / +-----+  .                 /  +-----+ \        / |
  `.    .'            `.              ,'            `.    .' |
    `'-`                `.,        ,.'               `'-`   ---
                           ``'''``

  LISP Site (Edge)            Core              LISP Site (Edge)
```

Figure 1: A schema of the LISP Architecture.

With LISP, the core uses RLOCs, an RLOC is an IPv4 or IPv6 address assigned to an core-facing network interface of an ITR or ETR.

A database which is typically distributed, called the Mapping System, stores mappings between EIDs and RLOCs. Such mappings relate the identity of the devices attached to LISP sites (EIDs) to the set of RLOCs configured at the LISP-capable routers servicing the site. Furthermore, the mappings also include traffic engineering policies and can be configured to achieve multihoming and load balancing. The LISP Mapping System is conceptually similar to the DNS where it is organized as a distributed multi-organization network database. With LISP, ETRs register mappings while ITRs retrieve them.

Finally, the LISP architecture emphasizes incremental deployment. Given that LISP represents an overlay to the current Internet architecture, end hosts as well as intra and inter-domain routers remain unchanged, and the only required changes to the existing infrastructure are to routers connecting the EID space with the RLOC space. Additionally, LISP requires the deployment of an independent Mapping System, such distributed database is a new network entity.

The following describes a simplified packet flow sequence between two nodes that are attached to LISP sites. Please note that typical LISP-capable routers are xTRs (both ITR and ETR). Client HostA wants to send a packet to server HostB.

```
                        /----------------\
                        |     Mapping    |
                        |     System     |
                       .|                |-
                      ` \----------------/ `.
                    ,`                       \
                   /                          `.
                  ,         _,..-..,,           ,
                 /        -`          `-,        \
               .'        ,               \        `,
               `        ,                 \        ,
        +-----+     |                    | RLOC_B1+-----+
HostA   |     |     |        RLOC        |-------|     |  HostB
EID_A--|ITR_A|----|        Space         |       |ETR_B|--EID_B
        |     | RLOC_A1                  |-------|     |
        +-----+     |                    | RLOC_B2+-----+
                     ,                  /
                      \                /
                       `',          ,-`
                          ``,,_,,``
```

Figure 2: Packet flow sequence in LISP.

1. HostA retrieves the EID_B of HostB, typically querying the DNS and obtaining an A or AAAA record. Then it generates an IP packet as in the Internet, the packet has source address EID_A and destination address EID_B.

2. The packet is forwarded towards ITR_A in the LISP site using standard intra-domain mechanisms.

3. ITR_A upon receiving the packet queries the Mapping System to retrieve the locator of ETR_B that is servicing HostB's EID_B. In order to do so it uses a LISP control message called Map-Request, the message contains EID_B as the lookup key. In turn it receives another LISP control message called Map-Reply, the message contains two locators: RLOC_B1 and RLOC_B2 along with traffic engineering policies: priority and weight per locator. Note that a Map-Reply can contain more locators if needed. ITR_A can cache the mapping in a local storage to speed-up forwarding of subsequent packets.

4. ITR_A encapsulates the packet towards RLOC_B1 (chosen according to the priorities/weights specified in the mapping). The packet contains two IP headers, the outer header has RLOC_A1 as source and RLOC_B1 as destination, the inner original header has EID_A as source and EID_B as destination. Furthermore ITR_A adds a LISP header, more details about LISP encapsulation can be found in Section 3.3.1.

5. The encapsulated packet is forwarded over the interconnecting core as a normal IP packet, making the EID invisible from the core.

6. Upon reception of the encapsulated packet by ETR_B, it decapsulates the packet and forwards it to HostB.

3.3. Data-Plane

This section provides a high-level description of the LISP data-plane, which is specified in detail in [I-D.ietf-lisp-rfc6830bis]. The LISP data-plane is responsible for encapsulating and decapsulating data packets and caching the appropriate forwarding state. It includes two main entities, the ITR and the ETR, both are LISP capable routers that connect the EID with the RLOC space (ITR) and vice versa (ETR).

3.3.1. LISP Encapsulation

ITRs encapsulate data packets towards ETRs. LISP data packets are encapsulated using UDP (port 4341), the source port is usually selected by the ITR using a 5-tuple hash of the inner header (so to be consistent in case of multi-path solutions such as ECMP [RFC2992]) and ignored on reception. LISP data packets are often encapsulated in UDP packets that include a zero checksum [RFC6935] [RFC6936] that may not be verified when it is received, because LISP data packets typically include an inner transport protocol header with a non-zero checksum. The use of UDP zero checksums over IPv6 for all tunneling protocols like LISP is subject to the applicability statement in [RFC6936]. If LISP data packets are encapsulated in UDP packets with non-zero checksums, the outer UDP checksums are verified when the UDP packets are received, as part of normal UDP processing.

LISP-encapsulated packets also include a LISP header (after the UDP header and before the original IP header). The LISP header is prepended by ITRs and striped by ETRs. It carries reachability information (see more details in Section 4.2) and the Instance ID field. The Instance ID field is used to distinguish traffic to/from different tenant address spaces at the LISP site and that may use overlapped but logically separated EID addressing.

Overall, LISP works on 4 headers, the inner header the source constructed, and the 3 headers a LISP encapsulator prepends ("outer" to "inner"):

1. Outer IP header containing RLOCs as source and destination addresses. This header is originated by ITRs and stripped by ETRs.

2. UDP header (port 4341), usually with zero checksum. This header is originated by ITRs and stripped by ETRs.

3. LISP header that contains various forwarding-plane features (such as reachability) and an Instance ID field. This header is originated by ITRs and stripped by ETRs.

4. Inner IP header containing EIDs as source and destination addresses. This header is created by the source end-host and is left unchanged by LISP data plane processing on the ITR and ETR.

Finally, in some scenarios Re-encapsulating and/or Recursive tunnels are useful to choose a specified path in the underlay network, for instance to avoid congestion or failure. Re-encapsulating tunnels are consecutive LISP tunnels and occur when a decapsulator (an ETR action) removes a LISP header and then acts as an encapsultor (an ITR

action) to prepend another one. On the other hand, Recursive tunnels are nested tunnels and are implemented by using multiple LISP encapsulations on a packet. Such functions are implemented by Reencapsulating Tunnel Routers (RTRs). An RTR can be thought of as a router that first acts as an ETR by decapsulating packets and then as an ITR by encapsulating them towards another locator, more information can be found at [I-D.ietf-lisp-rfc6830bis] and [I-D.ietf-lisp-rfc6833bis].

3.3.2. LISP Forwarding State

In the LISP architecture, ITRs keep just enough information to route traffic flowing through them. Meaning that, ITRs retrieve from the LISP Mapping System mappings between EID-prefixes (blocks of EIDs) and RLOCs that are used to encapsulate packets. Such mappings are stored in a local cache called the LISP Map-Cache for subsequent packets addressed to the same EID prefix. Note that, in case of overlapping EID-prefixes, following a single request, the ITR may receive a set of mappings, covering the requested EID-prefix and all more-specifics (cf., Section 5.5 [I-D.ietf-lisp-rfc6833bis]). Mappings include a (Time-to-Live) TTL (set by the ETR). More details about the Map-Cache management can be found in Section 4.1.

3.4. Control-Plane

The LISP control-plane, specified in [I-D.ietf-lisp-rfc6833bis], provides a standard interface to register and request mappings. The LISP Mapping System is a database that stores such mappings. The following first describes the mappings, then the standard interface to the Mapping System, and finally its architecture.

3.4.1. LISP Mappings

Each mapping includes the bindings between EID prefix(es) and set of RLOCs as well as traffic engineering policies, in the form of priorities and weights for the RLOCs. Priorities allow the ETR to configure active/backup policies while weights are used to load-balance traffic among the RLOCs (on a per-flow basis).

Typical mappings in LISP bind EIDs in the form of IP prefixes with a set of RLOCs, also in the form of IP addresses. IPv4 and IPv6 addresses are encoded using the appropriate Address Family Identifier (AFI) [RFC3232]. However LISP can also support more general address encoding by means of the ongoing effort around the LISP Canonical Address Format (LCAF) [RFC8060].

With such a general syntax for address encoding in place, LISP aims to provide flexibility to current and future applications. For instance LCAFs could support MAC addresses, geo-coordinates, ASCII names and application specific data.

3.4.2. Mapping System Interface

LISP defines a standard interface between data and control planes. The interface is specified in [I-D.ietf-lisp-rfc6833bis] and defines two entities:

Map-Server: A network infrastructure component that learns mappings from ETRs and publishes them into the LISP Mapping System. Typically Map-Servers are not authoritative to reply to queries and hence, they forward them to the ETR. However, they can also operate in proxy-mode, where the ETRs delegate replying to queries to Map-Servers. This setup is useful when the ETR has limited resources (e.g., CPU or power).

Map-Resolver: A network infrastructure component that interfaces ITRs with the Mapping System by proxying queries and in some cases responses.

The interface defines four LISP control messages which are sent as UDP datagrams (port 4342):

Map-Register: This message is used by ETRs to register mappings in the Mapping System and it is authenticated using a shared key between the ETR and the Map-Server.

Map-Notify: When requested by the ETR, this message is sent by the Map-Server in response to a Map-Register to acknowledge the correct reception of the mapping and convey the latest Map-Server state on the EID to RLOC mapping. In some cases a Map-Notify can be sent to the previous RLOCs when an EID is registered by a new set of RLOCs.

Map-Request: This message is used by ITRs or Map-Resolvers to resolve the mapping of a given EID.

Map-Reply: This message is sent by Map-Servers or ETRs in response to a Map-Request and contains the resolved mapping. Please note that a Map-Reply may contain a negative reply if, for example, the queried EID is not part of the LISP EID space. In such cases the ITR typically forwards the traffic natively (non encapsulated) to the public Internet, this behavior is defined to support incremental deployment of LISP.

3.4.3. Mapping System

LISP architecturally decouples control and data-plane by means of a standard interface. This interface glues the data-plane - routers responsible for forwarding data-packets - with the LISP Mapping System - a database responsible for storing mappings.

With this separation in place, the data and control-plane can use different architectures if needed and scale independently. Typically the data-plane is optimized to route packets according to hierarchical IP addresses. However the control-plane may have different requirements, for instance and by taking advantage of the LCAF, the Mapping System may be used to store non-hierarchical keys (such as MAC addresses), requiring different architectural approaches for scalability. Another important difference between the LISP control- and data- planes is that, as a result of the local mapping cache available at ITR, the Mapping System does not need to operate at line-rate.

Many of the existing mechanisms to create distributed systems have been explored and considered for the Mapping System architecture: graph-based databases in the form of LISP+ALT [RFC6836], hierarchical databases in the form of LISP-DDT [RFC8111], monolithic databases in the form of LISP-NERD [RFC6837], flat databases in the form of LISP-DHT [I-D.cheng-lisp-shdht], [Mathy], and a multicast-based database [I-D.curran-lisp-emacs]. Furthermore it is worth noting that, in some scenarios such as private deployments, the Mapping System can operate as logically centralized. In such cases it is typically composed of a single Map-Server/Map-Resolver.

The following focuses on the two mapping systems that have been implemented and deployed (LISP+ALT and LISP-DDT).

3.4.3.1. LISP+ALT

The LISP Alternative Topology (LISP+ALT) [RFC6836] was the first Mapping System proposed, developed and deployed on the LISP pilot network. It is based on a distributed BGP overlay participated by Map-Servers and Map-Resolvers. The nodes connect to their peers through static tunnels. Each Map-Server involved in the ALT topology advertises the EID-prefixes registered by the serviced ETRs, making the EID routable on the ALT topology.

When an ITR needs a mapping it sends a Map-Request to a Map-Resolver that, using the ALT topology, forwards the Map-Request towards the Map-Server responsible for the mapping. Upon reception the Map-Server forwards the request to the ETR that in turn, replies directly to the ITR.

3.4.3.2. LISP-DDT

LISP-DDT [RFC8111] is conceptually similar to the DNS, a hierarchical directory whose internal structure mirrors the hierarchical nature of the EID address space. The DDT hierarchy is composed of DDT nodes forming a tree structure, the leafs of the tree are Map-Servers. On top of the structure there is the DDT root node, which is a particular instance of a DDT node and that matches the entire address space. As in the case of DNS, DDT supports multiple redundant DDT nodes and/or DDT roots. Finally, Map-Resolvers are the clients of the DDT hierarchy and can query either the DDT root and/or other DDT nodes.

```
                          /---------\
                          |         |
                          | DDT Root|
                          |   /0    |
                        ,.\---------/-,
                    ,_,`        |        `',
                _,`             |          `_
             /-------\     /-------\    /-------\
             |  DDT  |     |  DDT  |    |  DDT  |
             | Node  |     | Node  |    | Note  |   ...
             |  0/8  |     |  1/8  |    |  2/8  |
             \-------/     \-------/    \-------/
         _.               _.            . _..,,,_
       _`               _`             \        ```\,,__
+------------+     +------------+   +------------+ +------------+
| Map-Server |     | Map-Server |   | Map-Server | | Map-Server |
| EID-prefix1|     | EID-prefix2|   | EID-prefix3| | EID-prefix4|
+------------+     +------------+   +------------+ +------------+
```

Figure 3: A schematic representation of the DDT tree structure, please note that the prefixes and the structure depicted should be only considered as an example.

The DDT structure does not actually index EID-prefixes but eXtended EID-prefixes (XEID). An XEID-prefix is just the concatenation of the following fields (from most significant bit to less significant bit): Database-ID, Instance ID, Address Family Identifier and the actual EID-prefix. The Database-ID is provided for possible future requirements of higher levels in the hierarchy and to enable the creation of multiple and separate database trees.

In order to resolve a query LISP-DDT operates in a similar way to the DNS but only supports iterative lookups. DDT clients (usually Map-Resolvers) generate Map-Requests to the DDT root node. In response

they receive a newly introduced LISP-control message: a Map-Referral. A Map-Referral provides the list of RLOCs of the set of DDT nodes matching a configured XEID delegation. That is, the information contained in the Map-Referral points to the child of the queried DDT node that has more specific information about the queried XEID-prefix. This process is repeated until the DDT client walks the tree structure (downwards) and discovers the Map-Server servicing the queried XEID. At this point the client sends a Map-Request and receives a Map-Reply containing the mappings. It is important to note that DDT clients can also cache the information contained in Map-Referrals, that is, they cache the DDT structure. This is used to reduce the mapping retrieving latency [Jakab].

The DDT Mapping System relies on manual configuration. That is Map-Resolvers are configured with the set of available DDT root nodes while DDT nodes are configured with the appropriate XEID delegations. Configuration changes in the DDT nodes are only required when the tree structure changes itself, but it doesn't depend on EID dynamics (RLOC allocation or traffic engineering policy changes).

3.5. Internetworking Mechanisms

EIDs are typically identical to either IPv4 or IPv6 addresses and they are stored in the LISP Mapping System, however they are usually not announced in the routing system beyond the local LISP domain. As a result LISP requires an internetworking mechanism to allow LISP sites to speak with non-LISP sites and vice versa. LISP internetworking mechanisms are specified in [RFC6832].

LISP defines two entities to provide internetworking:

Proxy Ingress Tunnel Router (PITR): PITRs provide connectivity from the legacy Internet to LISP sites. PITRs announce in the global routing system blocks of EID prefixes (aggregating when possible) to attract traffic. For each incoming packet from a source not in a LISP site (a non-EID), the PITR LISP-encapsulates it towards the RLOC(s) of the appropriate LISP site. The impact of PITRs in the routing table size of the Default-Free Zone (DFZ) is, in the worst-case, similar to the case in which LISP is not deployed. EID-prefixes will be aggregated as much as possible both by the PITR and by the global routing system.

Proxy Egress Tunnel Router (PETR): PETRs provide connectivity from LISP sites to the legacy Internet. In some scenarios, LISP sites may be unable to send encapsulated packets with a local EID address as a source to the legacy Internet. For instance when Unicast Reverse Path Forwarding (uRPF) is used by Provider Edge routers, or when an intermediate network between a LISP site and a

non-LISP site does not support the desired version of IP (IPv4 or IPv6). In both cases the PETR overcomes such limitations by encapsulating packets over the network. There is no specified provision for the distribution of PETR RLOC addresses to the ITRs.

Additionally, LISP also defines mechanisms to operate with private EIDs [RFC1918] by means of LISP-NAT [RFC6832]. In this case the xTR replaces a private EID source address with a routable one. At the time of this writing, work is ongoing to define NAT-traversal capabilities, that is xTRs behind a NAT using non-routable RLOCs.

PITRs, PETRs and, LISP-NAT enable incremental deployment of LISP, by providing significant flexibility in the placement of the boundaries between the LISP and non-LISP portions of the network, and making it easy to change those boundaries over time.

4. LISP Operational Mechanisms

This section details the main operational mechanisms defined in LISP.

4.1. Cache Management

LISP's decoupled control and data-plane, where mappings are stored in the control-plane and used for forwarding in the data-plane, requires a local cache in ITRs to reduce signaling overhead (Map-Request/Map-Reply) and increase forwarding speed. The local cache available at the ITRs, called Map-Cache, is used by the router to LISP-encapsulate packets. The Map-Cache is indexed by (Instance ID, EID-prefix) and contains basically the set of RLOCs with the associated traffic engineering policies (priorities and weights).

The Map-Cache, as any other cache, requires cache coherence mechanisms to maintain up-to-date information. LISP defines three main mechanisms for cache coherence:

Record Time-To-Live (TTL): Each mapping record contains a TTL set by the ETR, upon expiration of the TTL the ITR can't use the mapping until it is refreshed by sending a new Map-Request.

Solicit-Map-Request (SMR): SMR is an explicit mechanism to update mapping information. In particular a special type of Map-Request can be sent on demand by ETRs to request refreshing a mapping. Upon reception of a SMR message, the ITR must refresh the bindings by sending a Map-Request to the Mapping System. Further uses of SMRs are documented in [I-D.ietf-lisp-rfc6833bis].

Map-Versioning: This optional mechanism piggybacks in the LISP

header of data-packets the version number of the mappings used by an xTR. This way, when an xTR receives a LISP-encapsulated packet from a remote xTR, it can check whether its own Map-Cache or the one of the remote xTR is outdated. If its Map-Cache is outdated, it sends a Map-Request for the remote EID so to obtain the newest mappings. On the contrary, if it detects that the remote xTR Map-Cache is outdated, it sends a SMR to notify it that a new mapping is available. Further details are available in [I-D.ietf-lisp-6834bis].

Finally it is worth noting that in some cases an entry in the map-cache can be proactively refreshed using the mechanisms described in the section below.

4.2. RLOC Reachability

In most cases LISP operates with a pull-based Mapping System (e.g., DDT), this results in an edge to edge pull architecture. In such scenario the network state is stored in the control-plane while the data-plane pulls it on demand. This has consequences concerning the propagation of xTRs reachability/liveness information since pull architectures require explicit mechanisms to propagate this information. As a result LISP defines a set of mechanisms to inform ITRs and PITRS about the reachability of the cached RLOCs:

Locator Status Bits (LSB): LSB is a passive technique, the LSB field is carried by data-packets in the LISP header and can be set by a ETRs to specify which RLOCs of the ETR site are up/down. This information can be used by the ITRs as a hint about the reachability to perform additional checks. Also note that LSB does not provide path reachability status, only hints on the status of RLOCs as such they must not be used over the public Internet and should be coupled with Map-Versioning to prevent race conditions where LSB are interpreted as referring to different RLOCs than intended.

Echo-nonce: This is also a passive technique, that can only operate effectively when data flows bi-directionally between two communicating xTRs. Basically, an ITR piggybacks a random number (called nonce) in LISP data packets, if the path and the probed locator are up, the ETR will piggyback the same random number on the next data-packet, if this is not the case the ITR can set the locator as unreachable. When traffic flow is unidirectional or when the ETR receiving the traffic is not the same as the ITR that transmits it back, additional mechanisms are required. The echo-nonce mechanism must be used in trusted environments only, not over the public Internet.

RLOC-probing: This is an active probing algorithm where ITRs send probes to specific locators, this effectively probes both the locator and the path. In particular this is done by sending a Map-Request (with certain flags activated) on the data-plane (RLOC space) and waiting in return a Map-Reply, also sent on the data-plane. The active nature of RLOC-probing provides an effective mechanism to determine reachability and, in case of failure, switching to a different locator. Furthermore the mechanism also provides useful RTT estimates of the delay of the path that can be used by other network algorithms.

It is worth noting that RLOC probing and Echo-nonce can work together. Specifically if a nonce is not echoed, an ITR could RLOC-probe to determine if the path is up when it cannot tell the difference between a failed bidirectional path or the return path is not used (a unidirectional path).

Additionally, LISP also recommends inferring reachability of locators by using information provided by the underlay, in particular:

ICMP signaling: The LISP underlay -the current Internet- uses the ICMP protocol to signal unreachability (among other things). LISP can take advantage of this and the reception of a ICMP Network Unreachable or ICMP Host Unreachable message can be seen as a hint that a locator might be unreachable, this should lead to perform additional checks.

Underlay routing: Both BGP and IGP carry reachability information, LISP-capable routers that have access to underlay routing information can use it to determine if a given locator or path are reachable.

4.3. ETR Synchronization

All the ETRs that are authoritative to a particular EID-prefix must announce the same mapping to the requesters, this means that ETRs must be aware of the status of the RLOCs of the remaining ETRs. This is known as ETR synchronization.

At the time of this writing LISP does not specify a mechanism to achieve ETR synchronization. Although many well-known techniques could be applied to solve this issue it is still under research, as a result operators must rely on coherent manual configuration

4.4. MTU Handling

Since LISP encapsulates packets it requires dealing with packets that exceed the MTU of the path between the ITR and the ETR. Specifically LISP defines two mechanisms:

Stateless: With this mechanism the effective MTU is assumed from the ITR's perspective. If a payload packet is too big for the effective MTU, and can be fragmented, the payload packet is fragmented on the ITR, such that reassembly is performed at the destination host.

Stateful: With this mechanism ITRs keep track of the MTU of the paths towards the destination locators by parsing the ICMP Too Big packets sent by intermediate routers. ITRs will send ICMP Too Big messages to inform the sources about the effective MTU. Additionally ITRs can use mechanisms such as PMTUD [RFC1191] or PLPMTUD [RFC4821] to keep track of the MTU towards the locators.

In both cases if the packet cannot be fragmented (IPv4 with DF=1 or IPv6) then the ITR drops it and replies with a ICMP Too Big message to the source.

5. Mobility

The separation between locators and identifiers in LISP is suitable for traffic engineering purpose where LISP sites can change their attachment points to the Internet (i.e., RLOCs) without impacting endpoints or the Internet core. In this context, the border routers operate the xTR functionality and endpoints are not aware of the existence of LISP. This functionality is similar to Network Mobility [RFC3963]. However, this mode of operation does not allow seamless mobility of endpoints between different LISP sites as the EID address might not be routable in a visited site. Nevertheless, LISP can be used to enable seamless IP mobility when LISP is directly implemented in the endpoint or when the endpoint roams to an attached xTR. Each endpoint is then an xTR and the EID address is the one presented to the network stack used by applications while the RLOC is the address gathered from the network when it is visited. This functionality is similar to Mobile IP ([RFC5944] and [RFC6275]).

Whenever the device changes of RLOC, the xTR updates the RLOC of its local mapping and registers it to its Map-Server, typically with a low TTL value (1min). To avoid the need of a home gateway, the ITR also indicates the RLOC change to all remote devices that have ongoing communications with the device that moved. The combination of both methods ensures the scalability of the system as signaling is strictly limited the Map-Server and to hosts with which communications are ongoing. In the mobility case the EID-prefix can be as small as a full /32 or /128 (IPv4 or IPv6 respectively) depending on the specific use-case (e.g., subnet mobility vs single VM/Mobile node mobility).

The decoupled identity and location provided by LISP allows it to operate with other layer 2 and layer 3 mobility solutions.

6. Multicast

LISP also supports transporting IP multicast packets sent from the EID space, the operational changes required to the multicast protocols are documented in [RFC6831].

In such scenarios, LISP may create multicast state both at the core and at the sites (both source and receiver). When signaling is used to create multicast state at the sites, LISP routers unicast encapsulate PIM Join/Prune messages from receiver to source sites. At the core, ETRs build a new PIM Join/Prune message addressed to the RLOC of the ITR servicing the source. An simplified sequence is shown below

1. An end-host willing to join a multicast channel sends an IGMP report. Multicast PIM routers at the LISP site propagate PIM Join/Prune messages (S-EID, G) towards the ETR.

2. The join message flows to the ETR, upon reception the ETR builds two join messages, the first one unicast LISP-encapsulates the original join message towards the RLOC of the ITR servicing the source. This message creates (S-EID, G) multicast state at the source site. The second join message contains as destination address the RLOC of the ITR servicing the source (S-RLOC, G) and creates multicast state at the core.

3. Multicast data packets originated by the source (S-EID, G) flow from the source to the ITR. The ITR LISP-encapsulates the multicast packets, the outter header includes its own RLOC as the source (S-RLOC) and the original multicast group address (G) as the destination. Please note that multicast group address are logical and are not resolved by the mapping system. Then the multicast packet is transmitted through the core towards the receiving ETRs that decapsulates the packets and sends them using the receiver’s site multicast state.

Please note that the inner and outer multicast addresses are in general different, unless in specific cases where the underlay provider implements a tight control on the overlay. LISP specifications already support all PIM modes [RFC6831]. Additionally, LISP can support as well non-PIM mechanisms in order to maintain multicast state.

When multicast sources and receivers are active at LISP sites, and the core network between the sites does not provide multicast support, a signal-free mechanism can be used to create an overlay that will allow multicast traffic to flow between sites and connect the multicast trees at the different sites [RFC8378]. Registrations from the different receiver sites will be merged at the mapping system to assemble a multicast-replication-list inclusive of all Routing Locators (RLOCs) that lead to receivers for a particular multicast group or multicast channel. The replication list for each specific multicast entry is maintained as a database mapping entry in the LISP mapping system.

7. Use Cases

7.1. Traffic Engineering

A LISP site can strictly impose via which ETRs the traffic must enters the LISP site network even though the path followed to reach the ETR is not under the control of the LISP site. This fine control is implemented with the mappings. When a remote site is willing to send traffic to a LISP site, it retrieves the mapping associated to the destination EID via the mapping system. The mapping is sent directly by an authoritative ETR of the EID and is not altered by any intermediate network.

A mapping associates a list of RLOCs to an EID prefix. Each RLOC corresponds to an interface of an ETR (or set of ETRs) that is able to correctly forward packets to EIDs in the prefix. Each RLOC is tagged with a priority and a weight in the mapping. The priority is used to indicates which RLOCs should be preferred to send packets (the least preferred ones being provided for backup purpose). The weight permits to balance the load between the RLOCs with the same priority, proportionally to the weight value.

As mappings are directly issued by the authoritative ETR of the EID and are not altered while transmitted to the remote site, it offers highly flexible incoming inter-domain traffic engineering with even the possibility for a site to support a different mapping policy for each remote site.

7.2. LISP for IPv6 Co-existence

LISP encapsulations allows to transport packets using EIDs from a given address family (e.g., IPv6) with packets from other address families (e.g., IPv4). The absence of correlation between the address family of RLOCs and EIDs makes LISP a candidate to allow, e.g., IPv6 to be deployed when all of the core network may not have IPv6 enabled.

For example, two IPv6-only data centers could be interconnected via the legacy IPv4 Internet. If their border routers are LISP capable, sending packets between the data center is done without any form of translation as the native IPv6 packets (in the EID space) will be LISP encapsulated and transmitted over the IPv4 legacy Internet by the mean of IPv4 RLOCs.

7.3. LISP for Virtual Private Networks

It is common to operate several virtual networks over the same physical infrastructure. In such virtual private networks, it is essential to distinguish which virtual network a packet belongs and tags or labels are used for that purpose. When using LISP, the distinction can be made with the Instance ID field. When an ITR encapsulates a packet from a particular virtual network (e.g., known via the VRF or VLAN), it tags the encapsulated packet with the Instance ID corresponding to the virtual network of the packet. When an ETR receives a packet tagged with an Instance ID it uses the Instance ID to determine how to treat the packet.

The main usage of LISP for virtual private networks does not introduce additional requirements on the underlying network, as long as it runs IP.

7.4. LISP for Virtual Machine Mobility in Data Centers

A way to enable seamless virtual machine mobility in data center is to conceive the datacenter backbone as the RLOC space and the subnet where servers are hosted as forming the EID space. A LISP router is placed at the border between the backbone and each subnet. When a virtual machine is moved to another subnet, it can keep (temporarily) the address it had before the move so to continue without a transport layer connection reset. When an xTR detects a source address received on a subnet to be an address not assigned to the subnet, it registers the address to the Mapping System.

To inform the other LISP routers that the machine moved and where, and then to avoid detours via the initial subnetwork, mechanisms such as the Solicit-Map-Request messages are used.

8. Security Considerations

This section describes the security considerations associated to the LISP protocol.

While in a push mapping system, the state necessary to forward packets is learned independently of the traffic itself, with a pull architecture, the system becomes reactive and data-plane events

(e.g., the arrival of a packet for an unknown destination) may trigger control-plane events. This on-demand learning of mappings provides many advantages as discussed above but may also affect the way security is enforced.

Usually, the data-plane is implemented in the fast path of routers to provide high performance forwarding capabilities while the control-plane features are implemented in the slow path to offer high flexibility and a performance gap of several order of magnitude can be observed between the slow and the fast paths. As a consequence, the way data-plane events are notified to the control-plane must be thought carefully so to not overload the slow path and rate limiting should be used as specified in [I-D.ietf-lisp-rfc6830bis] and [I-D.ietf-lisp-rfc6833bis].

Care must also be taken so to not overload the mapping system (i.e., the control plane infrastructure) as the operations to be performed by the mapping system may be more complex than those on the data-plane, for that reason [I-D.ietf-lisp-rfc6830bis] and [I-D.ietf-lisp-rfc6833bis] recommends to rate limit the sending of messages to the mapping system.

To improve resiliency and reduce the overall number of messages exchanged, LISP offers the possibility to leak information, such as reachabilty of locators, directly into data plane packets. In environments that are not fully trusted, like the open Internet, control information gleaned from data-plane packets must not be used or must be verified before using it.

Mappings are the centrepiece of LISP and all precautions must be taken to avoid them to be manipulated or misused by malicious entities. Using trustable Map-Servers that strictly respect [I-D.ietf-lisp-rfc6833bis] and the authentication mechanism proposed by LISP-Sec [I-D.ietf-lisp-sec] reduces the risk of attacks to the mapping integrity. In more critical environments, secure measures may be needed. The way security is implemented for a given mapping system strongly depends on the architecture of the mapping system itself and the threat model assumed for the deployment. Thus, the mapping system security has to be discussed in the relevant documents proposing the mapping system architecture.

As with any other tunneling mechanism, middleboxes on the path between an ITR (or PITR) and an ETR (or PETR) must implement mechanisms to strip the LISP encapsulation to correctly inspect the content of LISP encapsulated packets.

Like other map-and-encap mechanisms, LISP enables triangular routing (i.e., packets of a flow cross different border routers depending on their direction). This means that intermediate boxes may have incomplete view on the traffic they inspect or manipulate. Moreover, LISP-encapsulated packets are routed based on the outer IP address (i.e., the RLOC), and can be delivered to an ETR that is not responsible of the destination EID of the packet or even to a network element that is not an ETR. The mitigation consists in applying appropriate filtering techniques on the network elements that can potentially receive un-expected LISP-encapsulated packets

More details about security implications of LISP are discussed in [RFC7835].

9. IANA Considerations

This memo includes no requests to IANA.

10. Acknowledgements

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

The authors would also like to thank Dino Farinacci, Fabio Maino, Luigi Iannone, Sharon Barkai, Isidoros Kouvelas, Christian Cassar, Florin Coras, Marc Binderberger, Alberto Rodriguez-Natal, Ronald Bonica, Chad Hintz, Robert Raszuk, Joel M. Halpern, Darrel Lewis, David Black.

11. References

11.1. Normative References

[I-D.ietf-lisp-6834bis]
Iannone, L., Saucez, D., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning", Work in Progress, Internet-Draft, draft-ietf-lisp-6834bis-09, 31 August 2021, <https://www.ietf.org/archive/id/draft-ietf-lisp-6834bis-09.txt>.

[I-D.ietf-lisp-rfc6830bis]
Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos-Aparicio, "The Locator/ID Separation Protocol (LISP)", Work in Progress, Internet-Draft, draft-ietf-lisp-rfc6830bis-36, 18 November 2020, <https://www.ietf.org/internet-drafts/draft-ietf-lisp-rfc6830bis-36.txt>.

[I-D.ietf-lisp-rfc6833bis]
Farinacci, D., Maino, F., Fuller, V., and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control-Plane", Work in Progress, Internet-Draft, draft-ietf-lisp-rfc6833bis-30, 18 November 2020, <https://www.ietf.org/internet-drafts/draft-ietf-lisp-rfc6833bis-30.txt>.

[I-D.ietf-lisp-sec]
Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", Work in Progress, Internet-Draft, draft-ietf-lisp-sec-22, 12 January 2021, <https://www.ietf.org/internet-drafts/draft-ietf-lisp-sec-22.txt>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/info/rfc1918>.

[RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992, DOI 10.17487/RFC2992, November 2000, <https://www.rfc-editor.org/info/rfc2992>.

[RFC3232] Reynolds, J., Ed., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, DOI 10.17487/RFC3232, January 2002, <https://www.rfc-editor.org/info/rfc3232>.

[RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, DOI 10.17487/RFC3963, January 2005, <https://www.rfc-editor.org/info/rfc3963>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[RFC4984] Meyer, D., Ed., Zhang, L., Ed., and K. Fall, Ed., "Report from the IAB Workshop on Routing and Addressing", RFC 4984, DOI 10.17487/RFC4984, September 2007, <https://www.rfc-editor.org/info/rfc4984>.

[RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised", RFC 5944, DOI 10.17487/RFC5944, November 2010, <https://www.rfc-editor.org/info/rfc5944>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <https://www.rfc-editor.org/info/rfc6275>.

[RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, DOI 10.17487/RFC6831, January 2013, <https://www.rfc-editor.org/info/rfc6831>.

[RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <https://www.rfc-editor.org/info/rfc6832>.

[RFC6835] Farinacci, D. and D. Meyer, "The Locator/ID Separation Protocol Internet Groper (LIG)", RFC 6835, DOI 10.17487/RFC6835, January 2013, <https://www.rfc-editor.org/info/rfc6835>.

[RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <https://www.rfc-editor.org/info/rfc6836>.

[RFC6837] Lear, E., "NERD: A Not-so-novel Endpoint ID (EID) to Routing Locator (RLOC) Database", RFC 6837, DOI 10.17487/RFC6837, January 2013, <https://www.rfc-editor.org/info/rfc6837>.

[RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <https://www.rfc-editor.org/info/rfc6935>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <https://www.rfc-editor.org/info/rfc6936>.

[RFC7052] Schudel, G., Jain, A., and V. Moreno, "Locator/ID Separation Protocol (LISP) MIB", RFC 7052, DOI 10.17487/RFC7052, October 2013, <https://www.rfc-editor.org/info/rfc7052>.

[RFC7215] Jakab, L., Cabellos-Aparicio, A., Coras, F., Domingo-Pascual, J., and D. Lewis, "Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations", RFC 7215, DOI 10.17487/RFC7215, April 2014, <https://www.rfc-editor.org/info/rfc7215>.

[RFC7835] Saucez, D., Iannone, L., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Threat Analysis", RFC 7835, DOI 10.17487/RFC7835, April 2016, <https://www.rfc-editor.org/info/rfc7835>.

[RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <https://www.rfc-editor.org/info/rfc8060>.

[RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <https://www.rfc-editor.org/info/rfc8111>.

[RFC8378] Moreno, V. and D. Farinacci, "Signal-Free Locator/ID Separation Protocol (LISP) Multicast", RFC 8378, DOI 10.17487/RFC8378, May 2018, <https://www.rfc-editor.org/info/rfc8378>.

11.2. Informative References

[I-D.cheng-lisp-shdht]
Cheng, L. and J. Wang, "LISP Single-Hop DHT Mapping Overlay", Work in Progress, Internet-Draft, draft-cheng-lisp-shdht-04, 15 July 2013, <http://www.ietf.org/internet-drafts/draft-cheng-lisp-shdht-04.txt>.

[I-D.curran-lisp-emacs]
Brim, S., Farinacci, D., Meyer, D., and J. Curran, "EID Mappings Multicast Across Cooperating Systems for LISP", Work in Progress, Internet-Draft, draft-curran-lisp-emacs-00, 9 November 2007, <http://tools.ietf.org/html/draft-curran-lisp-emacs-00>.

[Jakab] Jakab, L., Cabellos, A., Saucez, D., and O. Bonaventure, "LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System, IEEE Journal on Selected Areas in Communications, vol. 28, no. 8, pp. 1332-1343", October 2010.

[Mathy] Mathy, L., Iannone, L., and O. Bonaventure, "LISP-DHT: Towards a DHT to map identifiers onto locators. The ACM ReArch, Re-Architecting the Internet. Madrid (Spain)", December 2008.

[Quoitin] Quoitin, B., Iannone, L., Launois, C., and O. Bonaventure, ""Evaluating the Benefits of the Locator/Identifier Separation" in Proceedings of 2Nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture", 2007.

Appendix A. A Brief History of Location/Identity Separation

The LISP architecture 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 spontaneously formed immediately after that workshop, to work on an idea that came out of informal discussions at the workshop and on various mailing lists. The first Internet-Draft on LISP appeared in January, 2007.

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

LISP transitioned from an IRTF activity to an IETF WG in March 2009, and after numerous revisions, the basic specifications moved to becoming RFCs at the start of 2013 (although work to expand and improve it, and find new uses for it, continues, and undoubtly will for a long time to come). The LISP WG was rechartered in 2018 to continue work on the LISP base protocol and produce standard-track documents.

A.1. Old LISP Models

LISP, as initially conceived, had a number of potential operating modes, named 'models'. Although they are no used anymore, one occasionally sees mention of them, so they are briefly described here.

LISP 1: EIDs all appear in the normal routing and forwarding tables of the network (i.e. they are 'routable');this property is used to 'bootstrap' operation, by using this to load EID->RLOC mappings. Packets were sent with the EID as the destination in the outer wrapper; when an ETR saw such a packet, it would send a Map-Reply to the source ITR, giving the full mapping.

LISP 1.5: Similar to LISP 1, but the routability of EIDs happens on a separate network.

LISP 2: EIDs are not routable; EID->RLOC mappings are available from the DNS.

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

Authors' Addresses

Albert Cabellos
UPC-BarcelonaTech
c/ Jordi Girona 1-3
08034 Barcelona Catalonia
Spain

Email: acabello@ac.upc.edu

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

Email: damien.saucez@inria.fr

LISP Working Group
Internet-Draft
Intended status: Informational
Expires: August 22, 2013

J. N. Chiappa
Yorktown Museum of Asian Art

February 18, 2013

An Architectural Perspective on the LISP Location-Identity Separation System
draft-ietf-lisp-perspective-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.

[NOTE: This is still a somewhat rough draft version; a few sections at the end are just rough frameworks, but almost all the key sections, and all the front part of the document, are here, and in something like reasonably complete form.]

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[[I0: Things inside '[[]]' are editorial comments. For now they show up - when we go to produce the ID, I will make them invisible.]]

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.

[[I1: import text from DPLSCS doc for 'name', 'object', 'namespace', etc - or just refer to it?]]

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.

[[G0: Say more about this?]]

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.

[[G1: What else?]]

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.) [[A0: Is that true? Well, the echo nonce is an exception, I guess - but that really only applies to xTRs, not 'pure' ETRs. And what about fragmented packets which are fragmented on the ITR-ETR stretch?]]

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. [[A1: Where the balance lies between the two (with the advantages and disadvantages), only time will truly tell. - Add, or leave out?]]

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. [[A2: Maybe unnecessary diversion, ditch?]]

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.

[[A3: Is this really true? It’s slightly bizarre, it means that you have to feed MAC addresses into the EID->RLOC mapping system, etc. Joel wrote "many folks who are looking for this document will find the idea of outer-headers being non-IP to be counter-productive. Instead, can we turn the last paragraph ... around, and observe that the use of tunnel technologies allows us to wrap up non-IP packets using Identifiers (such as Ethernet MAC addresses) and deliver them over an IP infrastructure?" To me, doing this is even more counter-intuitive than my original example (next)!]]

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.

[[A4: Say more about this?]]

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.

[[A5: Can anyone explain to my _why_ it’s so horrible? Yeah, I get that it’s probably _better_ to segregate them, all else being equal, and I understand that high-speed switch designs may make it hard in practise to do it, but I don’t get how it’s so inherently evil. If you gotta send a packet anyway, why not two birds it?]]

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.

[[A6: It used to say this: Piggybacking is purely an optization. It is easy to perform some ’back of the envelope’ calculations to show that abandoning it would not cause that great an increase in overhead. but I think I just showed, above, that in some cases it would take a great deal more traffic indeed to get the same kind of response time.]]

[[A7: Should we actually throw some numbers in here?]]

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.

[[N0: Actually, technically speaking, it's not true that EIDs are always only identity. In double-encapsulation cases, the output (RLOC) of one stage of mapping turns into the input (EID) of another stage. We should probably mention that at some point. For now, I have thrown a few 'weasel words' in above.]]

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). [[N1: Check to make sure that's the real definition of EID.]]

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. [[N2: Dike this? It has been quite a hot bone, though, so I'm afraid that it we leave it out, we'll just have round 17 of this debate..]]

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. [[N3: Now, this might be duplicative.]]

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].)

[[N4: A LISP EID which is assigned to a virtual interface, one which is created _solely_ as a place to hold the system's identity, is effectively as pure an EID as one could wish. - Excessive, leave out?]]

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. [[N5: Is this true? Apparently people are working on other things?]]

[[N6: More? Can't think of anything else to say.]]

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. [[N7: Do we really want to point this out, or would it be better to be silent about the IPv4-extending capability?]]

5.4. LCAFs

{{To be written.}}[[N8:]]

--- 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.

[[S0: An illustration from another field may be useful here. The kind of complex and powerful security available in multi-level secure time-sharing systems like Multics appeared at one point to be a relic of another age, not useful in the age of personal machines. Complex and difficult, they appeared to be an expensive dead end. However, when active content first started to appear, it became apparent that allowing active content from the network was allowing almost anyone to share the computer with you - exactly the situation the security of multi-level secure time-sharing systems was designed to handle. - Probably too long / not important enough, delete?]]

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.}} [[S3: WTF is up with DDTSEC anyway?]]

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.

[[F0: Check Intro document to see what text should probably be moved back here with this.]]

10. Optimization

[[O0: May not need this section, most of these topics are covered elsewhere?]]

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

[[O1: does wiretapping ever/still happen? - yes LISP pg. x]]

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.

[[OI0: Should this section be in the Intro document?]]

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.

[[xx: This section will be deleted by the RFC Editor.]]

14. Security Considerations

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

[[yy: Do we need to say anything at all here, beyond something like 'LISP has some potential security issues, see XXX for details'?]]

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-chiappa-lisp-introduction-00 (work in progress), July 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

[[Gl1: I _hate_ it when people put this up front, it really cuts into the flow]]
[[Gl2: Also, most of these can probably be lifted directly from the LISP RFC.]]

- 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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 1, 2016

D. Saucez
INRIA
L. Iannone
Telecom ParisTech
O. Bonaventure
Universite catholique de Louvain

January 29, 2016

LISP Threats Analysis
draft-ietf-lisp-threats-15.txt

Abstract

This document provides a threat analysis of the Locator/Identifier Separation Protocol (LISP).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on August 1, 2016.

Table of Contents

1. Introduction

The Locator/ID Separation Protocol (LISP) is specified in [RFC6830]. This document provides an assessment of the potential security threats for the current LISP specifications if LISP is deployed in the Internet (i.e., a public non-trustable environment).

The document is composed of three main parts: the first defines a general threat model that attackers use to mount attacks. The second part, using this threat model, describes the techniques based on the LISP protocol and LISP architecture that attackers may use to construct attacks. The third part discusses mitigation techniques and general solutions to protect the LISP protocol and architecture from attacks.

This document does not consider all the possible uses of LISP as discussed in [RFC6830] and [RFC7215] and does not cover threats due to specific implementations. The document focuses on LISP unicast, including as well LISP Interworking [RFC6832], LISP Map-Server [RFC6833]), and LISP Map-Versioning [RFC6834]. Additional threats may be discovered in the future while deployment continues. The reader is assumed to be familiar with these documents for understanding the present document.

This document assumes a generic IP service and does not discuss the difference, from a security viewpoint, between using IPv4 or IPv6.

2. Threat model

This document assumes that attackers can be located anywhere in the Internet (either in LISP sites or outside LISP sites) and that attacks can be mounted either by a single attacker or by the collusion of several attackers.

An attacker is a malicious entity that performs the action of attacking a target in a network where LISP is (partially) deployed by leveraging the LISP protocol and/or architecture.

An attack is the action of performing an illegitimate action on a target in a network where LISP is (partially) deployed.

The target of an attack is the entity (i.e., a device connected to the network or a network) that is aimed to undergo the consequences of an attack. Other entities can potentially undergo side effects of an attack, even though they are not directly targeted by the attack. The target of an attack can be selected specifically, i.e., a particular entity, or arbitrarily, i.e., any entity. Finally, an

attacker can aim at attacking one or several targets with a single attack.

Section 2.1 specifies the different modes of operation that attackers can follow to mount attacks and Section 2.2 specifies the different categories of attacks that attackers can build.

2.1. Attacker's Operation Modes

In this document attackers are classified according to their modes of operation, i.e., the temporal and spacial diversity of the attacker. These modes are not mutually exclusive, they can be used by attackers in any combination, and other modes may be discovered in the future. Further, attackers are not at all bound by our classification scheme, so implementers and those deploying will always need to do additional risk analysis for themselves.

2.1.1. On-path vs. Off-path Attackers

On-path attackers, also known as Men-in-the-Middle, are able to intercept and modify packets between legitimate communicating entities. On-path attackers are located either directly on the normal communication path (either by gaining access to a node on the path or by placing themselves directly on the path) or outside the location path but manage to deviate (or gain a copy of) packets sent between the communication entities. On-path attackers hence mount their attacks by modifying packets initially sent legitimately between communication entities.

An attacker is called off-path attacker if it does not have access to packets exchanged during the communication or if there is no communication. In order for their attacks to succeed, off-path attackers must hence generate packets and inject them in the network.

2.1.2. Internal vs. External Attackers

An internal attacker launches its attack from a node located within a legitimate LISP site. Such an attacker is either a legitimate node of the site or it exploits a vulnerability to gain access to a legitimate node in the site. Because of their location, internal attackers are trusted by the site they are in.

On the contrary, an external attacker launches its attacks from the outside of a legitimate LISP site.

2.1.3. Live vs. Time-shifted attackers

A live attacker mounts attacks for which it must remain connected as long as the attack is mounted. In other words, the attacker must remain active for the whole duration of the attack. Consequently, the attack ends as soon as the attacker (or the used attack vector) is neutralized.

On the contrary, a time-shifted attacker mounts attacks that remain active after it disconnects from the Internet.

2.1.4. Control-plane vs. Data-plane attackers

A control-plane attacker mounts its attack by using control-plane functionalities, typically the mapping system.

A data-plane attacker mounts its attack by using data-plane functionalities.

As there is no complete isolation between the control-plane and the data-plane, an attacker can operate in the control-plane (or data-plane) to mount attacks targeting the data-plane (or control-plane) or keep the attacked and targeted planes at the same layer (i.e., from control-plane to control-plane or from data-plane to data-plane).

2.1.5. Cross mode attackers

The attacker modes of operation are not mutually exclusive and hence attackers can combine them to mount attacks.

For example, an attacker can launch an attack using the control-plane directly from within a LISP site to which it is able to get temporary access (i.e., internal + control-plane attacker) to create a vulnerability on its target and later on (i.e., time-shifted + external attacker) mount an attack on the data plane (i.e., data-plane attacker) that leverages the vulnerability.

2.2. Threat categories

Attacks can be classified according to the nine following categories. These categories are not mutually exclusive and can be used by attackers in any combination.

2.2.1. Replay attack

A replay attack happens when an attacker retransmits at a later time, and without modifying it, a packet (or a sequence of packets) that

has already been transmitted.

2.2.2. Packet manipulation

A packet manipulation attack happens when an attacker receives a packet, modifies the packet (i.e., changes some information contained in the packet) and finally transmits the packet to its final destination that can be the initial destination of the packet or a different one.

2.2.3. Packet interception and suppression

In a packet interception and suppression attack, the attacker captures the packet and drops it before it can reach its final destination.

2.2.4. Spoofing

With a spoofing attack, the attacker injects packets in the network pretending to be another node. Spoofing attacks are made by forging source addresses in packets.

It should be noted that with LISP, packet spoofing is similar to spoofing with any other existing tunneling technology currently deployed in the Internet. Generally the term "spoofed packet" indicates a packet containing a source IP address that is not the actual originator of the packet. Hence, since LISP uses encapsulation, the spoofed address could be in the outer header as well as in the inner header, this translates to two types of spoofing.

Inner address spoofing: the attacker uses encapsulation and uses a spoofed source address in the inner packet. In case of data-plane LISP encapsulation, that corresponds to spoofing the source EID (End-point IDentifier) address of the encapsulated packet.

Outer address spoofing: the attacker does not use encapsulation and spoofs the source address of the packet. In case of data-plane LISP encapsulation, that corresponds to spoofing the source RLOC (Routing LOCator) address of the encapsulated packet.

Note that the two types of spoofing are not mutually exclusive, rather all combinations are possible and could be used to perform different kinds of attacks. For example, an attacker outside a LISP site can generate a packet with a forged source IP address (i.e., outer address spoofing) and forward it to a LISP destination. The packet is then eventually encapsulated by a PITR (Proxy Ingress

Tunnel Router) so that once encapsulated the attack corresponds to a inner address spoofing. One can also imagine an attacker forging a packet with encapsulation where both inner and outer source addresses are spoofed.

It is important to note that the combination of inner and outer spoofing makes the identification of the attacker complex as the packet may not contain information that allows to detect the origin of the attack.

2.2.5. Rogue attack

In a rogue attack the attacker manages to appear as a legitimate source of information, without faking its identity (as opposed to a spoofing attacker).

2.2.6. Denial of Service (DoS) attack

A Denial of Service (DoS) attack aims at disrupting a specific targeted service to make it unable to operate properly.

2.2.7. Performance attack

A performance attacks aims at exploiting computational resources (e.g., memory, processor) of a targeted node so as to make it unable to operate properly.

2.2.8. Intrusion attack

In an intrusion attack, the attacker gains remote access to a resource (e.g., a host, a router, or a network) or information that it legitimately should not have access. Intrusion attacks can lead to privacy leakages.

2.2.9. Amplification attack

In an amplification attack, the traffic generated by the target of the attack in response to the attack is larger than the traffic that the attacker must generate.

In some cases, the data-plane can be several orders of magnitude faster than the control-plane at processing packets. This difference can be exploited to overload the control-plane via the data-plane without overloading the data-plane.

2.2.10. Passive Monitoring Attacks

An attacker can use pervasive monitoring, which is a technical attack [RFC7258], targeting information about LISP traffic that may or not be used to mount other type of attacks.

2.2.11. Multi-category attacks

Attacks categories are not mutually exclusive and any combination can be used to perform specific attacks.

For example, one can mount a rogue attack to perform a performance attack starving the memory of an ITR (Ingress Tunnel Router) resulting in a DoS (Denial-of-Service) on the ITR.

3. Attack vectors

This section presents attack techniques that may be used by attackers when leveraging the LISP protocol and/or architecture.

3.1. Gleaning

To reduce the time required to obtain a mapping, the optional gleaning mechanism defined for LISP allows an xTR (Ingress and/or Egress Tunnel Router) to directly learn a mapping from the LISP data encapsulated packets and the Map-Request packets that it receives. LISP encapsulated data packets contain a source RLOC, destination RLOC, source EID and destination EID. When an xTR receives an encapsulated data packet coming from a source EID for which it does not already know a mapping, it may insert the mapping between the source RLOC and the source EID in its EID-to-RLOC Cache. The same technique can be used when an xTR receives a Map-Request as the Map-Request also contains a source EID address and a source RLOC. Once a gleaned entry has been added to the EID-to-RLOC cache, the xTR sends a Map-Request to retrieve the actual mapping for the gleaned EID from the mapping system.

If a packet injected by an off-path attacker and with a spoofed inner address is gleaned by an xTR then the attacker may divert the traffic meant to be delivered to the spoofed EID as long as the gleaned entry is used by the xTR. This attack can be used as part of replay, packet manipulation, packet interception and suppression, or DoS attacks as the packets are sent to the attacker.

If the packet sent by the attacker contains a spoofed outer address instead of a spoofed inner address then it can achieve a DoS or a performance attack as the traffic normally destined to the attacker

will be redirected to the spoofed source RLOC. Such traffic may overload the owner of the spoofed source RLOC, preventing it from operating properly.

If the packet injected uses both inner and outer spoofing, the attacker can achieve a spoofing, a performance, or an amplification attack as traffic normally destined to the spoofed EID address will be sent to the spoofed RLOC address. If the attacked LISP site also generates traffic to the spoofed EID address, such traffic may have a positive amplification factor.

A gleaning attack does not only impact the data-plane but can also have repercussions on the control-plane as a Map-Request is sent after the creation of a gleaned entry. The attacker can then achieve DoS and performance attacks on the control-plane. For example, if an attacker sends a packet for each address of a prefix not yet cached in the EID-to-RLOC cache of an xTR, the xTR will potentially send a Map-Request for each such packet until the mapping is installed which leads to an over-utilisation of the control-plane as each packet generates a control-plane event. In order for this attack to succeed, the attacker may not need to use spoofing. This issue can occur even if gleaning is turned off since whether or not gleaning is used as the ITR may need to send a Map-Request in response to incoming packets whose EID is not currently in the cache.

Gleaning attacks are fundamentally involving a time-shifted mode of operation as the attack may last as long as the gleaned entry is kept by the targeted xTR. RFC 6830 [RFC6830] recommends to store the gleaned entries for only a few seconds which limits the duration of the attack.

Gleaning attacks always involve external data-plane attackers but results in attacks on either the control-plane or data-plane.

Note, the outer spoofed address does not need to be the RLOC of a LISP site, it may be any address.

3.2. Locator Status Bits

When the L bit in the LISP header is set to 1, it indicates that the second 32-bits longword of the LISP header contains the Locator Status Bits. In this field, each bit position reflects the status of one of the RLOCs mapped to the source EID found in the encapsulated packet. The reaction of a LISP xTR that receives such a packet is left as operational choice in [RFC6830].

When an attacker sends a LISP encapsulated packet with an illegitimately crafted LSB to an xTR, it can influence the xTR’s

choice of the locators for the prefix associated to the source EID. In case of an off-path attacker, the attacker must inject a forged packet in the network with a spoofed inner address. An on-path attacker can manipulate the LSB of legitimate packets passing through it and hence does not need to use spoofing. Instead of manipulating the LSB field, an on-path attacker can also obtain the same result of injecting packets with invalid LSB values by replaying packets.

The LSB field can be leveraged to mount a DoS attack by either declaring all RLOCs as unreachable (all LSB set to 0), or by concentrating all the traffic to one RLOC (e.g., all but one LSB set to 0) and hence overloading the RLOC concentrating all the traffic from the xTR, or by forcing packets to be sent to RLOCs that are actually not reachable (e.g., invert LSB values).

The LSB field can also be used to mount a replay, a packet manipulation, or a packet interception and suppression attack. Indeed, if the attacker manages to be on the path between the xTR and one of the RLOCs specified in the mapping, forcing packets to go via that RLOC implies that the attacker will gain access to the packets.

Attacks using the LSB are fundamentally involving a time-shifted mode of operation as the attack may last as long as the reachability information gathered from the LSB is used by the xTR to decide the RLOCs to be used.

3.3. Map-Version

When the Map-Version bit of the LISP header is set to 1, it indicates that the low-order 24 bits of the first 32 bits longword of the LISP header contain a Source and Destination Map-Version. When a LISP xTR receives a LISP encapsulated packet with the Map-Version bit set to 1, the following actions are taken:

- o It compares the Destination Map-Version found in the header with the current version of its own configured EID-to-RLOC mapping, for the destination EID found in the encapsulated packet. If the received Destination Map-Version is smaller (i.e., older) than the current version, the ETR should apply the SMR (Solicit-Map-Request) procedure described in [RFC6830] and send a Map-Request with the SMR bit set.

- o If a mapping exists in the EID-to-RLOC Cache for the source EID, then it compares the Map-Version of that entry with the Source Map-Version found in the header of the packet. If the stored mapping is older (i.e., the Map-Version is smaller) than the source version of the LISP encapsulated packet, the xTR should send a Map-Request for the source EID.

A cross-mode attacker can use the Map-Version bit to mount a DoS attack, an amplification attack, or a spoofing attack. For instance if the mapping cached at the xTR is outdated, the xTR will send a Map-Request to retrieve the new mapping which can yield to a DoS attack (by excess of signalling traffic) or an amplification attack if the data-plane packet sent by the attacker is smaller, or otherwise uses fewer resources, than the control-plane packets sent in response to the attacker’s packet. With a spoofing attack, and if the xTR considers that the spoofed ITR has an outdated mapping, it will send an SMR to the spoofed ITR which can result in performance, amplification, or DoS attack as well.

Map-Version attackers are inherently cross mode as the Map-Version is a method to put control information in the data-plane. Moreover, this vector involves live attackers. Nevertheless, on-path attackers do not have specific advantage over off-path attackers.

3.4. Routing Locator Reachability

The Nonce-Present and Echo-Nonce bits in the LISP header are used to verify the reachability of an xTR. A testing xTR sets the Echo-Nonce and the Nonce-Present bits in LISP data encapsulated packets and include a random nonce in the LISP header of packets. Upon reception of these packets, the tested xTR stores the nonce and echoes it whenever it returns a LISP encapsulated data packets to the testing xTR. The reception of the echoed nonce confirms that the tested xTR is reachable.

An attacker can interfere with the reachability test by sending two different types of packets:

1. LISP data encapsulated packets with the Nonce-Present bit set and a random nonce. Such packets are normally used in response to a reachability test.

2. LISP data encapsulated packets with the Nonce-Present and the Echo-Nonce bits both set. These packets will force the receiving ETR to store the received nonce and echo it in the LISP encapsulated packets that it sends. These packets are normally used as a trigger for a reachability test.

The first type of packets are used to make xTRs think that an other xTR is reachable while it is not. It is hence a way to mount a DoS attack (i.e., the ITR will send its packet to a non-reachable ETR when it should use another one).

The second type of packets could be exploited to attack the nonce-based reachability test. If the attacker sends a continuous flow of

packets that each have a different random nonce, the ETR that receives such packets will continuously change the nonce that it returns to the remote ITR, which can yield to a performance attack. If the remote ITR tries a nonce-reachability test, this test may fail because the ETR may echo an invalid nonce. This hence yields to a DoS attack.

In the case of an on-path attacker, a packet manipulation attack is necessary to mount the attack. To mount such an attack, an off-path attacker must mount an outer address spoofing attack.

If an xTR chooses to periodically check with active probes the liveness of entries in its EID-to-RLOC cache (as described in section 6.3 of [RFC6830]), then this may amplify the attack that caused the insertion of entries being checked.

3.5. Instance ID

LISP allows to carry in its header a 24-bits value called Instance ID and used on the ITR to indicate which local Instance ID has been used for encapsulation, while on the ETR the instance ID decides the forwarding table to use to forward the decapsulated packet in the LISP site.

An attacker (either a control-plane or data-plane attacker) can use the instance ID functionality to mount an intrusion attack.

3.6. Interworking

[RFC6832] defines Proxy-ITR and Proxy-ETR network elements to allow LISP and non-LISP sites to communicate. The Proxy-ITR has functionality similar to the ITR, however, its main purpose is to encapsulate packets arriving from the DFZ (Default-Free Zone) in order to reach LISP sites. A PETR (Proxy Egress Tunnel Router) has functionality similar to the ETR, however, its main purpose is to inject de-encapsulated packets in the DFZ in order to reach non-LISP sites from LISP sites. As a PITR (or PETR) is a particular case of ITR (or ETR), it is subject to similar attacks as ITRs (or ETRs).

As any other system relying on proxies, LISP interworking can be used by attackers to hide their exact origin in the network.

3.7. Map-Request messages

A control-plane off-path attacker can exploit Map-Request messages to mount DoS, performance, or amplification attacks. By sending Map-Request messages at high rate, the attacker can overload nodes involved in the mapping system. For instance sending Map-Requests at

high rate can considerably increase the state maintained in a Map-Resolver or consume CPU cycles on ETRs that have to process the Map-Request packets they receive in their slow path (i.e., performance or DoS attack). When the Map-Reply packet is larger than the Map-Request sent by the attacker, that yields to an amplification attack. The attacker can combine the attack with a spoofing attack to overload the node to which the spoofed address is actually attached.

Note, if the attacker sets the P bit (Probe Bit) in the Map-Request, it will cause legitimately sending the Map-Request directly to the ETR instead of passing through the mapping system.

The SMR bit can be used to mount a variant of these attacks.

For efficiency reasons, Map-Records can be appended to Map-Request messages. When an xTR receives a Map-Request with appended Map-Records, it does the same operations as for the other Map-Request messages and so is subject to the same attacks. However, it also installs in its EID-to-RLOC cache the Map-Records contained in the Map-Request. An attacker can then use this vector to force the installation of mappings in its target xTR. Consequently, the EID-to-RLOC cache of the xTR is polluted by potentially forged mappings allowing the attacker to mount any of the attacks categorized in Section 2.2 (see Section 3.8 for more details). Note, the attacker does not need to forge the mappings present in the Map-Request to achieve a performance or DoS attack. Indeed, if the attacker owns a large enough EID prefix it can de-aggregate it in many small prefixes, each corresponding to another mapping and it installs them in the xTR cache by mean of the Map-Request.

Moreover, attackers can use Map Resolver and/or Map Server network elements to relay its attacks and hide the origin of the attack. Indeed, on the one hand, a Map Resolver is used to dispatch Map-Request to the mapping system and, on the other hand, a Map Server is used to dispatch Map-Requests coming from the mapping system to ETRs that are authoritative for the EID in the Map-Request.

3.8. Map-Reply messages

Most of the security risks associated with Map-Reply messages will depend on the 64 bits nonce that is included in a Map-Request and returned in the Map-Reply. Given the size of the nonce (64 bits), if best current practice is used [RFC4086] and if an ETR does not accept Map-Reply messages with an invalid nonce, the risk of an off-path attack is limited. Nevertheless, the nonce only confirms that the Map-Reply received was sent in response to a Map-Request sent, it does not validate the contents of that Map-Reply.

If an attacker manages to send a valid (i.e., in response to a Map-Request and with the correct nonce) Map-Reply to an ITR, then it can perform any of the attacks categorised in Section 2.2 as it can inject forged mappings directly in the ITR EID-to-RLOC cache. For instance, if the mapping injected to the ITR points to the address of a node controlled by the attacker, it can mount replay, packet manipulation, packet interception and suppression, or DoS attacks, as it will receive every packet destined to a destination lying in the EID prefix of the injected mapping. In addition, the attacker can inject a plethora of mappings in the ITR to mount a performance attack by filling up the EID-to-RLOC cache of the ITR. The attacker can also mount an amplification attack if the ITR at that time is sending a large number of packets to the EIDs matching the injected mapping. In this case, the RLOC address associated to the mapping is the address of the real target of the attacker and so all the traffic of the ITR will be sent to the target which means that with one single packet the attacker may generate very high traffic towards its final target.

If the attacker is a valid ETR in the system, it can mount a rogue attack if it uses prefixes over-claiming. In such a scenario, the attacker ETR replies to a legitimate Map-Request message which it received with a Map-Reply message that contains an EID-Prefix that is larger than the prefix owned by the attacker. For example if the owned prefix is 192.0.2.0/25 but the Map-Reply contains a mapping for 192.0.2.0/24, then the mapping will influence packets destined to other EIDs than the one attacker has authority on. With such technique, the attacker can mount the attacks presented above as it can (partially) control the mappings installed on its target ITR. To force its target ITR to send a Map-Request, nothing prevents the attacker to initiate some communication with the ITR. This method can be used by internal attackers that want to control the mappings installed in their site. To that aim, they simply have to collude with an external attacker ready to over-claim prefixes on behalf of the internal attacker.

Note, when the Map-Reply is in response to a Map-Request sent via the mapping system (i.e., not send directly from the ITR to an ETR), the attacker does not need to use a spoofing attack to achieve its attack as by design the source IP address of a Map-Reply is not known in advance by the ITR.

Map-Request and Map-Reply messages are exposed to any type of attackers, on-path or off-path but also external or internal attackers. Also, even though they are control message, they can be leveraged by data-plane attackers. As the decision of removing mappings is based on the TTL indicated in the mapping, time-shifted attackers can take advantage of injecting forged mappings as well.

3.9. Map-Register messages

Map-Register messages are sent by ETRs to Map Servers to indicate to the mapping system the EID prefixes associated to them. The Map-Register message provides an EID prefix and the list of ETRs that are able to provide Map-Replies for the EID covered by the EID prefix.

As Map-Register messages are protected by an authentication mechanism, only a compromised ETR can register itself to its allocated Map Server.

A compromised ETR can over-claim the prefix it owns in order to influence the route followed by Map-Requests for EIDs outside the scope of its legitimate EID prefix (see Section 3.8 for the list of over-claiming attacks).

A compromised ETR can also de-aggregate its EID prefix in order to register more EID prefixes than necessary to its Map Servers (see Section 3.7 for the impact of de-aggregation of prefixes by an attacker).

Similarly, a compromised Map Server can accept an invalid registration or advertise an invalid EID prefix to the mapping system.

3.10. Map-Notify messages

Map-Notify messages are sent by a Map Server to an ETR to acknowledge the reception and processing of a Map-Register message.

Similarly to the pair Map-Request/Map-Reply, the pair Map-Register/Map-Notify is protected by a nonce making it difficult for an attacker to inject a falsified notification to an ETR to make this ETR believe that the registration succeeded when it has not.

4. Note on Privacy

As reviewed in [RFC6973], universal privacy considerations are difficult to establish as the privacy definitions may vary for different scenarios. As a consequence, this document does not aim at identifying privacy issues related to the LISP protocol but the security threats identified in this document could play a role in privacy threats as defined in section 5 of [RFC6973].

Similar to public deployments of any other control plane protocols, in an Internet deployment, LISP mappings are public and hence provide information about the infrastructure and reachability of LISP sites

(i.e., the addresses of the edge routers). Depending upon deployment details, LISP map replies might or might not provide finer grained and more detailed information than is available with currently deployed routing and control protocols.

5. Threats Mitigation

Most of the above threats can be mitigated with careful deployment and configuration (e.g., filter) and also by applying the general rules of security, e.g. only activating features that are necessary for the deployment and verifying the validity of the information obtained from third parties.

The control-plane is the most critical part of LISP from a security viewpoint and it is worth to notice that the LISP specifications already offer an authentication mechanism for mappings registration ([RFC6833]). This mechanism, combined with LISP-SEC [I-D.ietf-lisp-sec], strongly mitigates threats in non-trustable environments such as the Internet. Moreover, an authentication data field for Map-Request messages and Encapsulated Control messages was allocated [RFC6830]. This field provides a general authentication mechanism technique for the LISP control-plane which future specifications may use while staying backward compatible. The exact technique still has to be designed and defined. To maximally mitigate the threats on the mapping system, authentication must be used, whenever possible, for both Map-Request and Map-Reply messages and for messages exchanged internally among elements of the mapping system, such as specified in [I-D.ietf-lisp-sec] and [I-D.ietf-lisp-ddt].

Systematically applying filters and rate-limitation, as proposed in [RFC6830], will mitigate most of the threats presented in this document. In order to minimise the risk of overloading the control-plane with actions triggered from data-plane events, such actions should be rate limited.

Moreover, all information opportunistically learned (e.g., with LSB or gleaning) should be used with care until they are verified. For example, a reachability change learned with LSB should not be used directly to decide the destination RLOC, but instead should trigger a rate-limited reachability test. Similarly, a gleaned entry should be used only for the flow that triggered the gleaning procedure until the gleaned entry has been verified [Trilogy].

6. Security Considerations

This document provides a threat analysis and proposes mitigation techniques for the Locator/Identifier Separation Protocol.

7. IANA Considerations

This document makes no request to IANA.

8. Acknowledgments

This document builds upon the document of Marcelo Bagnulo ([I-D.bagnulo-lisp-threat]), where the flooding attack and the reference environment was first described.

The authors would like to thank Deborah Brungard, Ronald Bonica, Albert Cabellos, Ross Callon, Noel Chiappa, Florin Coras, Vina Ermagan, Dino Farinacci, Stephen Farrell, Joel Halpern, Emily Hiltzik, Darrel Lewis, Edward Lopez, Fabio Maino, Terry Manderson, and Jeff Wheeler for their comments.

This work has been partially supported by the INFSO-ICT-216372 TRILOGY Project (www.trilogy-project.org).

The work of Luigi Iannone has been partially supported by the ANR-13-INFR-0009 LISP-Lab Project (www.lisp-lab.org) and the EIT KIC ICT-Labs SOFNETS Project.

9. References

9.1. Normative References

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <http://www.rfc-editor.org/info/rfc6830>.

[RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <http://www.rfc-editor.org/info/rfc6832>.

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

DOI 10.17487/RFC6833, January 2013,
<http://www.rfc-editor.org/info/rfc6833>.

[RFC6834] Iannone, L., Saucez, D., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning", RFC 6834, DOI 10.17487/RFC6834, January 2013, <http://www.rfc-editor.org/info/rfc6834>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <http://www.rfc-editor.org/info/rfc6973>.

9.2. Informative References

[I-D.bagnulo-lisp-threat]
Bagnulo, M., "Preliminary LISP Threat Analysis", draft-bagnulo-lisp-threat-01 (work in progress), July 2007.

[I-D.ietf-lisp-ddt]
Fuller, V., Lewis, D., Ermagan, V., and A. Jain, "LISP Delegated Database Tree", draft-ietf-lisp-ddt-03 (work in progress), April 2015.

[I-D.ietf-lisp-sec]
Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-09 (work in progress), October 2015.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[RFC7215] Jakab, L., Cabellos-Aparicio, A., Coras, F., Domingo-Pascual, J., and D. Lewis, "Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations", RFC 7215, DOI 10.17487/RFC7215, April 2014, <http://www.rfc-editor.org/info/rfc7215>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

[Trilogy] Saucez, D. and L. Iannone, "How to mitigate the effect of scans on mapping systems", Trilogy Future Internet Summer

School., 2009.

Appendix A. Document Change Log (to be removed on publication)

- o Version 15 Posted January 2016.
  - * Few changes to address Stephen Farrel comments as part of the IESG Review.
- o Version 14 Posted December 2015.
  - * Editorial changes according to Deborah Brungard’s (Routing AD) review.
- o Version 13 Posted August 2015.
  - * Keepalive version.
- o Version 12 Posted March 2015.
  - * Addressed comments by Ross Callon on the mailing list (http://www.ietf.org/mail-archive/web/lisp/current/msg05829.html).
  - * Addition of a section discussing mitigation techniques for deployments in non-trustable environments.
- o Version 11 Posted December 2014.
  - * Editorial polishing. Clarifications added in few points.
- o Version 10 Posted July 2014.
  - * Document completely remodelled according to the discussions on the mailing list in the thread http://www.ietf.org/mail-archive/web/lisp/current/msg05206.html and to address comments from Ronald Bonica and Ross Callon.
- o Version 09 Posted March 2014.
  - * Updated document according to the review of A. Cabellos.
- o Version 08 Posted October 2013.
  - * Addition of a privacy consideration note.
  - * Editorial changes

- o Version 07 Posted October 2013.
  - * This version is updated according to the thorough review made during October 2013 LISP WG interim meeting.
  - * Brief recommendations put in the security consideration section.
  - * Editorial changes
- o Version 06 Posted October 2013.
  - * Complete restructuration, temporary version to be used at October 2013 interim meeting.
- o Version 05 Posted August 2013.
  - * Removal of severity levels to become a short recommendation to reduce the risk of the discussed threat.
- o Version 04 Posted February 2013.
  - * Clear statement that the document compares threats of public LISP deployments with threats in the current Internet architecture.
  - * Addition of a severity level discussion at the end of each section.
  - * Addressed comments from V. Ermagan and D. Lewis' reviews.
  - * Updated References.
  - * Further editorial polishing.
- o Version 03 Posted October 2012.
  - * Dropped Reference to RFC 2119 notation because it is not actually used in the document.
  - * Deleted future plans section.
  - * Updated References
  - * Deleted/Modified sentences referring to the early status of the LISP WG and documents at the time of writing early versions of the document.

* Further editorial polishing.

* Fixed all ID nits.

o Version 02 Posted September 2012.

* Added a new attack that combines over-claiming and de-aggregation (see Section 3.8).

* Editorial polishing.

o Version 01 Posted February 2012.

* Added discussion on LISP-DDT.

o Version 00 Posted July 2011.

* Added discussion on LISP-MS>.

* Added discussion on Instance ID.

* Editorial polishing of the whole document.

* Added "Change Log" appendix to keep track of main changes.

* Renamed "draft-saucez-lisp-security-03.txt.

Authors' Addresses

Damien Saucez
INRIA
2004 route des Lucioles BP 93
06902 Sophia Antipolis Cedex
France

Email: damien.saucez@inria.fr

Luigi Iannone
Telecom ParisTech
23, Avenue d'Italie, CS 51327
75214 PARIS Cedex 13
France

Email: ggx@gigix.net

Olivier Bonaventure
Universite catholique de Louvain
Place St. Barbe 2
Louvain la Neuve
Belgium

Email: olivier.bonaventure@uclouvain.be

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 25, 2013

F. Maino
V. Ermagan
D. Farinacci
Cisco Systems
M. Smith
Insieme Networks
October 22, 2012

LISP Control Plane for Network Virtualization Overlays
draft-maino-nvo3-lisp-cp-02

Abstract

The purpose of this draft is to analyze the mapping between the Network Virtualization over L3 (NVO3) requirements and the capabilities of the Locator/ID Separation Protocol (LISP) control plane. This information is provided as input to the NVO3 analysis of the suitability of existing IETF protocols to the NVO3 requirements.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The purpose of this draft is to analyze the mapping between the Network Virtualization over L3 (NVO3) [I-D.ietf-nvo3-overlay-problem-statement] requirements and the capabilities of the Locator/ID Separation Protocol (LISP) [I-D.ietf-lisp] control plane. This information is provided as input to the NVO3 analysis of the suitability of existing IETF protocols to the NVO3 requirements.

LISP is a flexible map and encap framework that can be used for overlay network applications, including Data Center Network Virtualization.

The LISP framework provides two main tools for NVO3: (1) a Data Plane that specifies how Endpoint Identifiers (EIDs) are encapsulated in Routing Locators (RLOCs), and (2) a Control Plane that specifies the interfaces to the LISP Mapping System that provides the mapping between EIDs and RLOCs.

This document focuses on the control plane for L2 over L3 LISP encapsulation, where EIDs are associated with MAC addresses. As such the LISP control plane can be used with the data path encapsulations defined in VXLAN [I-D.mahalingam-dutt-dcops-vxlan] and in NVGRE [I-D.sridharan-virtualization-nvgre]. The LISP control plane can, of course, be used with the L2 LISP data path encapsulation defined in [I-D.smith-lisp-layer2].

The LISP control plane provides the Mapping Service for the Network Virtualization Edge (NVE), mapping per-tenant end system identity information on the corresponding location at the NVE. As required by NVO3, LISP supports network virtualization and tenant separation to hide tenant addressing information, tenant-related control plane activity and service contexts from the underlay network.

The LISP control plane is extensible, and can support non-LISP data path encapsulations such as [I-D.sridharan-virtualization-nvgre], or other encapsulations that provide support for network virtualization. [I-D.ietf-lisp-interworking] specifies an open interworking framework to allow LISP to non-LISP sites communication.

Broadcast, unknown unicast, and multicast in the overlay network are supported by either replicated unicast, or core-based multicast as specified in [I-D.ietf-lisp-multicast], [I-D.farinacci-lisp-mr-signaling], and [I-D.farinacci-lisp-te].

Finally, the LISP architecture has a modular design that allows the use of different Mapping Databases, provided that the interface to

the Mapping System remains the same [I-D.ietf-lisp-ms]. This allows for different Mapping Databases that may fit different NVO3 deployments. As an example of the modularity of the LISP Mapping System, a worldwide LISP pilot network is currently using an hierarchical Delegated Database Tree [I-D.fuller-lisp-ddt], after having been operated for years with an overlay BGP mapping infrastructure [I-D.ietf-lisp-alt].

The LISP mapping system supports network virtualization, and a single mapping infrastructure can run multiple instances, either public or private, of the mapping database.

The rest of this document, after giving a quick a LISP overview in Section 3, follows the functional model defined in [I-D.lasserre-nvo3-framework] that provides in Section 4 an overview of the LISP NVO3 reference model, and in Section 5 a description of its functional components. Section 6 contains various considerations on key aspects of LISP NVO3, followed by security considerations in Section 7.

2. Definition of Terms

Flood-and-Learn: the use of dynamic (data plane) learning in VXLAN to discover the location of a given Ethernet/IEEE 802 MAC address in the underlay network.

ARP-Agent Reply: the ARP proxy-reply of an agent (e.g. an ITR) with a MAC address of some other system in response to an ARP request to a target which is not the agent’s IP address

For definition of NVO3 related terms, notably Virtual Network (VN), Virtual Network Identifier (VNI), Network Virtualization Edge (NVE), Data Center (DC), please consult [I-D.lasserre-nvo3-framework].

For definitions of LISP related terms, notably Map-Request, Map-Reply, Ingress Tunnel Router (ITR), Egress Tunnel Router (ETR), Map-Server (MS) and Map-Resolver (MR) please consult the LISP specification [I-D.ietf-lisp].

3. LISP Overview

This section provides a quick overview of L2 LISP, with focus on control plane operations.

The modular and extensible architecture of the LISP control plane allows its use with both L2 or L3 LISP data path encapsulation. In

fact, the LISP control plane can be used even with other L2 overlay data path encapsulations such as VXLAN and NVGRE. When used with VXLAN, the LISP control plane replaces the use of dynamic data plane learning (Flood-and-Learn), as specified in [I-D.mahalingam-dutt-dcops-vxlan] improving scalability and mitigating multicast requirements in the underlay network.

For a detailed LISP overview please refer to [I-D.ietf-lisp] and related drafts.

To exemplify LISP operations let’s consider two data centers (LISP sites) A and B that provide L2 network virtualization services to a number of tenant end systems, as depicted in Figure 1. The Endpoint Identifiers (EIDs) are encoded according to [I-D.ietf-lisp-lcaf] as an <IID,MAC> tuple that contains the Instance ID, or Virtual Network Identifier (VNI), and the endpoint Ethernet/IEEE 802 MAC address.

The data centers are connected via a L3 underlay network, hence the Routing Locators (RLOCs) are IP addresses (either IPv4 or IPv6) encoded according to [I-D.ietf-lisp-lcaf].

In LISP the network virtualization edge function is performed by Ingress Tunnel Routers (ITRs) that are responsible for encapsulating the LISP ingress traffic, and Egress Tunnel Routers (ETRs) that are responsible for decapsulating the LISP egress traffic. ETRs are also responsible to register the EID-to-RLOC mapping for a given LISP site in the LISP mapping database system. ITRs and ETRs are collectively referred as xTRs.

The EID-to-RLOC mapping is stored in the LISP mapping database, a distributed mapping infrastructure accessible via Map Servers (MS) and Map Resolvers (MR). [I-D.fuller-lisp-ddt] is an example of a mapping database used in many LISP deployments. Another example of of mapping database is [I-D.ietf-lisp-alt].

For small deployments the mapping infrastructure can be very minimal, in some cases even a single system running as MS/MR.

```
                                  ,---------.
                                 ,           ,
                                (Mapping System )
                                 '.         ,'
                                   '-+------+'
                                +--+--+   +-+---+
                                |MS/MR|   |MS/MR|
                                +-+---+   +-----+
                                    |        |
                                .--.--. .--. ..
                               (    '           '.--.
                             .-.'        L3          '
                            (         Underlay       )
                             (                      ,-'
                              ._.'--'._.'.-._.'.-._)
                     RLOC=IP_A //                  \\ RLOC=IP_B
                            +---+--+              +-+--+--+
                      .--.-.|xTR A |'.-.         .| xTR B |.-.
                     (      +---+--+    )       ( +-+--+--+   )
                    (                __.       (              '.
                  ..'  LISP Site A  )         .'   LISP Site B  )
                 (             .'-'          (             .'-'
                   '--'._.'.    )\            '--'._.'.    )\
                    /       '--'  \            /       '--'  \
                 '--------'    '--------'   '--------'   '--------'
                 :  End   :    :  End   :   :  End   :   :  End   :
                 : Device :    : Device :   : Device :   : Device :
                 '--------'    '--------'   '--------'   '--------'
                    EID=          EID=         EID=         EID=
               <IID1,MAC_W>  <IID2,MAC_X> <IID1,MAC_Y> <IID1,MAC_Z>
```

Figure 1: Example of L2 NVO3 Services

3.1. LISP Site Configuration

In each LISP site the xTRs are configured with an IP address (the site RLOCs) per each interface facing the underlay network.

Similarly the MS/MR are assigned an IP address in the RLOC space.

The configuration of the xTRs includes the RLOCs of the MS/MR and a shared secret that is optionally used to secure the communication between xTRs and MS/MR.

To provide support for multi-tenancy multiple instances of the mapping database are identified by a LISP Instance ID (IID), that is equivalent to the 24-bit VXLAN Network Identifier (VNI) or Tenant Network Identifier (TNI) that identifies tenants in

[I-D.mahalingam-dutt-dcops-vxlan].

3.2. End System Provisioning

We assume that a provisioning framework will be responsible for provisioning end systems (e.g. VMs) in each data center. The provisioning configures each end system with an Ethernet/IEEE 802 MAC address and provision the network with other end system specific attributes such as IP addresses, and VLAN information. LISP does not introduce new addressing requirements for end systems.

The provisioning infrastructure is also responsible to provide a network attach function, that notifies the network virtualization edge (the LISP site ETR) that the end system is attached to a given virtual network (identified by its VNI/IID) and that the end system is identified, within that virtual network, by a given Ethernet/IEEE 802 MAC address.

3.3. End System Registration

Upon notification of end system network attach, that includes the EID=<IID,MAC> tuple that identifies that end system, the ETR sends a LISP Map-Register to the Mapping System. The Map-Register includes the EID and RLOCs of the LISP site. The EID-to-RLOC mapping is now available, via the Mapping System Infrastructure, to other LISP sites that are hosting end systems that belong to the same tenant.

For more details on end system registration see [I-D.ietf-lisp-ms].

3.4. Packet Flow and Control Plane Operations

This section provides an example of the unicast packet flow and the control plane operations when in the topology shown in Figure 1 end system W, in LISP site A, wants to communicate to end system Y in LISP site B. We'll assume that W knows Y's EID MAC address (e.g. learned via ARP).

- o W sends an Ethernet/IEEE 802 MAC frame with destination EID=<IID1,MAC_Y> and source EID=<IID1,MAC_W>.

- o ITR A does a lookup in its local map-cache for the destination EID=<IID1, MAC_Y>. Since this is the first packet sent to MAC_Y, the map-cache is a miss, and the ITR sends a Map-request to the mapping database system looking up the EID=<IID1,MAC_Y>.

- o The mapping systems forwards the Map-Request to ETR B, that is aware of the EID-to-RLOC mapping for <IID1,MAC_Y>. Alternatively, depending on the mapping system configuration, a Map-Server which

is part of the mapping database system may send a Map-Reply directly to ITR A.

- o ETR B sends a Map-Reply to ITR A that includes the EID-to-RLOC mapping: EID=<IID1,MAC_Y> -> RLOC=IP_B, where IP_B is the locator of ETR B, hence the locator of LISP site B. In order to facilitate interoperability, the Map-Reply may also include attributes such as the data plane encapsulations supported by the ETR.

- o ITR A populates the local map-cache with the EID to RLOC mapping, and either L2 LISP, VXLAN, or NVGRE encapsulates all subsequent packets with a destination EID=<IID1,MAC_Y> with a destination RLOC=IP_B.

It should be noted how the LISP mapping system replaces the use of Flood-and-Learn based on multicast distribution trees instantiated in the underlay network (required by VXLAN’s dynamic data plane learning), with a unicast control plane and a cache mechanism that "pulls" on-demand the EID-to-RLOC mapping from the LISP mapping database. This improves scalability, and simplifies the configuration of the underlay network.

3.4.1. Supporting ARP Resolution with LISP Mapping System

A large majority of data center applications are IP based, and in those use cases end systems are provisioned with IP addresses as well as MAC addresses.

In this case, to eliminate the flooding of ARP traffic and further reduce the need for multicast in the underlay network, the LISP mapping system is used to support ARP resolution at the ITR. We assume that as shown in Figure 2: (1) end system W has an IP address IP_W, and end system Y has an IP address IP_Y, (2) end system W knows Y’s IP address (e.g. via DNS lookup). We also assume that during registration Y has registered both its MAC address and its IP address as EID. End system Y is then identified by the EID = <IID1,IP_Y,MAC_Y>.

```
                                  ,---------.
                                 ,           .
                                (Mapping System )
                                 '.          ,'
                                   '-+------+'
                                +--+--+   +-+---+
                                |MS/MR|   |MS/MR|
                                +-+---+   +-----+
                                    |        |
                                .--.--. .--. ..
                               (    '           '.--.
                             .-.'        L3          '
                            (         Underlay       )
                             (                      '-'
                              ._.'--'._.'.-._.'.-._)
                     RLOC=IP_A //                  \\ RLOC=IP_B
                            +---+--+              +-+--+--+
                      .--.-.|xTR A |'.-.         .| xTR B |.-.
                     (      +---+--+    )       ( +-+--+--+   )
                    (                __.       (              '.
                  ..'  LISP Site A  )         .'   LISP Site B  )
                 (             .'-'          (             .'-'
                   '--'._.'.    )\            '--'._.'.    )\
                    /       '--'  \            /       '--'  \
                ,--------'    '--------'   ,--------'   '--------'
                :  End   :    :  End   :   :  End   :   :  End   :
                : Device :    : Device :   : Device :   : Device :
                '--------'    '--------'   '--------'   '--------'
                   EID=          EID=         EID=         EID=
                <IID1,IP_W,  <IID2,IP_X,   <IID1,IP_Y,  <IID1,IP_Z,
                  MAC_W>        MAC_X>        MAC_Y>       MAC_Z>
```

Figure 2: Example of L3 NVO3 Services

The packet flow and control plane operation are as follows:

- o End system W sends a broadcast ARP message to discover the MAC address of end system Y. The message contains IP_Y in the ARP message payload.

- o ITR A, acting as a L2 switch, will receive the ARP message, but rather than flooding it on the overlay network sends a Map-Request to the mapping database system for EID = <IID1,IP_Y,*>.

- o The Map-Request is routed by the mapping system infrastructure to ETR B, that will send a Map-Reply back to ITR A containing the mapping EID=<IID1,IP_Y,MAC_Y> -> RLOC=IP_B, (the locator of ETR B). Alternatively, depending on the mapping system configuration,

a Map-Server in the mapping system may send directly a Map-Reply to ITR A.

- o ITR A populates the map-cache with the received entry, and sends an ARP-Agent Reply to W that includes MAC_Y and IP_Y.

- o End system W learns MAC_Y from the ARP message and can now send a packet to end system Y by including MAC_Y, and IP_Y, as destination addresses.

- o ITR A will then process the packet as specified in Section 3.4.

This example shows how LISP, by replacing dynamic data plane learning (Flood-and-Learn) largely reduces the need for multicast in the underlay network, that is needed only when broadcast, unknown unicast or multicast are required by the applications in the overlay. In practice, the LISP mapping system, constrains ARP within the boundaries of a link-local protocol. This simplifies the configuration of the underlay network and removes the significant scalability limitation imposed by VXLAN Flood-and-Learn.

It’s important to note that the use of the LISP mapping system, by pulling the EID-to-RLOC mapping on demand, also improves end system mobility across data centers.

3.5. End System Mobility

This section shows how the LISP control plane deals with mobility when end systems are migrated from one Data Center to another. We’ll assume that a signaling protocol, as described in [I-D.kompella-nvo3-server2nve], signals to the NVE operations such as creating/terminating/migrating an end system. The signaling protocol consists of three basic messages: "associate", "dissociate", and "pre-associate".

Let’s consider the scenario shown in Figure 3 where end system W moves from data center A to data center B.

```
                                   ,---------.
                                  ,           ,
                                 (Mapping System )
                                  '.         ,'
                                    '-+------+'
                                 +--+--+   +-+---+
                                 |MS/MR|   |MS/MR|
                                 +-+---+   +-----+
                                     |        |
                                 .--.--. .--. ..
                               (    '           '.--.
                            .-.'        L3          '
                           (         Underlay       )
                            (                      '-'
                             ._.'--'._.'.-._.'.-._)
                   RLOC=IP_A //                  \\ RLOC=IP_B
                          +---+--+              +-+--+--+
                     .--.-.|xTR A |'.-.         .| xTR B |.-.
                    (      +---+--+    )       ( +-+--+--+   )
                   (                __.       (              '.
                ..'  LISP Site A  )         .'   LISP Site B  )
               (             .'-'          (             .'-'
                '--'._.'.    )\             '--'._.'.    )\
                 /       '--'  \             /       '--'  \
              '--------'   '--------'     '--------'   '--------'
              :  End   :   :  End   : ==> :  End   :   :  End   :
              : Device :   : Device : ==> : Device :   : Device :
              '--------'   '--------'     '--------'   '--------'
                 EID=            EID=<IID1,MAC_W>         EID=
            <IID2,MAC_X>                              <IID1,MAC_Z>
```

Figure 3: End System Mobility

As a result of the end system registration, described in Section 3.3, the Mapping System contains the EID-to-RLOC mapping for end system W that associates EID=<IID1, MAC_W> with the RLOC(s) associated with LISP site A (IP_A).

The process of migrating end system W from data center A to data center B is initiated.

ETR B receives a pre-associate message that includes EID=<IID1, MAC_W>. ETR B sends a Map-Register to the mapping system registering RLOC=IP_B as an additional locator for end system W with priority set to 255. This means that the RLOC MUST NOT be used for unicast forwarding, but the mapping system is now aware of the new location.

During the migration process of end system W, ETR A receives a

dissociate message, and sends a Map-Register with Record TTL=0 to signal the mapping system that end system W is no longer reachable at RLOC=IP_A. xTR A will also add an entry in its forwarding table that marks EID=<IID1, MAC_W> as non-local. When end system W has completed its migration, ETR B receives an associate message for end system W, and sends a Map-Register to the mapping system setting a non-255 priority for RLOC=IP_B. Now the mapping system is updated with the new EID-to-RLOC mapping for end system W with the desired priority.

The remote ITRs that were corresponding with end system W during the migration will keep sending packets to ETR A. ETR A will keep forwarding locally those packets until it receives a dissociate message, and the entry in the forwarding table associated with EID=<IID1, MAC_W> is marked as non-local. Subsequent packets arriving at ETR A from a remore ITR, and destined to end system W will hit the entry in the forwarding table that will generate an exception, and will generate a Solicit-Map-Request (SMR) message that is returned to the remote ITR. Upon receiving the SMR the remote ITR will invalidate its local map-cache entry for EID=<IID1, MAC_W> and send a new Map-Request for that EID. The Map-Request will generate a Map-Reply that includes the new EID-to-RLOC mapping for end system W with RLOC=IP_B. Similarly, unencapsulated packets arriving at ITR A from local end systems and destined to end system W will hit the entry in the forwarding table marked as non-local, and will generate an exception that by sending a Map-Request for EID=<IID1, MAC_W> will populate the map-cache of ITR A with an EID-to-RLOC mapping for end system W with RLOC=IP_B.

3.6. L3 LISP

The two examples above shows how the LISP control plane can be used in combination with either L2 LISP, VXLAN, or NVGRE encapsulation to provide L2 network virtualization services across data centers.

There is a trend, led by Massive Scalable Data Centers, that is accelerating the adoption of L3 network services in the data center, to preserve the many benefits introduced by L3 (scalability, multi-homing, ...).

LISP, as defined in [I-D.ietf-lisp], provides L3 network virtualization services over an L3 underlay network that, as an alternative to L2 overlay solutions, matches the requirements for DC Network Virtualization. L2 overlay solutions are necessary for Data Centers that rely on non IPv4/IPv6 protocols, but when IP is pervasive L3 LISP provides a better and more scalable overlay.

4. Reference Model

4.1. Generic LISP NVE Reference Model

In the generic NVO3 reference model described in [I-D.lasserre-nvo3-framework], a Tenant End System attaches to a Network Virtualization Edge (NVE) either directly or via a switched network.

In a LISP NVO3 network the Tenant End Systems are part of a LISP site, and the NVE function is provided by LISP xTRs. xTRs provide for tenant separation, perform the encap/decap function, and interface with the LISP Mapping System that maps tenant addressing information (in the EID name space) on the underlay L3 infrastructure (in the RLOC name space).

Tenant segregation across LISP sites is provided by the LISP Instance ID (IID), a 24-bit value that is used by the LISP routers as the Virtual Network Identifier (VNI). Virtualization and Segmentation with LISP is addressed in section 5.5 of [I-D.ietf-lisp].

```
              ...............          ,---------.          ...............
              .  +--------+ .        ,'           '.        . +--------+ .
              .  | Tenant | .       (Mapping System )       . | Tenant | .
              .  |  End   +---+      '.           ,'      +---|  End   | .
              .  | System | . |        '-+------+'        | . | System | .
              .  +--------+ . |    ..................    | . +--------+ .
              .             . |  +-+--+           +--+-+  | .            .
              .             . |  | NV |           | NV |  | .            .
              .  LISP Site  . +--|Edge|           |Edge|--+ . LISP Site  .
              .             .    +-+--+           +--+-+    .            .
              .             .   / (xTR) L3 Overlay (xTR)\   .            .
              .  +--------+ .  /   .     Network     .   \  .  +--------+.
              .  | Tenant +---+    .                 .    +----| Tenant |.
              .  |  End   | .      .    (xTR)        .       . |  End   |.
              .  | System | .      .    +----+       .       . | System |.
              .  +--------+ .      .....| NV |........       . +--------+.
              ...............           |Edge|               .............
                                        +----+
                                 .........|............
                                 .        |LISP Site  .
                                 .        |           .
                                 .     +--------+     .
                                 .     | Tenant |     .
                                 .     |  End   |     .
                                 .     | System |     .
                                 .     +--------+     .
```

.....

Generic reference model for DC NVO3 LISP infrastructure

4.2. LISP NVE Service Types

LISP can be used to support both L2 NVE and L3 NVE service types thanks to the flexibility provided by the LISP Canonical Address Format [I-D.ietf-lisp-lcaf], that allows for EIDs to be encoded either as MAC addresses or IP addresses.

4.2.1. LISP L2 NVE Services

The frame format defined in [I-D.mahalingam-dutt-dcops-vxlan], has a header compatible with the LISP data path encapsulation header, when MAC addresses are used as EIDs, as described in section 4.12.2 of [I-D.ietf-lisp-lcaf].

The LISP control plane is extensible, and can support non-LISP data path encapsulations such as NVGRE [I-D.sridharan-virtualization-nvgre], or other encapsulations that provide support for network virtualization.

4.2.2. LISP L3 NVE Services

LISP is defined as a virtualized IP routing and forwarding service in [I-D.ietf-lisp], and as such can be used to provide L3 NVE services.

5. Functional Components

This section describes the functional components of a LISP NVE as defined in Section 3 of [I-D.lasserre-nvo3-framework].

5.1. Generic Service Virtualization Components

The generic reference model for NVE is depicted in Section 3.1 of [I-D.lasserre-nvo3-framework].

```
                          +------- L3 Network ------+
                          |                         |
                          |       Tunnel Overlay    |
            +------------+---------+       +---------+------------+
            | +----------+-------+ |       | +---------+--------+ |
            | |  Overlay Module  | |       | |  Overlay Module  | |
            | +---------+--------+ |       | +---------+--------+ |
            |           |VN context|       | VN context|          |
            |           |          |       |           |          |
            |  +--------+-------+  |       |  +--------+-------+  |
            |  |     VNI        |  |       |  |       VNI      |  |
       NVE1 |  +-+------------+-+  |       |  +-+-----------+--+  | NVE2
            |    |   VAPs     |    |       |    |    VAPs   |     |
            +----+------------+----+       +----+------------+----+
                 |            |                 |            |
         -------+------------+-----------------+------------+-------
                 |            |     Tenant      |            |
                 |            |   Service IF    |            |
                Tenant End Systems            Tenant End Systems
```

Generic reference model for NV Edge

5.1.1. Virtual Attachment Points (VAPs)

In a LISP NVE, Tunnel Routers (xTRs) implement the NVE functionality on ToRs or Virtual Switches. Tenant End Systems attach to the Virtual Access Points (VAPs) provided by the xTRs (either a physical port or a virtual interface).

5.1.2. Overlay Modules and Tenant ID

The xTR also implements the function of NVE Overlay Module, by mapping the addressing information (EIDs) of the tenant packet on the appropriate locations (RLOCs) in the underlay network. The Tenant Network Identifier (TNI) is encoded in the encapsulated packet (either in the 24-bit IID field of the LISP header for L2/L3 LISP encapsulation, or in the 24-bit VXLAN Network Identifier field for VXLAN encapsulation, or in the 24-bit NVGRE Tenant Network Identifier field of NVGRE). In a LISP NVE globally unique (per administrative domain) TNIs are used to identify the Tenant instances.

The mapping of the tenant packet address onto the underlay network location is "pulled" on-demand from the mapping system, and cached at the NVE in a per-TNI map-cache.

5.1.3. Tenant Instance

Tenants are mapped on LISP Instance IDs (IIDs), and the xTR keeps an instance of the LISP control protocol per each IID. The ETR is responsible to register the Tenant End System to the LISP mapping system, via the Map-Register service provided by LISP Map-Servers (MS). The Map-Register includes the IID that is used to identify the tenant.

5.1.4. Tunnel Overlays and Encapsulation Options

The LISP control protocol, as defined today, provides support for L2 LISP and VXLAN L2 over L3 encapsulation, and LISP L3 over L3 encapsulation.

We believe that the LISP control Protocol can be easily extended to support different IP tunneling options (such as NVGRE).

5.1.5. Control Plane Components

5.1.5.1. Auto-provisioning/Service Discovery

The LISP framework does not include mechanisms to provision the local NVE with the appropriate Tenant Instance for each Tenant End Systems. Other protocols, such as VDP (in IEEE P802.1Qbg), should be used to implement a network attach/detach function.

The LISP control plane can take advantage of such a network attach/detach function to trigger the registration of a Tenant End System to the Mapping System. This is particularly helpful to handle mobility across DC of the Tenant End System.

It is possible to extend the LISP control protocol to advertise the tenant service instance (tenant and service type provided) to other NVEs, and facilitate interoperability between NVEs that are using different service types.

5.1.5.2. Address Advertisement and Tunnel mapping

As traffic reaches an ingress NVE, the corresponding ITR uses the LISP Map-Request/Reply service to determine the location of the destination End System.

The LISP mapping system combines the distribution of address advertisement and (stateless) tunneling provisioning.

When EIDs are mapped on both IP addresses and MACs, the need to flood ARP messages at the NVE is eliminated resolving the issues with

explosive ARP handling.

5.1.5.3. Tunnel Management

LISP defines several mechanisms for determining RLOC reachability, including Locator Status Bits, "nonce echoing", and RLOC probing. Please see Sections 5.3 and 6.3 of [I-D.ietf-lisp].

6. Key Aspects of Overlay

6.1. Overlay Issues to Consider

6.1.1. Data Plane vs. Control Plane Driven

The use of LISP control plane minimizes the need for multicast in the underlay network overcoming the scalability limitations of VXLAN dynamic data plane learning (Flood-and-Learn).

Multicast or ingress replication in the underlay network are still required, as specified in [I-D.ietf-lisp-multicast], [I-D.farinacci-lisp-mr-signaling], and [I-D.farinacci-lisp-te], to support broadcast, unknown, and multicast traffic in the overlay, but multicast in the underlay is no longer required (at least for IP traffic) for unicast overlay services.

6.1.2. Data Plane and Control Plane Separation

LISP introduces a clear separation between data plane and control plane functions. LISP modular design allows for different mapping databases, to achieve different scalability goals and to meet requirements of different deployments.

6.1.3. Handling Broadcast, Unknown Unicast and Multicast (BUM) Traffic

Packet replication in the underlay network to support broadcast, unknown unicast and multicast overlay services can be done by:

- o Ingress replication

- o Use of underlay multicast trees

[I-D.ietf-lisp-multicast] specifies how to map a multicast flow in the EID space during distribution tree setup and packet delivery in the underlay network. LISP-multicast doesn’t require packet format changes in multicast routing protocols, and doesn’t impose changes in the internal operation of multicast in a LISP site. The only operational changes are required in PIM-ASM [RFC4601], MSDP

[RFC3618], and PIM-SSM [RFC4607].

7. Security Considerations

[I-D.ietf-lisp-sec] defines a set of security mechanisms that provide origin authentication, integrity and anti-replay protection to LISP's EID-to-RLOC mapping data conveyed via mapping lookup process. LISP-SEC also enables verification of authorization on EID-prefix claims in Map-Reply messages.

Additional security mechanisms to protect the LISP Map-Register messages are defined in [I-D.ietf-lisp-ms].

The security of the Mapping System Infrastructure depends on the particular mapping database used. The [I-D.fuller-lisp-ddt] specification, as an example, defines a public-key based mechanism that provides origin authentication and integrity protection to the LISP DDT protocol.

8. IANA Considerations

This document has no IANA implications

9. Acknowledgements

The authors want to thank Victor Moreno and Paul Quinn for the early review, insightful comments and suggestions.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3618] Fenner, B. and D. Meyer, "Multicast Source Discovery Protocol (MSDP)", RFC 3618, October 2003.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

10.2. Informative References

[I-D.farinacci-lisp-mr-signaling]
Farinacci, D. and M. Napierala, "LISP Control-Plane Multicast Signaling", draft-farinacci-lisp-mr-signaling-00 (work in progress), July 2012.

[I-D.farinacci-lisp-te]
Farinacci, D., Lahiri, P., and M. Kowal, "LISP Traffic Engineering Use-Cases", draft-farinacci-lisp-te-00 (work in progress), March 2012.

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

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

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

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

[I-D.ietf-lisp-lcaf]
Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", draft-ietf-lisp-lcaf-00 (work in progress), August 2012.

[I-D.ietf-lisp-ms]
Fuller, V. and D. Farinacci, "LISP Map Server Interface", draft-ietf-lisp-ms-14 (work in progress), December 2011.

[I-D.ietf-lisp-multicast]
Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "LISP for Multicast Environments", draft-ietf-lisp-multicast-14 (work in progress), February 2012.

[I-D.ietf-lisp-sec]

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

[I-D.ietf-nvo3-overlay-problem-statement]
Narten, T., Black, D., Dutt, D., Fang, L., Gray, E., Kreeger, L., Napierala, M., and M. Sridhavan, "Problem Statement: Overlays for Network Virtualization", draft-ietf-nvo3-overlay-problem-statement-00 (work in progress), September 2012.

[I-D.kompella-nvo3-server2nve]
Kompella, K., Rekhter, Y., and T. Morin, "Using Signaling to Simplify Network Virtualization Provisioning", draft-kompella-nvo3-server2nve-00 (work in progress), July 2012.

[I-D.lasserre-nvo3-framework]
Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for DC Network Virtualization", draft-lasserre-nvo3-framework-03 (work in progress), July 2012.

[I-D.mahalingam-dutt-dcops-vxlan]
Sridhar, T., Bursell, M., Kreeger, L., Dutt, D., Wright, C., Mahalingam, M., Duda, K., and P. Agarwal, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", draft-mahalingam-dutt-dcops-vxlan-01 (work in progress), February 2012.

[I-D.smith-lisp-layer2]
Smith, M. and D. Dutt, "Layer 2 (L2) LISP Encapsulation Format", draft-smith-lisp-layer2-00 (work in progress), March 2011.

[I-D.sridharan-virtualization-nvgre]
Sridhavan, M., Duda, K., Ganga, I., Greenberg, A., Lin, G., Pearson, M., Thaler, P., Tumuluri, C., and Y. Wang, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-00 (work in progress), September 2011.

Authors' Addresses

Fabio Maino
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: fmaino@cisco.com

Vina Ermagan
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: vermagan@cisco.com

Dino Farinacci
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: dino@cisco.com

Michael Smith
Insieme Networks
California
USA

Email: michsmit@insiemenetworks.com