Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: November 2, 2015

D. Farinacci
lispers.net
B. Weis
cisco Systems

May 1, 2015

LISP Data-Plane Confidentiality
draft-ietf-lisp-crypto-01

Abstract

This document describes a mechanism for encrypting LISP encapsulated traffic. The design describes how key exchange is achieved using existing LISP control-plane mechanisms as well as how to secure the LISP data-plane from third-party surveillance attacks.

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 November 2, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Overview 3
3. Diffie-Hellman Key Exchange 3
4. Encoding and Transmitting Key Material 4
5. Shared Keys used for the Data-Plane 6
6. Data-Plane Operation 8
7. Procedures for Encryption and Decryption 10
8. Dynamic Rekeying 11
9. Future Work 12
10. Security Considerations 12
10.1. SAAG Support 12
10.2. LISP-Crypto Security Threats 12
11. IANA Considerations 13
12. References 13
12.1. Normative References 13
12.2. Informative References 14
Appendix A. Acknowledgments 14
Appendix B. Document Change Log 14
B.1. Changes to draft-ietf-lisp-crypto-01.txt 15
B.2. Changes to draft-ietf-lisp-crypto-00.txt 15
B.3. Changes to draft-farinacci-lisp-crypto-01.txt 15
B.4. Changes to draft-farinacci-lisp-crypto-00.txt 16
Authors' Addresses 16

1. Introduction

The Locator/ID Separation Protocol [RFC6830] defines a set of functions for routers to exchange information used to map from non-routable Endpoint Identifiers (EIDs) to routable Routing Locators (RLOCs). LISP ITRs and PITRs encapsulate packets to ETRs and RTRs. Packets that arrive at the ITR or PITR are typically not modified. Which means no protection or privacy of the data is added. If the source host encrypts the data stream then the encapsulated packets can be encrypted but would be redundant. However, when plaintext packets are sent by hosts, this design can encrypt the user payload to maintain privacy on the path between the encapsulator (the ITR or PITR) to a decapsulator (ETR or RTR). The encrypted payload is unidirectional. However, return traffic uses the same procedures but with different key values by the same xTRs or potentially different xTRs when the paths between LISP sites are asymmetric.

This draft has the following requirements for the solution space:

- o Do not require a separate Public Key Infrastructure (PKI) that is out of scope of the LISP control-plane architecture.

- o The budget for key exchange MUST be one round-trip time. That is, only a two packet exchange can occur.

- o Use symmetric keying so faster cryptography can be performed in the LISP data plane.

- o Avoid a third-party trust anchor if possible.

- o Provide for rekeying when secret keys are compromised.

- o Support Authenticated Encryption with packet integrity checks.

- o Support multiple cipher suites so new crypto algorithms can be easily introduced.

2. Overview

The approach proposed in this draft is to NOT rely on the LISP mapping system (or any other key infrastructure system) to store security keys. This will provide for a simpler and more secure mechanism. Secret shared keys will be negotiated between the ITR and the ETR in Map-Request and Map-Reply messages. Therefore, when an ITR needs to obtain the RLOC of an ETR, it will get security material to compute a shared secret with the ETR.

The ITR can compute 3 shared-secrets per ETR the ITR is encapsulating to. And when the ITR encrypts a packet before encapsulation, it will identify the key it used for the crypto calculation so the ETR knows which key to use for decrypting the packet after decapsulation. By using key-ids in the LISP header, we can also get real-time rekeying functionality.

3. Diffie-Hellman Key Exchange

LISP will use a Diffie-Hellman [RFC2631] key exchange sequence and computation for computing a shared secret. The Diffie-Hellman parameters will be passed via Cipher Suite code-points in Map-Request and Map-Reply messages.

Here is a brief description how Diff-Hellman works:

```
+----------------------------+---------+----------------------------+
|              ITR           |         |           ETR              |
+------+--------+------------+---------+------------+---------------+
|Secret| Public | Calculates |  Sends  | Calculates | Public |Secret|
+------|--------|------------|---------|------------|--------|------+
|  i   |  p,g   |            | p,g --> |            |        |  e   |
+------|--------|------------|---------|------------|--------|------+
|  i   | p,g,I  |g^i mod p=I |  I -->  |            | p,g,I  |  e   |
+------|--------|------------|---------|------------|--------|------+
|  i   | p,g,I  |            |  <-- E  |g^e mod p=E |  p,g   |  e   |
+------|--------|------------|---------|------------|--------|------+
| i,s  |p,g,I,E |E^i mod p=s |         |I^e mod p=s |p,g,I,E | e,s  |
+------|--------|------------|---------|------------|--------|------+
```

Public-key exchange for computing a shared private key [DH]

Diffie-Hellman parameters ’p’ and ’g’ must be the same values used by the ITR and ETR. The ITR computes public-key ’I’ and transmits ’I’ in a Map-Request packet. When the ETR receives the Map-Request, it uses parameters ’p’ and ’g’ to compute the ETR’s public key ’E’. The ETR transmits ’E’ in a Map-Reply message. At this point, the ETR has enough information to compute ’s’, the shared secret, by using ’I’ as the base and the ETR’s private key ’e’ as the exponent. When the ITR receives the Map-Reply, it uses the ETR’s public-key ’E’ with the ITR’s private key ’i’ to compute the same ’s’ shared secret the ETR computed. The value ’p’ is used as a modulus to create the width of the shared secret ’s’.

4. Encoding and Transmitting Key Material

The Diffie-Hellman key material is transmitted in Map-Request and Map-Reply messages. Diffie-Hellman parameters are encoded in the LISP Security Type LCAF [LCAF].

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 11   |      Rsvd2    |             6 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Key Count   |      Rsvd3    |A| Cipher Suite|   Rsvd4     |R|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Key Length          |     Public Key Material ...   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    ... Public Key Material                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |       Locator Address ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cipher Suite field contains DH Key Exchange and Cipher/Hash Functions

The ’Key Count’ field encodes the number of {’Key-Length’, ’Key-Material’} fields included in the encoded LCAF. The maximum number of keys that can be encoded are 3, each identified by key-id 1, followed by key-id 2, an finally key-id 3.

The ’R’ bit is not used for this use-case of the Security Type LCAF but is reserved for [LISP-DDT] security.

When the A-bit is set, it indicates that Authentication only is performed according to the Integrity hash function defined in the Cipher Suites. That is an encapsulator will perform an Integrity computation over an unencrypted packet and include an ICV value. Since the packet contains no ciphertext, there is no IV value included in the message. The 7-bit ’Cipher Suite’ field defines the following code-points:

```
Cipher Suite 0:
  Reserved

Cipher Suite 1:
  Diffie-Hellman Group: 1024-bit Modular Exponential (MODP) [RFC2409]
  Encryption:           AES with 128-bit keys in CBC mode [AES-CBC]
  Integrity:            HMAC-SHA1-96 [RFC2404]

Cipher Suite 2:
  Diffie-Hellman Group: 2048-bit MODP [RFC3526]
  Encryption:           AES with 128-bit keys in CBC mode [AES-CBC]
  Integrity:            HMAC-SHA1-96 [RFC2404]

Cipher Suite 3:
  Diffie-Hellman Group: 3072-bit MODP [RFC3526]
  Encryption:           AES with 128-bit keys in CBC mode [AES-CBC]
  Integrity:            HMAC-SHA1-96 [RFC2404]
```

The "Public Key Material" field contains the public key generated by one of the Cipher Suites defined above. The length of the key in octets is encoded in the "Key Length" field.

When an ITR or PITR send a Map-Request, they will encode their own RLOC in the Security Type LCAF format within the ITR-RLOCs field. When a ETR or RTR sends a Map-Reply, they will encode their RLOCs in Security Type LCAF format within the RLOC-record field of each EID-record supplied.

If an ITR or PITR sends a Map-Request with the Security Type LCAF included and the ETR or RTR does not want to have encapsulated traffic encrypted, they will return a Map-Reply with no RLOC records encoded with the Security Type LCAF. This signals to the ITR or PITR that it should not encrypt traffic (it cannot encrypt traffic anyways since no ETR public-key was returned).

Likewise, if an ITR or PITR wish to include multiple key-ids in the Map-Request but the ETR or RTR wish to use some but not all of the key-ids, they return a Map-Reply only for those key-ids they wish to use.

5. Shared Keys used for the Data-Plane

When an ITR or PITR receives a Map-Reply accepting the Cipher Suite sent in the Map-Request, it is ready to create data plane keys. The same process is followed by the ETR or RTR returning the Map-Reply.

The first step is to create a shared secret, using the peer’s shared Diffie-Hellman Public Key Material combined with device’s own private

keying material as described in Section 3. The Diffie-Hellman group used is defined in the Cipher Suite sent in the Map-Request and copied into the Map-Reply.

The resulting shared secret is used to compute Encryption and Integrity keys for the algorithms specified in the Cipher Suite. A Key Derivation Function (KDF) in counter mode as specified by [NIST-SP800-108] is used to generate the data-plane keys. The amount of keying material that is derived depends on the algorithms in the cipher suite.

The inputs to the KDF are as follows:

- o KDF function. This is HMAC-SHA-256.

- o A key for the KDF function. This is the most significant 16 octets of the computed Diffie-Hellman shared secret.

- o Context that binds the use of the data-plane keys to this session. The context is made up of the following fields, which are concatenated and provided as the data to be acted upon by the KDF function.

Context:

- o A counter, represented as a two-octet value in network-byte order.

- o The null-terminated string "lisp-crypto".

- o The ITR’s nonce from the the Map-Request the Cipher Suite was included in.

- o The number of bits of keying material required (L), represented as a two-octet value in network byte order.

The counter value in the context is first set to 1. When the amount of keying material exceeds the number of bits returned by the KDF function, then the KDF function is called again with the same inputs except that the counter increments for each call. When enough keying material is returned, it is concatenated and used to create keys.

For example, AES with 128-bit keys requires 16 octets (128 bits) of keying material, and HMAC-SHA1-96 requires another 16 octets (128 bits) of keying material in order to maintain a consistent 128-bits of security. Since 32 octets (256 bits) of keying material are required, and the KDF function HMAC-SHA-256 outputs 256 bits, only one call is required. The inputs are as follows:

```
key-material = HMAC-SHA-256(dh-shared-secret, context)

    where: context = 0x0001 || "lisp-crypto" || <itr-nonce> || 0x0100
```

In contrast, a cipher suite specifying AES with 256-bit keys requires 32 octets (256 bits) of keying material, and HMAC-SHA256-128 requires another 32 octets (256 bits) of keying material in order to maintain a consistent 256-bits of security. Since 64 octets (512 bits) of keying material are required, and the KDF function HMAC-SHA-256 outputs 256 bits, two calls are required.

```
key-material-1 = HMAC-SHA-256(dh-shared-secret, context)

    where: context = 0x0001 || "lisp-crypto" || <itr-nonce> || 0x0200

key-material-2 = HMAC-SHA-256(dh-shared-secret, context)

    where: context = 0x0002 || "lisp-crypto" || <itr-nonce> || 0x0200

key-material = key-material-1 || key-material-2
```

If the key-material is longer than the required number of bits (L), then only the most significant L bits are used.

From the derived key-material, the most significant bits are used for the Encryption key, and least significant bits are used for the Integrity key. For example, if the Cipher Suite contains both AES with 128-bit keys and HMAC-SHA1-96, the most significant 128 bits become the ITR’s data-plane encryption key, and the next 128-bit become the ITR’s Integrity key.

6. Data-Plane Operation

The LISP encapsulation header [RFC6830] requires changes to encode the key-id for the key being used for encryption.

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  / |       Source Port = xxxx      |       Dest Port = 4341        |
UDP +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  \ |           UDP Length          |        UDP Checksum           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
L / |N|L|E|V|I|P|K|K|            Nonce/Map-Version                  | \
I   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
S \ |                 Instance ID/Locator-Status-Bits               |  |
P   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
    |                    Initialization Vector (IV)                 |  I
E   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  C
n / |                                                               |  V
c   |                                                               |  |
r   |                Packet Payload with EID Header ...             |  |
y   |                                                               |  |
p \ |                                                               | /
t   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                    Integrity Check Value (ICV)                |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

K-bits indicate when packet is encrypted and which key used

When the KK bits are 00, the encapsulated packet is not encrypted. When the value of the KK bits are 1, 2, or 3, it encodes the key-id of the secret keys computed during the Diffie-Hellman Map-Request/ Map-Reply exchange. When the KK bits are not 0, the payload is prepended with an Initialization Vector (IV) and appended with an Integrity Check Value (ICV). The length of the IV and ICV fields depend on the Cipher Suite negotiated in the control-plane.

When an ITR or PITR receives a packet to be encapsulated, they will first decide what key to use, encode the key-id into the LISP header, and use that key to encrypt all packet data that follows the LISP header. Therefore, the outer header, UDP header, and LISP header travel as plaintext.

There is an open working group item to discuss if the data encapsulation header needs change for encryption or any new applications. This draft proposes changes to the existing header so experimentation can continue without making large changes to the data-plane at this time.

7. Procedures for Encryption and Decryption

When an ITR, PITR, or RTR encapsulate a packet and have already computed an encryption-key and integrity-key (detailed in section Section 5) that is associated with a destination RLOC, the following encryption and encapsulation procedures are performed:

1. The encapsulator creates a random number used as the IV. Prepends the IV value to the packet being encapsulated. The IV is incremented for every packet sent to the destination RLOC.

2. Next encrypt with cipher function AES-CBC using the encryption-key over the packet payload. This does not include the IV. The IV must be transmitted as plaintext so the decrypter can use it as input to the decryption cipher. The payload should be padded to an integral number of bytes a block cipher may require.

3. Prepend the LISP header. The key-id field of the LISP header is set to the key-id value that corresponds to key-pair used for the encryption cipher and for the ICV hash.

4. Next compute the ICV value by hashing the packet (which includes the LISP header, the IV, and the packet payload) with the HMAC-SHA1 function using the integrity-key. The resulting ICV value is appended to the packet. The ICV is not ciphertext so a fast integrity check can be performed without decryption at the receiver.

5. Lastly, prepend the UDP header and outer IP header onto the encrypted packet and send packet to destination RLOC.

When an ETR, PETR, or RTR receive an encapsulated packet, the following decapsulation and decryption procedures are performed:

1. The outer IP header and UDP header are stripped from the start of the packet and the ICV is stripped from the end of the packet.

2. Next the ICV is computed by running the Integrity function from the cipher suite using the integrity-key over the packet (which includes the LISP header, the IV and packet payload) using the integrity-key. If the result does not match the ICV value from the packet, the packet was been tampered with, and is dropped, and an optional log message may be issued. The integrity-key is obtained from a local-cache associated with the key-id value from the LISP header.

3. If the hashed result matches the ICV value from the packet, then the LISP header is stripped and decryption occurs over the packet payload using the plaintext IV in the packet.

4. The IV is stripped from the packet.

5. The packet is decrypted using the encryption-key and the IV from the packet. The encryption-key is obtained from a local-cache associated with the key-id value from the LISP header. The result of the decryption function is a plaintext packet payload.

6. The resulting packet is forwarded to the destination EID.

8. Dynamic Rekeying

Since multiple keys can be encoded in both control and data messages, an ITR can encapsulate and encrypt with a specific key while it is negotiating other keys with the same ETR. Soon as an ETR or RTR returns a Map-Reply, it should be prepared to decapsulate and decrypt using the new keys computed with the new Diffie-Hellman parameters received in the Map-Request and returned in the Map-Reply.

RLOC-probing can be used to change keys or cipher suites by the ITR at any time. And when an initial Map-Request is sent to populate the ITR’s map-cache, the Map-Request flows across the mapping system where a single ETR from the Map-Reply RLOC-set will respond. If the ITR decides to use the other RLOCs in the RLOC-set, it MUST send a Map-Request directly to negotiate security parameters with the ETR. This process may be used to test reachability from an ITR to an ETR initially when a map-cache entry is added for the first time, so an ITR can get both reachability status and keys negotiated with one Map-Request/Map-Reply exchange.

A rekeying event is defined to be when an ITR or PITR changes the cipher suite or public-key in the Map-Request. The ETR or RTR compares the cipher suite and public-key it last received from the ITR for the key-id, and if any value has changed, it computes a new public-key and cipher suite requested by the ITR from the Map-Request and returns it in the Map-Reply. Now a new shared secret is computed and can be used for the key-id for encryption by the ITR and decryption by the ETR. When the ITR or PITR starts this process of negotiating a new key, it must not use the corresponding key-id in encapsulated packets until it receives a Map-Reply from the ETR with the same cipher suite value it expects (the values it sent in a Map-Request).

Note when RLOC-probing continues to maintain RLOC reachability and rekeying is not desirable, the ITR or RTR can either not include the

Security Type LCAF in the Map-Request or supply the same key material as it received from the last Map-Reply from the ETR or RTR. This approach signals to the ETR or RTR that no rekeying event is requested.

9. Future Work

For performance considerations, newer Elliptic-Curve Diffie-Hellman (ECDH) groups can be used as specified in [RFC4492] and [RFC6090] to reduce CPU cycles required to compute shared secret keys.

For better security considerations as well as to be able to build faster software implementations, newer approaches to ciphers and authentication methods will be researched and tested. Some examples are chacha20 and poly1305 [CHACHA-POLY].

10. Security Considerations

10.1. SAAG Support

The LISP working group has and will continue to seek help from the SAAG working group for security advice. The SAAG has been involved early in the design process so they have early input and review.

10.2. LISP-Crypto Security Threats

Since ITRs and ETRs participate in key exchange over a public non-secure network, a man-in-the-middle (MITM) could circumvent the key exchange and compromise data-plane confidentiality. This can happen when the MITM is acting as a Map-Replier, provides its own public key so the ITR and the MITM generate a shared secret key among each other. If the MITM is in the data path between the ITR and ETR, it can use the shared secret key to decrypt traffic from the ITR.

Since LISP can secure Map-Replies by the authentication process specified in [LISP-SEC], the ITR can detect when a MITM has signed a Map-Reply for an EID-prefix it is not authoritative for. When an ITR determines the signature verification fails, it discards and does not reuse the key exchange parameters, avoids using the ETR for encapsulation, and issues a severe log message to the network administrator. Optionally, the ITR can send RLOC-probes to the compromised RLOC to determine if can reach the authoritative ETR. And when the ITR validates the signature of a Map-Reply, it can begin encrypting and encapsulating packets to the RLOC of ETR.

11. IANA Considerations

This draft may require the use of the registry that selects Security parameters. Rather than convey the key exchange parameters and crypto functions directly in LISP control packets, the cipher suite values can be assigned and defined in a registry. For example, Diffie-Hellman group-id values can be used from [RFC2409] and [RFC3526].

This draft specifies how the 7-bit cipher suite values from the Security Type LCAF are partitioned. The partitions are:

```
0:      Reserved
1-96:   Allocated by registry, but first 3 values defined in this document
97-127: Private use
```

12. References

12.1. Normative References

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, June 2005.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2011.

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

12.2. Informative References

[AES-CBC] McGrew, D., Foley, J., and K. Paterson, "Authenticated Encryption with AES-CBC and HMAC-SHA", draft-mcgrew-aead-aes-cbc-hmac-sha2-05.txt (work in progress).

[CHACHA-POLY] Langley, A., "ChaCha20 and Poly1305 based Cipher Suites for TLS", draft-agl-tls-chacha20poly1305-00 (work in progress).

[DH] "Diffie-Hellman key exchange", Wikipedia http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange.

[LCAF] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format", draft-ietf-lisp-lcaf-04.txt (work in progress).

[LISP-DDT] Fuller, V., Lewis, D., Ermaagan, V., and A. Jain, "LISP Delegated Database Tree", draft-fuller-lisp-ddt-03 (work in progress).

[LISP-SEC] Maino, F., Ermagan, V., Cabellos, A., and D. Saucez, "LISP-Secuirty (LISP-SEC)", draft-ietf-lisp-sec-06 (work in progress).

[NIST-SP800-108] "National Institute of Standards and Technology, "Recommendation for Key Derivation Using Pseudorandom Functions NIST SP800-108"", NIST SP 800-108, October 2009.

Appendix A. Acknowledgments

The author would like to thank Dan Harkins, Joel Halpern, Fabio Maino, Ed Lopez, Roger Jorgensen, Watson Ladd, and Ilari Liusvaara for their interest, suggestions, and discussions about LISP data-plane security.

In addition, the support and suggestions from the SAAG working group were helpful and appreciative.

Appendix B. Document Change Log

B.1. Changes to draft-ietf-lisp-crypto-01.txt

- o Posted May 2015.
- o Create cipher suites and encode them in the Security LCAF.
- o Add IV to beginning of packet header and ICV to end of packet.
- o AEAD procedures are now part of encrpytion process.

B.2. Changes to draft-ietf-lisp-crypto-00.txt

- o Posted January 2015.
- o Changing draft-farinacci-lisp-crypto-01 to draft-ietf-lisp-crypto-00. This draft has become a working group document
- o Add text to indicate the working group may work on a new data encapsulation header format for data-plane encryption.

B.3. Changes to draft-farinacci-lisp-crypto-01.txt

- o Posted July 2014.
- o Add Group-ID to the encoding format of Key Material in a Security Type LCAF and modify the IANA Considerations so this draft can use key exchange parameters from the IANA registry.
- o Indicate that the R-bit in the Security Type LCAF is not used by lisp-crypto.
- o Add text to indicate that ETRs/RTRs can negotiate less number of keys from which the ITR/PITR sent in a Map-Request.
- o Add text explaining how LISP-SEC solves the problem when a man-in-the-middle becomes part of the Map-Request/Map-Reply key exchange process.
- o Add text indicating that when RLOC-probing is used for RLOC reachability purposes and rekeying is not desired, that the same key exchange parameters should be used so a reallocation of a pubic key does not happen at the ETR.
- o Add text to indicate that ECDH can be used to reduce CPU requirements for computing shared secret-keys.

B.4. Changes to draft-farinacci-lisp-crypto-00.txt

- o Initial draft posted February 2014.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, California 95120
USA

Phone: 408-718-2001
Email: farinacci@gmail.com

Brian Weis
cisco Systems
170 West Tasman Drive
San Jose, California 95124-1706
USA

Phone: 408-526-4796
Email: bew@cisco.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: December 14, 2015

D. Farinacci
lispers.net
D. Meyer
Brocade
J. Snijders
NTT Communications
June 12, 2015

LISP Canonical Address Format (LCAF)
draft-ietf-lisp-lcaf-10

Abstract

This draft defines a canonical address format encoding used in LISP control messages and in the encoding of lookup keys for the LISP Mapping Database System.

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 December 14, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Definition of Terms 4
3. LISP Canonical Address Format Encodings 4
4. LISP Canonical Address Applications 7
4.1. Segmentation using LISP 7
4.2. Carrying AS Numbers in the Mapping Database 8
4.3. Convey Application Specific Data 9
4.4. Assigning Geo Coordinates to Locator Addresses 10
4.5. Generic Database Mapping Lookups 12
4.6. NAT Traversal Scenarios 13
4.7. PETR Admission Control Functionality 15
4.8. Multicast Group Membership Information 16
4.9. Traffic Engineering using Re-encapsulating Tunnels . . . 18
4.10. Storing Security Data in the Mapping Database 19
4.11. Source/Destination 2-Tuple Lookups 20
4.12. Replication List Entries for Multicast Forwarding 21
4.13. Data Model Encoding 22
4.14. Encoding Key/Value Address Pairs 23
4.15. Multiple Data-Planes 24
4.16. Applications for AFI List Type 26
4.16.1. Binding IPv4 and IPv6 Addresses 26
4.16.2. Layer-2 VPNs 27
4.16.3. ASCII Names in the Mapping Database 28
4.16.4. Using Recursive LISP Canonical Address Encodings . . 29
4.16.5. Compatibility Mode Use Case 30
5. Security Considerations 31
6. IANA Considerations 31
7. References 32
7.1. Normative References 32
7.2. Informative References 33
Appendix A. Acknowledgments 34
Appendix B. Document Change Log 35
B.1. Changes to draft-ietf-lisp-lcaf-10.txt 35
B.2. Changes to draft-ietf-lisp-lcaf-09.txt 35
B.3. Changes to draft-ietf-lisp-lcaf-08.txt 35
B.4. Changes to draft-ietf-lisp-lcaf-07.txt 36
B.5. Changes to draft-ietf-lisp-lcaf-06.txt 36
B.6. Changes to draft-ietf-lisp-lcaf-05.txt 36
B.7. Changes to draft-ietf-lisp-lcaf-04.txt 36
B.8. Changes to draft-ietf-lisp-lcaf-03.txt 36
B.9. Changes to draft-ietf-lisp-lcaf-02.txt 37
B.10. Changes to draft-ietf-lisp-lcaf-01.txt 37
B.11. Changes to draft-ietf-lisp-lcaf-00.txt 37

Authors' Addresses 37

1. Introduction

The LISP architecture and protocols [RFC6830] introduces two new numbering spaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) which are intended to replace most use of IP addresses on the Internet. To provide flexibility for current and future applications, these values can be encoded in LISP control messages using a general syntax that includes Address Family Identifier (AFI), length, and value fields.

Currently defined AFIs include IPv4 and IPv6 addresses, which are formatted according to code-points assigned in [AFI] as follows:

IPv4 Encoded Address:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            AFI = 1            |       IPv4 Address ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ...  IPv4 Address         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IPv6 Encoded Address:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            AFI = 2            |       IPv6 Address ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ...  IPv6 Address  ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ...  IPv6 Address  ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ...  IPv6 Address  ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ...  IPv6 Address         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This document describes the currently-defined AFIs the LISP protocol uses along with their encodings and introduces the LISP Canonical Address Format (LCAF) that can be used to define the LISP-specific encodings for arbitrary AFI values.

2. Definition of Terms

Address Family Identifier (AFI): a term used to describe an address encoding in a packet. An address family currently defined for IPv4 or IPv6 addresses. See [AFI] and [RFC1700] for details. The reserved AFI value of 0 is used in this specification to indicate an unspecified encoded address where the the length of the address is 0 bytes following the 16-bit AFI value of 0.

Unspecified Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            AFI = 0            |    <nothing follows AFI=0>    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Endpoint ID (EID): a 32-bit (for IPv4) or 128-bit (for IPv6) value used in the source and destination address fields of the first (most inner) LISP header of a packet. The host obtains a destination EID the same way it obtains a destination address today, for example through a DNS lookup or SIP exchange. The source EID is obtained via existing mechanisms used to set a host’s "local" IP address. An EID is allocated to a host from an EID-prefix block associated with the site where the host is located. An EID can be used by a host to refer to other hosts.

Routing Locator (RLOC): the IPv4 or IPv6 address of an egress tunnel router (ETR). It is the output of a EID-to-RLOC mapping lookup. An EID maps to one or more RLOCs. Typically, RLOCs are numbered from topologically aggregatable blocks that are assigned to a site at each point to which it attaches to the global Internet; where the topology is defined by the connectivity of provider networks, RLOCs can be thought of as PA addresses. Multiple RLOCs can be assigned to the same ETR device or to multiple ETR devices at a site.

3. LISP Canonical Address Format Encodings

IANA has assigned AFI value 16387 (0x4003) to the LISP architecture and protocols. This specification defines the encoding format of the LISP Canonical Address (LCA).

The Address Family AFI definitions from [AFI] only allocate code-points for the AFI value itself. The length of the address or entity that follows is not defined and is implied based on conventional experience. Where the LISP protocol uses LISP Canonical Addresses

specifically, the address length definitions will be in this specification and take precedent over any other specification.

The first 6 bytes of an LISP Canonical Address are followed by a variable length of fields:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type       |     Rsvd2     |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Rsvd1: this 8-bit field is reserved for future use and MUST be transmitted as 0 and ignored on receipt.

Flags: this 8-bit field is for future definition and use. For now, set to zero on transmission and ignored on receipt.

Type: this 8-bit field is specific to the LISP Canonical Address formatted encodings, values are:

Type 0: Null Body Type

Type 1: AFI List Type

Type 2: Instance ID Type

Type 3: AS Number Type

Type 4: Application Data Type

Type 5: Geo Coordinates Type

Type 6: Opaque Key Type

Type 7: NAT-Traversal Type

Type 8: Nonce Locator Type

Type 9: Multicast Info Type

Type 10: Explicit Locator Path Type

Type 11: Security Key Type

Type 12: Source/Dest Key Type

Type 13: Replication List Entry Type

Type 14: JSON Data Model Type

Type 15: Key/Value Address Pair Type

Type 16: Encapsulation Format Type

Rsvd2: this 8-bit field is reserved for future use and MUST be transmitted as 0 and ignored on receipt.

Length: this 16-bit field is in units of bytes and covers all of the LISP Canonical Address payload, starting and including the byte after the Length field. So any LCAF encoded address will have a minimum length of 8 bytes when the Length field is 0. The 8 bytes include the AFI, Flags, Type, Reserved, and Length fields. When the AFI is not next to encoded address in a control message, then the encoded address will have a minimum length of 6 bytes when the Length field is 0. The 6 bytes include the Flags, Type, Reserved, and Length fields.

[RFC6830] states RLOC records are sorted when encoded in control messages so the locator-set has consistent order across all xTRs for a given EID. The sort order is based on sort-key {afi, RLOC-address}. When an RLOC is LCAF encoded, the sort-key is {afi, LCAF-Type, payload}. Therefore, when a locator-set has a mix of AFI records and LCAF records, all LCAF records will appear after all the AFI records.

4. LISP Canonical Address Applications

4.1. Segmentation using LISP

When multiple organizations inside of a LISP site are using private addresses [RFC1918] as EID-prefixes, their address spaces must remain segregated due to possible address duplication. An Instance ID in the address encoding can aid in making the entire AFI based address unique.

Another use for the Instance ID LISP Canonical Address Format is when creating multiple segmented VPNs inside of a LISP site where keeping EID-prefix based subnets is desirable.

Instance ID LISP Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 2    | IID mask-len  |             4 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Instance ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |         Address  ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IID mask-len: if the AFI is set to 0, then this format is not encoding an extended EID-prefix but rather an instance-ID range where the 'IID mask-len' indicates the number of high-order bits used in the Instance ID field for the range.

Length value n: length in bytes of the AFI address that follows the Instance ID field including the AFI field itself.

Instance ID: the low-order 24-bits that can go into a LISP data header when the I-bit is set. See [RFC6830] for details.

AFI = x: x can be any AFI value from [AFI].

This LISP Canonical Address Type can be used to encode either EID or RLOC addresses.

4.2. Carrying AS Numbers in the Mapping Database

When an AS number is stored in the LISP Mapping Database System for either policy or documentation reasons, it can be encoded in a LISP Canonical Address.

AS Number LISP Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|n   Type = 3    |     Rsvd2     |             4 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           AS Number                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |         Address  ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of the AFI address that follows the AS Number field including the AFI field itself.

AS Number: the 32-bit AS number of the autonomous system that has been assigned either the EID or RLOC that follows.

AFI = x: x can be any AFI value from [AFI].

The AS Number Canonical Address Type can be used to encode either EID or RLOC addresses. The former is used to describe the LISP-ALT AS number the EID-prefix for the site is being carried for. The latter is used to describe the AS that is carrying RLOC based prefixes in the underlying routing system.

4.3. Convey Application Specific Data

When a locator-set needs to be conveyed based on the type of application or the Per-Hop Behavior (PHB) of a packet, the Application Data Type can be used.

Application Data LISP Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 4    |     Rsvd2     |            12 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       IP TOS, IPv6 TC, or Flow Label          |    Protocol   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Local Port (lower-range)   |    Local Port (upper-range)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Remote Port (lower-range)   |   Remote Port (upper-range)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |         Address  ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of the AFI address that follows the 8-byte Application Data fields including the AFI field itself.

IP TOS, IPv6 TC, or Flow Label: this field stores the 8-bit IPv4 TOS field used in an IPv4 header, the 8-bit IPv6 Traffic Class or Flow Label used in an IPv6 header.

Local Port/Remote Port Ranges: these fields are from the TCP, UDP, or SCTP transport header. A range can be specified by using a lower value and an upper value. When a single port is encoded, the lower and upper value fields are the same.

AFI = x: x can be any AFI value from [AFI].

The Application Data Canonical Address Type is used for an EID encoding when an ITR wants a locator-set for a specific application. When used for an RLOC encoding, the ETR is supplying a locator-set for each specific application is has been configured to advertise.

4.4. Assigning Geo Coordinates to Locator Addresses

If an ETR desires to send a Map-Reply describing the Geo Coordinates for each locator in its locator-set, it can use the Geo Coordinate Type to convey physical location information.

Coordinates are specified using the WGS-84 (World Geodetic System) reference coordinate system [WGS-84].

Geo Coordinate LISP Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 5    |     Rsvd2     |            12 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|N|     Latitude Degrees        |    Minutes    |    Seconds    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|     Longitude Degrees       |    Minutes    |    Seconds    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Altitude                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |         Address  ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of the AFI address that follows the 8-byte Longitude and Latitude fields including the AFI field itself.

N: When set to 1 means North, otherwise South.

Latitude Degrees: Valid values range from 0 to 90 degrees above or below the equator (northern or southern hemisphere, respectively).

Latitude Minutes: Valid values range from 0 to 59.

Latitude Seconds: Valid values range from 0 to 59.

E: When set to 1 means East, otherwise West.

Longitude Degrees: Value values are from 0 to 180 degrees right or left of the Prime Meridian.

Longitude Minutes: Valid values range from 0 to 59.

Longitude Seconds: Valid values range from 0 to 59.

Altitude: Height relative to sea level in meters. This is a signed integer meaning that the altitude could be below sea level. A value of 0x7fffffff indicates no Altitude value is encoded.

AFI = x: x can be any AFI value from [AFI].

The Geo Coordinates Canonical Address Type can be used to encode either EID or RLOC addresses. When used for EID encodings, you can determine the physical location of an EID along with the topological location by observing the locator-set.

4.5. Generic Database Mapping Lookups

When the LISP Mapping Database system holds information accessed by a generic formatted key (where the key is not the usual IPv4 or IPv6 address), an opaque key may be desirable.

Opaque Key LISP Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 6    |     Rsvd2     |             3 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Key Field Num |      Key Wildcard Fields      |   Key . . .   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       . . . Key                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of the type’s payload. The value n is the number of bytes that follow this Length field.

Key Field Num: the number of fields (minus 1) the key can be broken up into. The width of the fields are fixed length. So for a key size of 8 bytes, with a Key Field Num of 4 allows 4 fields of 2 bytes in length. Valid values for this field range from 0 to 15 supporting a maximum of 16 field separations.

Key Wildcard Fields: describes which fields in the key are not used as part of the key lookup. This wildcard encoding is a bitfield. Each bit is a don’t-care bit for a corresponding field in the key. Bit 0 (the low-order bit) in this bitfield corresponds the first field, right-justified in the key, bit 1 the second field, and so on. When a bit is set in the bitfield it is a don’t-care bit and should not be considered as part of the database lookup. When the entire 16-bits is set to 0, then all bits of the key are used for the database lookup.

Key: the variable length key used to do a LISP Database Mapping lookup. The length of the key is the value n (shown above) minus 3.

4.6. NAT Traversal Scenarios

When a LISP system is conveying global address and mapped port information when traversing through a NAT device, the NAT-Traversal LCAF Type is used. See [LISP-NATT] for details.

NAT-Traversal Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 7    |     Rsvd2     |             4 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       MS UDP Port Number      |      ETR UDP Port Number      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |  Global ETR RLOC Address  ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |       MS RLOC Address  ...    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          | Private ETR RLOC Address  ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |      RTR RLOC Address 1 ...   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |      RTR RLOC Address k ...   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of the AFI addresses that follows the UDP Port Number field including the AFI fields themselves.

MS UDP Port Number: this is the UDP port number of the Map-Server and is set to 4342.

ETR UDP Port Number: this is the port number returned to a LISP system which was copied from the source port from a packet that has flowed through a NAT device.

AFI = x: x can be any AFI value from [AFI].

Global ETR RLOC Address: this is an address known to be globally unique built by NAT-traversal functionality in a LISP router.

MS RLOC Address: this is the address of the Map-Server used in the destination RLOC of a packet that has flowed through a NAT device.

Private ETR RLOC Address: this is an address known to be a private address inserted in this LCAF format by a LISP router that resides on the private side of a NAT device.

RTR RLOC Address: this is an encapsulation address used by an ITR or PITR which resides behind a NAT device. This address is known to have state in a NAT device so packets can flow from it to the LISP ETR behind the NAT. There can be one or more NTR addresses supplied in these set of fields. The number of NTRs encoded is determined by the LCAF length field. When there are no NTRs supplied, the NTR fields can be omitted and reflected by the LCAF length field or an AFI of 0 can be used to indicate zero NTRs encoded.

4.7. PETR Admission Control Functionality

When a public PETR device wants to verify who is encapsulating to it, it can check for a specific nonce value in the LISP encapsulated packet. To convey the nonce to admitted ITRs or PITRs, this LCAF format is used in a Map-Register or Map-Reply locator-record.

Nonce Locator Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 8    |     Rsvd2     |             4 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Reserved    |                  Nonce                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |         Address  ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of the AFI address that follows the Nonce field including the AFI field itself.

Reserved: must be set to zero and ignore on receipt.

Nonce: this is a nonce value returned by an ETR in a Map-Reply locator-record to be used by an ITR or PITR when encapsulating to the locator address encoded in the AFI field of this LCAF type.

AFI = x: x can be any AFI value from [AFI].

4.8. Multicast Group Membership Information

Multicast group information can be published in the mapping database so a lookup on an EID based group address can return a replication list of group addresses or a unicast addresses for single replication or multiple head-end replications. The intent of this type of unicast replication is to deliver packets to multiple ETRs at receiver LISP multicast sites. The locator-set encoding for this EID record type can be a list of ETRs when they each register with "Merge Semantics". The encoding can be a typical AFI encoded locator address. When an RTR list is being registered (with multiple levels according to [LISP-RE]), the Replication List Entry LCAF type is used for locator encoding.

This LCAF encoding can be used to send broadcast packets to all members of a subnet when each EIDs are away from their home subnet location.

Multicast Info Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 9    |  Rsvd2  |R|L|J|             8 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Instance-ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Reserved           | Source MaskLen| Group MaskLen |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |   Source/Subnet Address  ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |       Group Address  ...      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of fields that follow.

Reserved: must be set to zero and ignore on receipt.

R-bit: this is the RP-bit that represents PIM (S,G,RP-bit) multicast state. This bit can be set for Joins (when the J-bit is set) or for Leaves (when the L-bit is set). See [LISP-MRSIG] for more usage details.

L-bit: this is the Leave-Request bit and is used when this LCAF type is present in the destination EID-prefix field of a Map-Request. See [LISP-MRSIG] for details.

J-bit: this is the Join-Request bit and is used when this LCAF type is present in the destination EID-prefix field of a Map-Request. See [LISP-MRSIG] for details. The J-bit MUST not be set when the L-bit is also set in the same LCAF block. A receiver should not take any specific Join or Leave action when both bits are set.

Instance ID: the low-order 24-bits that can go into a LISP data header when the I-bit is set. See [RFC6830] for details. The use of the Instance-ID in this LCAF type is to associate a multicast forwarding entry for a given VPN. The instance-ID describes the VPN and is registered to the mapping database system as a 3-tuple of (Instance-ID, S-prefix, G-prefix).

Source MaskLen: the mask length of the source prefix that follows.

Group MaskLen: the mask length of the group prefix that follows.

AFI = x: x can be any AFI value from [AFI]. When a specific AFI has its own encoding of a multicast address, this field must be either a group address or a broadcast address.

4.9. Traffic Engineering using Re-encapsulating Tunnels

For a given EID lookup into the mapping database, this LCAF format can be returned to provide a list of locators in an explicit re-encapsulation path. See [LISP-TE] for details.

Explicit Locator Path (ELP) Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 10   |     Rsvd2     |               n               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Rsvd3         |L|P|S|           AFI = x             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Reencap Hop 1  ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Rsvd3         |L|P|S|           AFI = x             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Reencap Hop k  ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of fields that follow.

Lookup bit (L): this is the Lookup bit used to indicate to the user of the ELP to not use this address for encapsulation but to look it up in the mapping database system to obtain an encapsulating RLOC address.

RLOC-Probe bit (P): this is the RLOC-probe bit which means the Reencap Hop allows RLOC-probe messages to be sent to it. When the R-bit is set to 0, RLOC-probes must not be sent. When a Reencap Hop is an anycast address then multiple physical Reencap Hops are using the same RLOC address. In this case, RLOC-probes are not needed because when the closest RLOC address is not reachable another RLOC address can reachable.

Strict bit (S): this the strict bit which means the associated Rencap Hop is required to be used. If this bit is 0, the reencapsulator can skip this Reencap Hop and go to the next one in the list.

AFI = x: x can be any AFI value from [AFI]. When a specific AFI has its own encoding of a multicast address, this field must be either a group address or a broadcast address.

4.10. Storing Security Data in the Mapping Database

When a locator in a locator-set has a security key associated with it, this LCAF format will be used to encode key material. See [LISP-DDT] for details.

Security Key Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 11   |      Rsvd2    |             6 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Key Count   |      Rsvd3    | Key Algorithm |   Rsvd4     |R|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Key Length          |       Key Material ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        ... Key Material                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |       Locator Address ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of fields that start with the Key Material field.

Key Count: the Key Count field declares the number of Key sections included in this LCAF.

Key Algorithm: the Algorithm field identifies the key's cryptographic algorithm and specifies the format of the Public Key field.

R bit: this is the revoke bit and, if set, it specifies that this Key is being Revoked.

Key Length: this field determines the length in bytes of the Key Material field.

Key Material: the Key Material field stores the key material. The format of the key material stored depends on the Key Algorithm field.

AFI = x: x can be any AFI value from [AFI].This is the locator address that owns the encoded security key.

4.11. Source/Destination 2-Tuple Lookups

When both a source and destination address of a flow needs consideration for different locator-sets, this 2-tuple key is used in EID fields in LISP control messages. When the Source/Dest key is registered to the mapping database, it can be encoded as a source-prefix and destination-prefix. When the Source/Dest is used as a key for a mapping database lookup the source and destination come from a data packet.

Source/Dest Key Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 12   |     Rsvd2     |             4 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Reserved           |   Source-ML   |    Dest-ML    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |         Source-Prefix ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |     Destination-Prefix ...    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of fields that follow.

Reserved: must be set to zero and ignore on receipt.

Source-ML: the mask length of the source prefix that follows.

Dest-ML: the mask length of the destination prefix that follows.

AFI = x: x can be any AFI value from [AFI]. When a specific AFI has its own encoding of a multicast address, this field must be either a group address or a broadcast address.

Refer to [LISP-TE] for usage details.

4.12. Replication List Entries for Multicast Forwarding

The Replication List Entry LCAF type is an encoding for a locator being used for unicast replication according to the specification in [LISP-RE]. This locator encoding is pointed to by a Multicast Info LCAF Type and is registered by Re-encapsulating Tunnel Routers (RTRs) that are participating in an overlay distribution tree. Each RTR will register its locator address and its configured level in the distribution tree.

Replication List Entry Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 13   |    Rsvd2      |             4 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Rsvd3            |     Rsvd4     |  Level Value  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |           RTR/ETR #1 ...      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Rsvd3            |     Rsvd4     |  Level Value  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |           RTR/ETR  #n ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of fields that follow.

Rsvd{1,2,3,4}: must be set to zero and ignore on receipt.

Level Value: this value is associated with the level within the overlay distribution tree hierarchy where the RTR resides. The level numbers are ordered from lowest value being close to the ITR (meaning that ITRs replicate to level-0 RTRs) and higher levels are further downstream on the distribution tree closer to ETRs of multicast receiver sites.

AFI = x: x can be any AFI value from [AFI]. A specific AFI has its own encoding of either a unicast or multicast locator address. All RTR/ETR entries for the same level should be combined together by a Map-Server to avoid searching through the entire multi-level list of locator entries in a Map-Reply message.

4.13. Data Model Encoding

This type allows a JSON data model to be encoded either as an EID or RLOC.

JSON Data Model Type Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 14   |    Rsvd2    |B|             2 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           JSON length         | JSON binary/text encoding ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |       Optional Address ...    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of fields that follow.

Rsvd{1,2}: must be set to zero and ignore on receipt.

B bit: indicates that the JSON field is binary encoded according to [JSON-BINARY] when the bit is set to 1. Otherwise the encoding is based on text encoding according to [RFC4627].

JSON length: length in octets of the following ’JSON binary/text encoding’ field.

JSON binary/text encoding field: a variable length field that contains either binary or text encodings.

AFI = x: x can be any AFI value from [AFI]. A specific AFI has its own encoding of either a unicast or multicast locator address. All RTR/ETR entries for the same level should be combined together by a Map-Server to avoid searching through the entire multi-level list of locator entries in a Map-Reply message.

4.14. Encoding Key/Value Address Pairs

The Key/Value pair is for example useful for attaching attributes to other elements of LISP packets, such as EIDs or RLOCs. When attaching attributes to EIDs or RLOCs, it's necessary to distinguish between the element that should be used as EID or RLOC, and hence as key for lookups, and additional attributes. This is especially the case when the difference cannot be determined from the types of the elements, such as when two IP addresses are being used.

Key/Value Pair Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 15   |     Rsvd2     |               n               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |       Address as Key ...      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |       Address as Value ...    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes of fields that follow.

Rsvd{1,2}: must be set to zero and ignore on receipt.

AFI = x: x can be any AFI value from [AFI]. A specific AFI has its own encoding of either a unicast or multicast locator address. All RTR/ETR entries for the same level should be combined together by a Map-Server to avoid searching through the entire multi-level list of locator entries in a Map-Reply message.

Address as Key: this AFI encoded address will be attached with the attributes encoded in "Address as Value" which follows this field.

Address as Value: this AFI encoded address will be the attribute address that goes along with "Address as Key" which precedes this field.

4.15. Multiple Data-Planes

Overlays are becoming popular in many parts of the network which have created an explosion of data-plane encapsulation headers. Since the LISP mapping system can hold many types of address formats, it can represent the encapsulation format supported by an RLOC as well. When an encapsulator receives a Map-Reply with an Encapsulation Format LCAF Type encoded in an RLOC-record, it can select an encapsulation format, that it can support, from any of the encapsulation protocols which have the bit set to 1 in this LCAF type.

Encapsulation Format Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 16   |     Rsvd2     |             4 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Reserved-for-Future-Encapsulations       |U|G|N|v|V|l|L|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = x          |          Address ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Rsvd1/Rsvd2: must be set to zero and ignored on receipt.

Length value n: length in bytes of the AFI address that follows the next 32-bits including the AFI field itself.

Reserved-for-Future-Encapsulations: must be set to zero and ignored on receipt. This field will get bits allocated to future encapsulations, as they are created.

L: The RLOCs listed in the AFI encoded addresses in the next longword can accept layer3 LISP encapsulation using destination UDP port 4341 [RFC6830].

l: The RLOCs listed in the AFI encoded addresses in the next longword can accept layer2 LISP encapsulation using destination UDP port 8472 [L2-LISP].

V: The RLOCs listed in the AFI encoded addresses in the next longword can accept VXLAN encapsulation using destination UDP port 4789 [RFC7348].

v: The RLOCs listed in the AFI encoded addresses in the next longword can accept VXLAN-GPE encapsulation using destination UDP port 4790 [GPE].

N: The RLOCs listed in the AFI encoded addresses in the next longword can accept NV-GRE encapsulation using IPv4/ IPv6 protocol number 47 [NVGRE].

G: The RLOCs listed in the AFI encoded addresses in the next longword can accept GENEVE encapsulation using destination UDP port 6081 [GENEVE].

U: The RLOCs listed in the AFI encoded addresses in the next longword can accept GUE encapsulation using destination UDP port TBD [GUE].

4.16. Applications for AFI List Type

4.16.1. Binding IPv4 and IPv6 Addresses

When header translation between IPv4 and IPv6 is desirable a LISP Canonical Address can use the AFI List Type to carry multiple AFIs in one LCAF AFI.

Address Binding LISP Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 1    |     Rsvd2     |         2 + 4 + 2 + 16        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            AFI = 1            |       IPv4 Address ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ...  IPv4 Address         |            AFI = 2            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          IPv6 Address ...                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ...  IPv6 Address  ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ...  IPv6 Address  ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ...  IPv6 Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length: length in bytes is fixed at 24 when IPv4 and IPv6 AFI encoded addresses are used.

This type of address format can be included in a Map-Request when the address is being used as an EID, but the Mapping Database System lookup destination can use only the IPv4 address. This is so a Mapping Database Service Transport System, such as LISP-ALT [RFC6836], can use the Map-Request destination address to route the control message to the desired LISP site.

4.16.2. Layer-2 VPNs

When MAC addresses are stored in the LISP Mapping Database System, the AFI List Type can be used to carry AFI 6.

MAC Address LISP Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 1    |     Rsvd2     |             2 + 6             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             AFI = 6           |    Layer-2 MAC Address  ...   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ... Layer-2 MAC Address                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length: length in bytes is fixed at 8 when MAC address AFI encoded addresses are used.

This address format can be used to connect layer-2 domains together using LISP over an IPv4 or IPv6 core network to create a layer-2 VPN. In this use-case, a MAC address is being used as an EID, and the locator-set that this EID maps to can be an IPv4 or IPv6 RLOCs, or even another MAC address being used as an RLOC.

4.16.3. ASCII Names in the Mapping Database

If DNS names or URIs are stored in the LISP Mapping Database System, the AFI List Type can be used to carry an ASCII string where it is delimited by length ’n’ of the LCAF Length encoding.

ASCII LISP Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 1    |     Rsvd2     |             2 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             AFI = 17          |      DNS Name or URI  ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length value n: length in bytes AFI=17 field and the null-terminated ASCII string (the last byte of 0 is included).

4.16.4. Using Recursive LISP Canonical Address Encodings

When any combination of above is desirable, the AFI List Type value can be used to carry within the LCAF AFI another LCAF AFI.

Recursive LISP Canonical Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 1    |     Rsvd2     |             8 + 18            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 4    |     Rsvd2     |            12 + 6             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   IP TOS, IPv6 QQS or Flow Label              |    Protocol   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Local Port (lower-range)   |    Local Port (upper-range)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Remote Port (lower-range)   |   Remote Port (upper-range)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            AFI = 1            |       IPv4 Address ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ...  IPv4 Address         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length: length in bytes is fixed at 18 when an AFI=1 IPv4 address is included.

This format could be used by a Mapping Database Transport System, such as LISP-ALT [RFC6836], where the AFI=1 IPv4 address is used as an EID and placed in the Map-Request destination address by the sending LISP system. The ALT system can deliver the Map-Request to the LISP destination site independent of the Application Data Type AFI payload values. When this AFI is processed by the destination LISP site, it can return different locator-sets based on the type of application or level of service that is being requested.

4.16.5. Compatibility Mode Use Case

A LISP system should use the AFI List Type format when sending to LISP systems that do not support a particular LCAF Type used to encode locators. This allows the receiving system to be able to parse a locator address for encapsulation purposes. The list of AFIs in an AFI List LCAF Type has no semantic ordering and a receiver should parse each AFI element no matter what the ordering.

Compatibility Mode Address 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 1    |     Rsvd2     |          8 + 14 + 6           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 5    |     Rsvd2     |            12 + 2             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|N|     Latitude Degrees        |    Minutes    |    Seconds    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|     Longitude Degrees       |    Minutes    |    Seconds    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Altitude                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              AFI = 0          |           AFI = 1             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          IPv4 Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If a system does not recognized the Geo Coordinate LCAF Type that is accompanying a locator address, an encoder can include the Geo Coordinate LCAF Type embedded in a AFI List LCAF Type where the AFI in the Geo Coordinate LCAF is set to 0 and the AFI encoded next in the list is encoded with a valid AFI value to identify the locator address.

A LISP system is required to support the AFI List LCAF Type to use this procedure. It would skip over 10 bytes of the Geo Coordinate LCAF Type to get to the locator address encoding (an IPv4 locator address). A LISP system that does support the Geo Coordinate LCAF Type can support parsing the locator address within the Geo Coordinate LCAF encoding or in the locator encoding that follows in the AFI List LCAF.

5. Security Considerations

There are no security considerations for this specification. The security considerations are documented for the protocols that use LISP Canonical Addressing. Refer to the those relevant specifications.

6. IANA Considerations

This document defines a canonical address format encoding used in LISP control messages and in the encoding of lookup keys for the LISP Mapping Database System. Such address format is based on a fixed AFI (16387) and a LISP LCAF Type field.

The LISP LCAF Type field is an 8-bit field specific to the LISP Canonical Address formatted encodings, for which IANA is to create and maintain a new registry (as outlined in [RFC5226]) entitled "LISP LCAF Type". Initial values for the LISP LCAF Type registry are given below. Future assignments are to be made through RFC Publication. Assignments consist of a LISP LCAF Type name and its associated value:

Value	LISP LCAF Type Name	Definition
0	Null Body Type	Section 3
1	AFI List Type	Section 3
2	Instance ID Type	Section 3
3	AS Number Type	Section 3
4	Application Data Type	Section 3
5	Geo Coordinates Type	Section 3
6	Opaque Key Type	Section 3
7	NAT-Traversal Type	Section 3
8	Nonce Locator Type	Section 3
9	Multicast Info Type	Section 3
10	Explicit Locator Path Type	Section 3
11	Security Key Type	Section 3
12	Source/Dest Key Type	Section 3
13	Replication List Entry Type	Section 3
14	JSON Data Model Type	Section 3
15	Key/Value Address Pair Type	Section 3
16	Encapsulation Format Type	Section 3

Table 1: LISP LCAF Type Initial Values

7. References

7.1. Normative References

[RFC1700] Reynolds, J. and J. Postel, "Assigned Numbers", RFC 1700, October 1994.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

[RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, January 2013.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.

7.2. Informative References

[AFI] IANA, , "Address Family Identifier (AFIs)", ADDRESS FAMILY NUMBERS http://www.iana.org/numbers.html, Febuary 2007.

[GENEVE] Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, I., Agarwal, P., Duda, K., Dutt, D., and J. Hudson, "Geneve: Generic Network Virtualization Encapsulation", draft-gross-geneve-02 (work in progress).

[GPE] Quinn, P., Agarwal, P., Fernando, R., Kreeger, L., Kreeger, L., Lewis, D., Maino, F., Smith, M., Yadav, N., Yong, L., Xu, X., Elzur, U., and P. Garg, "Generic Protocol Extension for VXLAN", draft-quinn-vxlan-gpe-03.txt (work in progress).

[GUE] Herbert, T. and L. Yong, "Generic UDP Encapsulation", draft-herbert-gue-02.txt (work in progress).

[JSON-BINARY] "Universal Binary JSON Specification", URL http://ubjson.org.

[L2-LISP] Smith, M., Dutt, D., Farinacci, D., and F. Maino, "Layer 2 (L2) LISP Encapsulation Format", draft-smith-lisp-layer2-03.txt (work in progress).

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

[LISP-MRSIG] Farinacci, D. and M. Napierala, "LISP Control-Plane Multicast Signaling", draft-farinacci-lisp-mr-signaling-03.txt (work in progress).

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

[LISP-RE] Coras, F., Cabellos-Aparicio, A., Domingo-Pascual, J., Maino, F., and D. Farinacci, "LISP Replication Engineering", draft-coras-lisp-re-03.txt (work in progress).

[LISP-TE] Farinacci, D., Lahiri, P., and M. Kowal, "LISP Traffic Engineering Use-Cases", draft-farinacci-lisp-te-03.txt (work in progress).

[NVGRE] Sridharan, M., Greenberg, A., Wang, Y., Garg, P., Venkataramiah, N., Duda, K., Ganga, I., Lin, G., Pearson, M., Thaler, P., and C. Tumuluri, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-06.txt (work in progress).

[WGS-84] Geodesy and Geophysics Department, DoD., "World Geodetic System 1984", NIMA TR8350.2, January 2000, <http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf>.

Appendix A. Acknowledgments

The authors would like to thank Vince Fuller, Gregg Schudel, Jesper Skriver, Luigi Iannone, Isidor Kouvelas, and Sander Steffann for their technical and editorial commentary.

The authors would like to thank Victor Moreno for discussions that lead to the definition of the Multicast Info LCAF type.

The authors would like to thank Parantap Lahiri and Michael Kowal for discussions that lead to the definition of the Explicit Locator Path (ELP) LCAF type.

The authors would like to thank Fabio Maino and Vina Ermagan for discussions that lead to the definition of the Security Key LCAF type.

The authors would like to thank Albert Cabellos-Aparicio and Florin Coras for discussions that lead to the definition of the Replication List Entry LCAF type.

Thanks goes to Michiel Blokzijl and Alberto Rodriguez-Natal for suggesting new LCAF types.

Thanks also goes to Terry Manderson for assistance obtaining a LISP AFI value from IANA.

Appendix B. Document Change Log

B.1. Changes to draft-ietf-lisp-lcaf-10.txt

- o Submitted June 2015.
- o Fix coauthor Job’s contact information.

B.2. Changes to draft-ietf-lisp-lcaf-09.txt

- o Submitted June 2015.
- o Fix IANA Considerations section to request a registry to allocate and track LCAF Type values.

B.3. Changes to draft-ietf-lisp-lcaf-08.txt

- o Submitted April 2015.
- o Comment from Florin. The Application Data Type length field has a typo. The field should be labeled "12 + n" and not "8 + n".
- o Fix length fields in the sections titled "Using Recursive LISP Canonical Address Encodings", "Generic Database Mapping Lookups", and "Data Model Encoding".

B.4. Changes to draft-ietf-lisp-lcaf-07.txt

- o Submitted December 2014.
- o Add a new LCAF Type called "Encapsulation Format" so decapsulating xTRs can inform encapsulating xTRs what data-plane encapsulations they support.

B.5. Changes to draft-ietf-lisp-lcaf-06.txt

- o Submitted October 2014.
- o Make it clear how sorted RLOC records are done when LCAFs are used as the RLOC record.

B.6. Changes to draft-ietf-lisp-lcaf-05.txt

- o Submitted May 2014.
- o Add a length field of the JSON payload that can be used for either binary or text encoding of JSON data.

B.7. Changes to draft-ietf-lisp-lcaf-04.txt

- o Submitted January 2014.
- o Agreement among ELP implementors to have the AFI 16-bit field adjacent to the address. This will make the encoding consistent with all other LCAF type address encodings.

B.8. Changes to draft-ietf-lisp-lcaf-03.txt

- o Submitted September 2013.
- o Updated references and author’s affilations.
- o Added Instance-ID to the Multicast Info Type so there is relative ease in parsing (S,G) entries within a VPN.
- o Add port range encodings to the Application Data LCAF Type.
- o Add a new JSON LCAF Type.
- o Add Address Key/Value LCAF Type to allow attributes to be attached to an address.

B.9. Changes to draft-ietf-lisp-lcaf-02.txt

- o Submitted March 2013.
- o Added new LCAF Type "Replication List Entry" to support LISP replication engineering use-cases.
- o Changed references to new LISP RFCs.

B.10. Changes to draft-ietf-lisp-lcaf-01.txt

- o Submitted January 2013.
- o Change longitude range from 0-90 to 0-180 in section 4.4.
- o Added reference to WGS-84 in section 4.4.

B.11. Changes to draft-ietf-lisp-lcaf-00.txt

- o Posted first working group draft August 2012.
- o This draft was renamed from draft-farinacci-lisp-lcaf-10.txt.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Dave Meyer
Brocade
San Jose, CA
USA

Email: dmm@1-4-5.net

Job Snijders
NTT Communications
Theodorus Majofskistraat 100
Amsterdam 1065 SZ
NL

Email: job@ntt.net

Locator/ID Separation Protocol
Internet-Draft
Intended status: Experimental
Expires: January 7, 2016

M. Menth
A. Stockmayer
M. Schmidt
University of Tuebingen

July 6, 2015

LISP Hybrid Access
draft-menth-lisp-ha-00

Abstract

Hybrid access (HA) allows simultaneous usage of multiple access links. Advantages are increased bandwidth and improved resilience. This document presents LISP Hybrid Access (LISP-HA), a mechanism to provide HA based on LISP technology. The document discusses potential changes needed to perform dynamic load-balancing and per packet load-balancing, which both increase the efficiency of HA. To that end, modified usage of some fields in the LISP header is proposed. Discussed use cases include the bundling of multiple access technologies for mobile devices and residential access routers. Additionally, we provide some considerations how LISP-HA can be deployed by providers.

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 January 7, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents

(http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Terminology 3
3. LISP-HA-Architecture 4
3.1. Basic Operation 4
3.2. Message Sequence Diagram for Basic Operation. 6
3.3. Policy-Based Path Selection 8
3.4. Operation of an HAxTR behind a NAT 8
3.5. Extensions for Dynamic Load Balancing 10
3.6. Extensions for Packet-Based Load Balancing 11
3.7. Deployment Considerations 12
4. Packet Formats 12
4.1. Dataplane Header 12
4.2. Control Message 13
5. Use Cases 14
5.1. Connecting Residential Users to the General Internet . . 14
5.2. Smartphones with Mobile Node. 15
6. Security Considerations 17
7. Acknowledgements 17
8. References 17
8.1. Normative References 17
8.2. Informative References 17
Authors' Addresses 18

1. Introduction

Hybrid access (HA) enables a device to simultaneously use multiple access links both in upstream and downstream direction. A challenge of HA is to make load balancing of the traffic onto multiple paths transparent to endpoints. HA may be supported on various layers. Multilink PPP (ML-PPP) [RFC1990] offers support for fragmented protocol data units (PDU) on the same local network. Therefore, it cannot combine network layer paths so that it is unable to bundle paths provided by different Internet service providers. A network architecture for HA is presented in [I-D.lhwxz-hybrid-access-network-architecture]. It focuses on bundling DSL and LTE for residential access by means of dedicated customer premises equipment (CPE) which does not support mobile devices in general. Multipath TCP (MP-TCP) leverages multiple paths

on the transport layer [RFC6824]. It requires both source and destination endpoint to support MP-TCP. MP-TCP proxies are currently discussed for inter-operation with non-MP-TCP nodes [I-D.hampel-mptcp-proxies-anchors]. HA can be also supported by MP-TCP, but that approach is limited to TCP traffic. Furthermore, supporting policies such as cheapest link first seems challenging with this approach.

LISP by itself has basic capabilities to support HA with static load balancing that is not agnostic to current link loads and characteristics. That means, a multihomed LISP xTR may perform inbound traffic engineering. This is achieved by setting appropriate weights for multiple RLOCs in register messages so that it receives traffic over more than a single interface. Outbound traffic may be sent over multiple interfaces according to local policies.

This document proposes LISP-HA as a novel solution for HA with improved load balancing capabilities for better resource efficiency.

2. Terminology

Hybrid Access (HA): Using two or more access lines to improve bandwidth and resilience; both technologies are used at the same time.

Mobile Node (MN): A LISP node that includes its own xTR and can connect to more than a single network at a time [I-D.meyer-lisp-mn].

Mapping System (MS): The LISP Mapping System as defined in RFC 6830 [RFC6830].

LISP Tunnel Router (xTR): A combination of ITR and ETR [RFC6830].

LISP Proxy Tunnel Router (PxTR): iUsed to communicate with the legacy Internet [RFC6830].

LISP Reencapsulating Tunnel Router (RTR): LISP router to forward LISP-TE packets [I-D.farinacci-lisp-te].

LISP NAT Traversal Router (NTR): A LISP router that allows to communicate with LISP nodes behind a NAT [I-D.ermagan-lisp-nat-traversal].

LISP Canonical Address Format (LCAF): Extension to the AFI type system to associate the AFI, e.g. with policies [I-D.farinacci-lisp-lcaf].

Explicit Locator Path (ELP): A mapping storing a sequence of RLOCs, indicating RTRs that receive and forward LISP packets to the next RLOC in that list; defined by LISP-TE [I-D.farinacci-lisp-te].

Hybrid Acces xTR (HAxTR): An xTR with multiple RLOCs that supports LISP-HA.

Hybrid Access RTR (HARTR): An RTR that supports LISP-HA.

Hybrid Access Load Balancing (HALB): Function in HAxTR and HARTR that distributes traffic over multiple paths.

Hybrid Access Reorder and Feedback (HARF): Function in HAxTR and HRTR that reorders traffic sent by a corresponding HALB over multiple paths and provides feedback about the link condition to the HALBs.

3. LISP-HA-Architecture

This section describes the basic operation of LISP-HA as well as policy-based path selection, its operation with LISP nodes behind NATs, extensions for dynamic load balancing, and extensions for packet-based load balancing. Finally, we consider how it could be useful for providers to operate their own HARTR.

3.1. Basic Operation

LISP-HA allows to load-balance traffic over multiple paths between a HAxTR and HARTR. This is transparent to nodes not on the path between HAxTR and HARTR. Load-balancing works in both directions, therefore, both the HAxTR and the HARTR implement a HA Load Balancing function (HALB) and a HA Recombination function (HARF).

We present how LISP-HA makes EIDs globally reachable over multiple paths between HARTR and HAxTR. To that end, we consider the setting illustrated in Figure 1.

```
             Mapping System
          +----------------------------------------+
          |  EID |              Entry              |
          -------+--------------------------------+
          | EID0 | ELP: RLOC-HARTR -> RLOC-HAxTR-0 |
          | EID0 | ELP: RLOC-HARTR -> RLOC-HAxTR-1 |
          ----------------------------------------+


                        +------+
                  +-----| Wifi |----+
                  |     +------+    |
                  |                 |
                  |                 |
   +------+   +-----------+     +-----------+           +------+
   | EID0 |---|   HAxTR   |     |   HAR     |--- ... ---| EID1 |
   +------+   | HALB/HARF |     | HALB/HARF |           +------+
              +-----------+     +-----------+
                  |                 |
                  |                 |
                  |     +-----+     |
                  +-----| LTE |-----+
                        +-----+
```

Figure 1: LISP-HA load-balances traffic between HAxTR and HARTR over multiple network layer paths.

A HAxTR is configured with the address of a HARTR and registers its EID prefixes at the MS. To that end, it uses explicit locator paths (ELPs), containing the RLOC of the HARTR in the penultimate positoin of the ELP and one of its own RLOCs in the last position of the ELP. The HAxTR must send one register message for each of its RLOCs and over the interface that corresponds to that RLOC so that the MS can detect whether the HAxTR is behind a NAT.

The HALB functions of the HAxTR and the HARTR distribute the traffic over multiple network layer paths between them. Flow-based or packet-based load-balancing may be supported.

Figure 2 shows a communication scenario between two LISP nodes. The HAxTR is connected to the Internet / LISP net over multiple access technologies and LISP-HA is applied between HAxTR and HARTR. The endpoints EID0 and EID1 exchange messages over the HAxTR, the HARTR, and the xTR. The figure shows the destination EIDs and RLOCs of these messages. The HAxTR/xTR add RLOCs to the messages through encapsulation, the HARTR exchanges the RLOCs, and the xTR/HAxTR

remove RLOCs from the messages through decapsulation. The HAxTR and xTR add RLOCs to the messages through encapsulation and the HARTR exchanges the RLOCs. In upstream direction, the HAxTR adds the RLOC of the HARTR and selects the path by choosing the appropriate interface. The HARTR exchanges its own RLOC by the RLOC of the xTR. In downstream direction, the xTR adds the RLOC of the HARTR as specified in the ELP. The HARTR exchanges this RLOC with the appropriate RLOC of the HAxTR. Thereby the desired path is selected.

Upstream example:

```
EID0  --->  HAxTR  --->   HARTR   --->   xTR  --->   EID1

      EID1         EID1           EID1        EID1
                RLOC-HARTR       RLOC-xTR
```

Downstream example:

```
EID0  <---  HAxTR  <---   HARTR   <---   xTR  <---   EID1

      EID0         EID0           EID0        EID0
               RLOC-HAxTR-i    RLOC-HARTR
```

Figure 2: Destination EIDs and RLOCs of a LISP packet between two communicating endpoints.

3.2. Message Sequence Diagram for Basic Operation.

Figure 3 and Figure 4 illustrate the basic operation of LISP-HA by a diagram showing all control and data messages exchanged to send data in upstream and downstream direction. In both cases we assume that the HAxTR and the xTR have registered their EIDsi EID0 and EID1 at the MS, but the required mappings are not available in caches.

When the endhost with EID0 starts sending data to EID1, it forwards them to its HAxTR. The HAxTR requests the mapping for RLOC of EID1 from the MS to verify that EID1 is globally reachble before sending it to the HAxTR. The HAxTR encapsulates the packet with the configured address of the HARTR as destination RLOC, using the access line selected by its HALB. Upon receipt of the packet, the HARTR also requests the mapping for EID1, exchnages the destination RLOC in the packet with the RLOC of the xTR provided in the map-reply and sends the packet to the xTR. Upon receipt, the xTR decapsulates the LISP packet and forwards it to the endhost with EID1.

```
EID0        HAxTR        MS        HARTR        xTR        EID1
      data
  ----------->
              request for EID1
                ---------->
              reply containing RLOC-xTR
                <----------
        LISP data with dst:EID1 / RLOC-HARTR
                --------------------->
                          request for EID1
                           <----------
                    reply containing RLOC-xTR
                           ---------->
                       LISP data dst:EID1/RLOC-xTR
                                      ----------->
                                                        data
                                                  ---------->
```

Figure 3: Message sequence diagram for upstream communication.

When endhost with EID1 starts sending data to EID0, it forwards them to its xTR. The HAxTR requests the mapping for EID0 from the MS, receives the ELP for EID0, encapsulates the packet with RLOC-HARTR, and sends it to the HARTR. Upon receipt of the packet, the HARTR requests the mapping for EID0 from the MS. The HALB function of the HARTR selects the path to the HAxTR, the HARTR exchanges the destination RLOC in the LISP packet with RLOC-HAxTR and sends the packet over the determined path. Upon receipt, the HAxTR decapsulates the LISP packet and forwards it to the endhost with EID0.

```
   EID0        HAxTR        MS        HARTR        xTR        EID1
                                                        data
                                                <----------
                                  request for EID0
                              <-----------------
            reply containing ELP:RLOC-HAxTR-i -> HARTR
                             ----------------->
                                   LISP data with dst:EID0/RLOC-HARTR
                                        <-----------
                              request for EID0
                             <----------
                            reply containing ELP:RLOC-HAxTR-i -> HARTR
                             ---------->
              LISP data with dst:EID0/RLOC-HAxTR-i
                <---------------------
        data
    <------------
```

Figure 4: Message sequence diagram for downstream communication.

3.3. Policy-Based Path Selection

For specific kinds of traffic, e.g., for realtime communication, the usage of specific paths may be desired. LISP-HA supports such requirements through LISP LCAF extensions both on upstream and downstream. To that end, the HAxTR is configured with LCAF extensions, e.g., indicating that traffic for realtime communication must be forwarded over a specific path. The HAxTR uses this LCAF as local policy to encapsulate the traffic. In addition, Tthe HAxTR registers the same LCAF at the MS. As a consequence, xTRs encapsulate traffic towards the HAxTR using the specific RLOC in the LCAF.

3.4. Operation of an HAxTR behind a NAT

A HAxTR may be located behind a NAT. We consider this case as this scenario is common for HAxTRs connected via LTE.

```
             Mapping System
         +------+--------------------------------+
         | EID  |              Entry             |
         +------+--------------------------------+
         | EID0 | ELP: RLOC-HARTR -> RLOC-HAxTR-0 |
         | EID0 | ELP: RLOC-HARTR -> RLOC-NTR     |
         +------+--------------------------------+


                       +------+
                  +-----| Wifi |-----------+
                  |     +------+           |
                  |                        |
                  |                        |
   +------+   +-----------+            +-----------+           +------+
   | EID0 |---|   HAxTR   |            |   HARTR   |--- ... ---| EID1 |
   +------+   | HALB/HARF |            | HALB/HARF |           +------+
              +-----------+            +-----------+
                  |                        |
                  |                        |
                  |     +-----------+   +-----+
                  +-----| LTE | NAT |---| NTR |
                        +-----------+   +-----+
```

Figure 5: LISP-HA load-balances traffic between HAxTR and HARTRT while part of the traffic traverses a NAT.

We show how LISP-HA leverages existing LISP NAT traversal [I-D.ermagan-lisp-nat-traversal] so that it does not require additional mechanisms to cope with NATs.

Figure 5 shows a HAxTR that is connected to the Internet / LISP net via LTE and through a NAT. The HAxTR registers its RLOC at the MS, the MS detects that the RLOC in the LISP register message differs from the RLOC in the outer header encapsulating it. As a consequence, the MS does not register the mapping and informs the HAxTR proposing a list of potential NAT Traversal Routers (NTRs). Then, the HAxTR selects one of the NTRs and registers again at the MS via this NTR. The NTR exchanges the RLOC in the mapping by its own RLOC (RLOC-NTR). As a result, traffic for the HAxTR is directed to the NTR which forwards it to the HAxTR. This mechanism works for LISP-HA only if the HAxTR registers its ELPs over the corresponding interfaces; otherwise the MS cannot securely detect that the HAxTR is behind a NAT.

The usage of general NTRs leads to triangle routing and adds significant delay which may be prohibitive. However, an NTR may be combined with the HARTR and configured with the HAxTR so that triangle routing and added path delay are avoided.

To cope with carrier grade NATs, messages need to be exchanged frequently enough over the NTR to avoid that NAT table entries are removed. This, however, is to be fixed for the NTR draft [I-D.ermagan-lisp-nat-traversal]. Moreover, HAxTR and HARTR exchange feedback messages for dynamic load balancing frequently enough so that NAT table entries will not be removed.

3.5. Extensions for Dynamic Load Balancing

One goal of HA is to increase a HAxTR’s overall access bandwidth by combining the bandwidth of all available paths. However, static load balancing leads to statistical variations [Menth06p] so that some paths are already overloaded while others are underutilized. Dynamic load balancing takes the current load on the links into account and can achieve better resource utilization without overloading paths.

We propose dynamic load balancing for LISP-HA with the purpose to increase resource efficiency, thereby providing higher bandwidth to the user. A challenge is that path properties like available bandwidth and delay are possibly unknown and vary over time, especially if the path is shared and includes wireless links. Also flow rates vary over time and the rate of elastic flows depends on path characteristics. Nevertheless, incipient congestion can be inferred from increasing path-specifc packet loss and delay. So the idea is to obtain feedback about path-specific packet loss and delay and leverage this information for improved load balancing. To that end, the HARF functions continuously monitor the quality of all paths perceived by transmitted traffic so that the HALB can leverage path-specific information about packet loss and delay for load balancing decisions. In the following we briefly explain how a pair of corresponding HALB and HARF functions on a one-directional path can obtain information about packet loss and delay.

To estimate packet loss, the HALB function equips the overall packet stream with sequence numbers before load-balancing them over multiple interfaces. These sequence numbers are also used for packet reordering if needed.

The HALB function counts the number of packets sent per path up to some checkpoint sequence number. The corresponding HARF function counts the number of packets received per path up to the same checkpoint sequence numbers and reports them to the corresponding HALB. The difference between the number of sent and received packets

between checkpoint sequence numbers gives an estimate about packet loss per path.

Measuring packet delay between HALB and HARF is rather difficult as their clocks may not be synchronized and have a drift. Therefore, only relative delays are measured over time.

The HALB equips packets with timestamps before sending them to an interface and the HARF computes the difference to its current time upon receipt, yielding a biased packet delay due to potentially missing clock synchronization. Assuming that all biased packet delays have the same bias, the evolution of the biased packet delay reflects the evolution of the real packet delay. The HARF reports to the HALB the path-specific numbers of recently received packets and delay information. Delay metrics may be minimum, maximum, average delay as well as a standard deviation. However, the metrics are not clear, yet, because the load-balancing algorithm is still under research. The same holds for the frequency of checkpoint sequence numbers and the frequency of feedback reports from the HARF to the HALB.

3.6. Extensions for Packet-Based Load Balancing

Per-flow load balancing forwards packets of a flow over the same path. Therefore, packets will arrive in order unless they are reordered for other reasons on the path. Thus, per-flow load balancing avoids additional packet reordering by LISP-HA. However, it is more difficult to efficiently use the bandwidth of existing paths with per-flow load-balancing than with per-packet load balancing. Moreover, as flow-based load balancing forwards packets of a single flow only over a single path, very large flows cannot profit from several paths.

Packet-based load balancing may cause reordered packets in particular if paths have different delay. As packet reordering has negative impact on some transport protocols and applications, it should either be avoided or packets should be reordered. We propose that the HARF function performs packet reordering if needed so that the HALB function can load-balance per packet if desired. However, packet reordering causes some additional delay leading to a tradeoff between reordered packets and increased delay.

A HALB may be configured to load-balance certain flows per flow and other flows per packet, depending on the needs of specific transport protocols or applications. In addition, the HARC may reorder packets from per-packet load balanced flows into order. To that end, such packets need to be marked with a "Reorder" flag so that other packets do not suffer from reordering delay.

The HARF leverages the sequence numbers of all packets for packet reordering. It holds back packets which require reordering before forwarding so long that it is unlikely that packets from the same flow with lower sequence numbers will be received. Details are subject to future work.

3.7. Deployment Considerations

HARTRs are infrastructure that need to be operated and cause costs in the first place. However, the operation of a HARTR allows to perform traffic engineering by appropriate load balancing. E.g., an LTE network operator prefers to offload its resources and can achieve that with the HARTR if other paths have sufficient capacity. Thus, ISPs may have interest to set up HARTRs to have strategic control over traffic forwarding in the network.

Apart from that, offering a HARTR as a service by a third party for a small fee may also be an option. Thereby, customers could book cheap contracts for LTE and residential access and combine these technologies via the third party HARTR.

4. Packet Formats

A modified LISP dataplane header is presented to convey information from HALB to HARF and a new LISP control message is proposed to report feedback information from HARF to HALB.

4.1. Dataplane Header

LISP-HA reuses some fields of the dataplane header of encapsulated LISP data packets to convey information for dynamic load balancing from the HALB to the HARF. The original dataplane header is illustrated in Figure 6. The usage of the fields is defined in [RFC6830].

```
 x x x x 1 0 0 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|N|L|E|V|I|flags|            Nonce/Map-Version                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Instance ID                   |     LSBs      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Original dataplane header.

The Nonce was proposed for security purpose, but has no application in the LISP-HA context. An Instance ID is used to uniquely identify

the source in a virtual address space, which is neither applicable in the LISP-HA context. Three header flags are unused. Therefore we propose to reuse the Nonce and the Instance ID fields of the original dataplane header definition to convey sequence numbers and timestamps from the HALB to the HARF together with an indication whether a packet needs to be forwarded in-order. The modified LISP header is illustrated in Figure 7. The modified header fields are explained in the following.

```
 0 x 0 x 0 x 0 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|N|L|E|V|I|R|flg|              Sequence number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Timestamp                    |     LSBs      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Modified LISP Dataplane Header.

R: Reorder flag to indicate whether a packet needs to be forwarded in-order. It is 1 if the packet has to be reordered, and it is 0 if the packet can be forwarded immediately.

Sequence number: Sequence number for packets sent from HALB to HARF, used for packet loss detection and reordering.

Timestamp: The lower 24 bits of the timestamp of the sender.

All other fields: All fields not explicitly described here have the same meaning as in [RFC6830].

In upstream direction the HAxTR sends packet with the modified header to the HARTR. The HARTR modifies the modified header in upstream direction to be compliant to [RFC6830]. In downstream direction the HARTR receives LISP packets with a header format com,pliant to [RFC6830] and modifies the header as proposed in this section. The HAxTR removes this header. NTRs may be located between HAxTR and HARTR, but they do not need to process the modified header fields. Therefore, only HAxTR and HARTR need to implement, understand, and process the modified header format.

4.2. Control Message

We propose the unused Type number 5 for LISP-HA Feedback Messages to report feedback about packet loss and delay from the HARF to the HALB. These messages contain information about the number of received packets between sequence number checkpoint and information

about packet delay as described in Section 3.5. The exact values and format is are subject to further research.

5. Use Cases

In the following, we describe two typical use cases of LISP-HA. The first use case explains how LISP-HA can be used for residential users to benefit from HA to connect to the Internet. In the second use case, we present how a mobile node, e.g. a smartphone, can use LISP-HA.

5.1. Connecting Residential Users to the General Internet

For customers with cable internet high bandwidth can not be guaranteed as cable internet is based on a shared medium. Especially in the evening hours when many customers need bandwidth at the same time, the rate can drop significantly. For those customers it would be a great benefit if they could bundle their, sometimes slow, cable access with LTE to improve their bandwidth. Figure 8 illustrates a potential solution using LISP-HA.

The HAxTR may be implemented in the home router, it has a public EID which it registers at the MS. The customer typically uses a private address space in his home LAN which is connected through the NAT of the home router. The HAxTR is connected to the Internet through a DSL and an LTE interface and there is a provider NAT on the LTE path. An NTR is used to make the HAxTR reachable vial LTE from the Internet. As LISP nodes cannot communicate directly with the Internet, the HAxTR is configured with an appropriate PxTR, to send traffic to non-LISP IP addresses. To minimize path stretch and delay, both NTR and PxTR may be integrated in the same box as the HARTR.

```
+------+---------------------------------+
| EID  |              Entry              |
+------+---------------------------------+
| EID0 | ELP: RLOC-HARTR -> RLOC-HAxTR-0 |
| EID1 | ELP: RLOC-HARTR -> RLOC-NTR     |
+------+---------------------------------+
```

```
                    +----------+
                    |  cable   |
                +---| internet |-------+
                |   +----------+       |
                |                      |
                |                      |
---      +-------------+               |                       --------
   \     | Home Router |               |                      /
    \    +-------------+         +-----------+   +------+    /
 LAN |---|    (NAT)    |         |   HARTR   |---| PxTR |---| Internet
    /    +------------ +         | HALB/HARF |   +------+    \
   /     |    HAxTR    |         +-----------+                \
---      |  HALB/HARF  |               |                       --------
         +-------------+               |
                |                      |
                |                      |
                |   +-----------+   +-----+
                +---| LTE | NAT |---| NTR |
                    +-----------+   +-----+
```

Figure 8: LISP-HA used for residential access to the Internet.

This solution may be attractive to users who are not even aware of LISP. Their traffic is just load-balanced over DSL and LTE between the home router and the HARTR and decapsulated by the PxTR. In case of a public address space in the customer’s LAN the HAxTR can register the entire subnet of the LAN as EID prefix at the MS. The PxTR has to announce the EIDs registered by the HAxTR to the Internet so that traffic for the HAxTR is sent to PxTR.

5.2. Smartphones with Mobile Node.

Today’s smartphones include multiple radio interfaces that allow to connect to multiple access technologies like LTE and Wifi at he same time. The default policy that is implemented in smartphones is to offload traffic from the cellular network to Wifi access. However, it would be desireable to use both technologies to increase the available bandwidth for normal internet traffic like downloads and to select the Wifi connection for realtime applications like VoIP. In

Figure 9, we present a potential setup how a MN can use LISP-HA to realize the scenario.

```
            Mapping System
         +------+--------------------------------+
         | EID  |              Entry             |
         +------+--------------------------------+
         | EID0 | ELP: RLOC-HARTR -> RLOC-NTR-0  |
         | EID0 | ELP: RLOC-HARTR -> RLOC-NTR-1  |
         | EID0 | LCAF: VoIP use RLOC-NTR-0      |
         +------+--------------------------------+


           +------+-----+   +-----+   +-----+
        +--| Wifi | NAT |---| DSL |---| NTR |
        |  +------+-----+   +-----+   +-----+
        |                                |
  +-----------+                          |                      --------
  |    MN     |                          |                     /
  +-----------+                   +-----------+   +------+    /
  |   HAxTR   |                   |   HARTR   |---| PxTR |---| Internet
  | HALB/HARF |                   | HALB/HARF |   +------+    \
  +-----------+                   +-----------+                \
        |                                |                      --------
        |                                |
        |  +-----------+   +-----+       |
        +--| LTE | NAT |---| NTR |-------+
           +-----------+   +-----+
```

Figure 9: LISP-HA used for Smartphones with Mobile Node.

To realize HA on a MN, the MN has to implement its own HAxTR consisting of HALB and HARF. Typically, the MN has access to the Internet most of the time using cellular networks like LTE or HSDPA. So the MN registers its EID through the cellular network at the MS. A provider NAT in the LTE network is handled like described in Section 3.4. Wifi access becomes available if the MN is in reach of a public Wifi hotspot, in the home LAN of the user or other known Wifi networks. Once Wifi access is available, the MN registers its EID over Wifi, too. Additionally, the MN can register an LCAF for VoIP traffic to use Wifi. If Wifi is available no more because the MN left the Wifi cell, the MN should de-register the Wifi mappings at the MS.

6. Security Considerations

HA by itself does not raise any security concerns. However, LISP-HA is based on LISP so that the same security considerations apply as described in [RFC6830]. There is no authentication of endhosts at the xTR and no authentication between xTRs which allows every node to send any amount of traffic to any xTR which makes it vulnerable to DOS attacks. This also counts for the HARTR and the HAxTR. LISP traffic is not encrypted, so if it is required to encrypt the communication, this has to be realized by the endhosts.

7. Acknowledgements

The authors would like to thank Gerd Pflueger and Wilhelm Boeddinghaus for their helpful input and discussions.

8. References

8.1. Normative References

[RFC1990] Sklower, K., Lloyd, B., McGregor, G., Carr, D., and T. Coradetti, "The PPP Multilink Protocol (MP)", RFC 1990, August 1996.

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

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013.

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

8.2. Informative References

[I-D.ermagan-lisp-nat-traversal]
Ermagan, V., Farinacci, D., Lewis, D., Skriver, J., Maino, F., and C. White, "NAT traversal for LISP", draft-ermagan-lisp-nat-traversal-07 (work in progress), February 2015.

[I-D.farinacci-lisp-lcaf]
Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", draft-farinacci-lisp-lcaf-10 (work in progress), July 2012.

[I-D.farinacci-lisp-te]
Farinacci, D., Kowal, M., and P. Lahiri, "LISP Traffic Engineering Use-Cases", draft-farinacci-lisp-te-08 (work in progress), March 2015.

[I-D.hampel-mptcp-proxies-anchors]
Hampel, G. and T. Klein, "MPTCP Proxies and Anchors", draft-hampel-mptcp-proxies-anchors-00 (work in progress), February 2012.

[I-D.lhwxz-hybrid-access-network-architecture]
Leymann, N., Heidemann, C., Wasserman, M., Xue, L., and M. Zhang, "Hybrid Access Network Architecture", draft-lhwxz-hybrid-access-network-architecture-02 (work in progress), January 2015.

[I-D.meyer-lisp-mn]
Farinacci, D., Lewis, D., Meyer, D., and C. White, "LISP Mobile Node", draft-meyer-lisp-mn-12 (work in progress), January 2015.

[Menth06p]
Milbrandt, J., Humm, K., and M. Menth, "Adaptive Bandwidth Allocation: Impact of Routing and Load Balancing on Tunnel Capacity Requirements", in Proceedings of 2nd Conference on Next Generation Internet Design and Engineering, Valencia, Spain , April 2006.

Authors' Addresses

Michael Menth
University of Tuebingen
room B302, Institute of Computer Science
Sand 13
Tuebingen 72076
Germany

Phone: +49 7071 29-70505
Email: menth@uni-tuebingen.de

Andreas Stockmayer
University of Tuebingen
room B305, Institute of Computer Science
Sand 13
Tuebingen 72076
Germany

Phone: +49 7071 29-70511
Email: andreas.stockmayer@uni-tuebingen.de

Mark Schmidt
University of Tuebingen
room B305, Institute of Computer Science
Sand 13
Tuebingen 72076
Germany

Phone: +49 7071 29-70510
Email: mark-thomas.schmidt@uni-tuebingen.de

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: January 6, 2016

A. Rodriguez-Natal
A. Cabellos-Aparicio
Technical University of Catalonia
S. Barkai
ConteXtream, Inc.
V. Ermagan
D. Lewis
F. Maino
Cisco Systems
D. Farinacci
lispers.net
July 5, 2015

LISP support for Multi-Tuple EIDs
draft-rodrigueznatal-lisp-multi-tuple-eids-00

Abstract

This document describes extensions for LISP to support EIDs based on tuples of multiple elements.

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 January 6, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
1.1. Requirements Language 2
2. Definition of terms 3
3. Overview 3
4. Protocol Operation with Extended-EIDs 4
4.1. LISP Tunnel Routers 4
4.2. Mapping System 4
5. Extended-EIDs Encoding 5
5.1. 5-Tuple 5
6. Extended-EID Lookups 5
6.1. 5-Tuple (Coarse) 5
6.2. 5-Tuple (Exact) 6
7. Mapping Databases for Extended-EIDs 6
7.1. 5-Tuple (Coarse) 7
7.2. 5-Tuple (Exact) 7
8. A Note on Instance-ID 7
9. Acknowledgments 7
10. IANA Considerations 7
11. Security Considerations 7
12. Normative References 7
Authors' Addresses 8

1. Introduction

This document describes how LISP handles lookups based on Extended-EIDs, i.e. tuples of n elements. Particularly it describes how the Tunnel Routers and the Mapping System operate when Extended-EIDs are in place, the different types of Extended-EIDs defined so far, how the lookup is performed for each Extended-EID type and which mapping databases are recommended to use depending on the kind of Extended-EIDs used.

1.1. 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 RFC 2119 [RFC2119].

2. Definition of terms

- o n-tuple: The term n-tuple is used in this document to describe the set of n elements present in a data packet (e.g. IP address, port, protocol) that can be used to identify unequivocally a packet or a set of packets.

- o 5-tuple: The term 5-tuple is used in this document to describe the set comprised by 5 elements, being these the source IP address, the destination IP address, the level 4 protocol number, the level 4 protocol source port and the level 4 protocol destination port of a data packet.

- o Extended-EID: This document uses the term Extended-EID to refer to any n-tuple (including a 5-tuple) used in a EID role. See as well [I-D.ietf-lisp-ddt]

- o Flow: The term flow is used in this document to refer to the sequence of packets identified by the same n-tuple.

- o MT-[xTR, RTR, MS, MR]: A LISP [xTR, RTR, MS, MR] that supports the enhanced operation for Multi-Tuple Extended-EIDs described in this document.

- o MT-TR: A LISP tunnel router (e.g. xTR, RTR) that supports the enhanced operation for Multi-Tuple Extended-EIDs described in this document, e.g. MT-xTR, MT-RTR.

The rest of the terms are defined in their respective documents. See the LISP specification [RFC6830] for most of the definitions, [I-D.ietf-lisp-lcaf] for LCAF and [I-D.farinacci-lisp-te] for RTR.

3. Overview

This document describes extensions for LISP to support Multi-Tuple Extended-EIDs. Protocol operation follows the specification defined on [RFC6830] except for the following. Besides of IP mappings, a Mapping System can store Extended-EID mappings. Being Extended-EID a n-tuple identifying a flow. LISP routers perform look-ups based on these Extended-EIDs, instead of on destination IPs. Apart from using n-tuples instead of IPs, retrieving information from the Mapping System follows LISP standard mechanisms (i.e. Map-Request, Map-Reply).

4. Protocol Operation with Extended-EIDs

4.1. LISP Tunnel Routers

LISP tunnel routers with enhanced operation for Multi-Tuple Extended-EIDs, or MT-TRs (MT-xTRs and MT-RTRs), behave as specified on [RFC6830] and [RFC6833], with the particularity that MT-TRs perform mapping lookups based on Extended-EIDs (n-tuples).

Any MT-TR must keep an internal map-cache indexed by Extended-EIDs. When a MT-TR receives a packet to encapsulate, it extracts the fields required by the n-tuple lookup in use and stores them in an Extended-EID structure. In the case of a 5-tuple lookup, it will extract the source address, destination address, level 4 protocol, source port (if any) and destination port (if any) from the packet. The MT-TR uses the Extended-EID to perform a look-up into the map-cache. The lookup process must follow the procedure described in section Section 6. If there is an entry on the map-cache that matches the Extended-EID, the MT-TR retrieves the mapping information, selects a destination RLOC and encapsulates the packet as defined in [RFC6830]

If the map-cache of the MT-TR contains no entry for the Extended-EID, the MT-TR sends a Map-Request to a MT-MR. The MT-TR MUST be provisioned with the RLOC of at least one MT-MR. The Map-Request sent carries the Extended-EID (encoded in the specific LCAF for that Extended-EID type) in the EID-prefix field of the Map-Request. This Map-Request will eventually trigger a Map-Reply to be sent back the requester MT-TR, see section Section 4.2. This Map-Reply carries an Extended-EID on the EID-prefix field. The MT-TR stores, as defined in [RFC6830], the mapping for the Extended-EID.

4.2. Mapping System

Mapping System elements (comprising Map Servers and Map Resolvers) behave as specified on [RFC6830] and [RFC6833] when implementing enhanced Multi-Tuple Extended-EIDs operation, with the particularity that MT-MRs resolve Map-Requests based on Extended-EIDs and MT-MSs store mappings indexed by Extended-EIDs.

MT-MRs must be capable of processing Map-Requests with an Extended-EID on the EID-prefix field, of finding the appropriate MT-MS for the Extended-EID and of forwarding the Map-Request to it. This is done according to the lookup rules described in section Section 6 and using the mapping database described in section Section 7 which differs depending on the specific Extended-EID.

LISP elements must perform the mapping update mechanisms defined in [RFC6830] (e.g, SMR) using as EID the Extended-EID.

5. Extended-EIDs Encoding

This section describes the Extended-EID types defined so far and the LCAFs to support them.

5.1. 5-Tuple

The 5-tuple LCAF is a combination of Application Data Type 4 and Source/Dest Type 12 LCAFs. Experimental deployment may indicate that a specific 5-tuple type LCAF is necessary.

6. Extended-EID Lookups

This section describes the lookup process to be followed when using Extended-EID. At this point, this document only covers 5-tuple kind of Extended-EID lookups (with options for coarse or exact lookup). It is expected to include lookup mechanism for n-tuple lookups with more complex protocol combinations.

6.1. 5-Tuple (Coarse)

When using a coarse lookup for 5-tuple, the encoding described in Section 5.1 is used to carry the Extended-EID. Note that a coarse lookup also covers exact lookups. The lookup is (logically) done at steps, one per each element of the tuple. The lookup MUST follow this strict order:

(1) Destination address

(2) Source address

(3) Protocol number

(4) Destination port

(5) Source port

This means that for a given 5-tuple, the lookup process will first select from the available 5-tuples present in the system, the ones that match the destination address. Among them, those that also match the source address. This is iterated for the rest of the elements in the tuple. If a 5-tuple matches several entries, then the one with the longest prefix match or shortest port-range has priority. To clarify the process an example is provided below.

Suppose that a MT-MS stores the mappings indexed by the tuples (A), (B), (C) and (D). A Map-Request carrying the Extended-EID (T) is received by the MT-MS. The Extended-EID (T) can match both (A) and

(B) entries, however destination address has preference over source address and therefore entry (A) is the one selected. If the Map-Request carries (U) instead of (T), the lookup will return that no entry exists for the 5-tuple, however if it carries (V), the lookup will return the (B) entry. On the other hand, a lookup for (W) will return (C) and not (D), although both match the tuple, since the destination port range is shorter in (C). In the case of (X) the lookup will return (D).

```
              dst-add     src-add    pr   dst-prt    src-prt


        (A) [1.1.1.0/24, 2.2.0.0/16, 17, 1000-3000, 1000-3000]
        (B) [1.1.0.0/16, 2.2.2.0/24, 17, 1000-3000, 1000-3000]
        (C) [3.3.3.0/24, 4.4.4.0/24, 17, 4000-4500, 7000-8000]
        (D) [3.3.3.0/24, 4.4.4.0/24, 17, 4000-6000, 7000-8000]



        (T) [ 1.1.1.8,     2.2.2.9,  17,   2000,      2000   ]
        (U) [ 1.1.8.8,     2.2.9.9,  17,   2000,      2000   ]
        (V) [ 1.1.8.8,     2.2.2.9,  17,   2000,      2000   ]
        (W) [ 3.3.3.3,     4.4.4.4,   6,   4300,      7500   ]
        (X) [ 3.3.3.3,     4.4.4.4,   6,   5000,      7500   ]
```

5-tuple example for coarse lookup

6.2. 5-Tuple (Exact)

In scenarios where 5-tuple coarse lookup is not required, the lookup can be optimized to only account for exact matchs. When using a exact lookup for 5-tuple, the encoding described in Section 6.1 is used to carry the Extended-EID. The exact match lookup is performed by serializing the elements of the 5-tuple as a single vector of bits. The order to serialize the elements is the same that is described in Section 5.1 This (unique) vector of bits is then used as the key to perform a exact match lookup over the available entries.

7. Mapping Databases for Extended-EIDs

The mapping database, i.e. the system to interconnect (MT-)MRs and (MT-)MSs should be optimal for each one of the different Extended-EIDs types. This section covers recommended mapping databases for each of the Extended-EIDs described in this document.

7.1. 5-Tuple (Coarse)

The mapping database to be used for a coarse lookup of 5-tuples can leverage on the LISP-DDT mapping database [I-D.ietf-lisp-ddt] since it supports multi-tuple lookups. Note that a LISP-DDT based database can support also a exact lookup.

7.2. 5-Tuple (Exact)

Although a LISP-DDT based mapping database supports both coarse and exact lookups, the particularities of the latter benefit of using a mapping database optimized for flat namespaces rather than one optimized for hierarchical data. In that sense, the exact match mechanism should be supported by a DHT-like mapping database, such [I-D.cheng-lisp-shdht] or [LISP-DHT].

8. A Note on Instance-ID

Instance-ID is a special case to be considered. If it is in use, its lookup is resolved before the lookup for the Extended-EID begins (regardless of the Extended-EID type). In terms of implementation this means that if the Instance-ID is present, it will have always more priority that any other field within the multi-tuple EID. In other words, Instance-ID is the high-order parts of the destination and source addresses and a longest match lookup should be applied to it before looking up the address itself.

9. Acknowledgments

10. IANA Considerations

This memo includes no request to IANA.

11. Security Considerations

Security Considerations TBD

12. Normative References

[I-D.cheng-lisp-shdht]
Cheng, L. and J. Wang, "LISP Single-Hop DHT Mapping Overlay", draft-cheng-lisp-shdht-04 (work in progress), July 2013.

[I-D.farinacci-lisp-te]
Farinacci, D., Kowal, M., and P. Lahiri, "LISP Traffic Engineering Use-Cases", draft-farinacci-lisp-te-08 (work in progress), March 2015.

[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-lcaf]
Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", draft-ietf-lisp-lcaf-10 (work in progress), June 2015.

[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.

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

Authors' Addresses

Alberto Rodriguez-Natal
Technical University of Catalonia
Barcelona
Spain

Email: arnatal@ac.upc.edu

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain

Email: acabello@ac.upc.edu

Sharon Barkai
ConteXtream, Inc.
Mountain View, CA
USA

Email: sbarkai@gmail.com

Vina Ermagan
Cisco Systems
170 Tasman Drive
San Jose, CA
USA

Email: vermagan@cisco.com

Darrel Lewis
Cisco Systems
170 Tasman Drive
San Jose, CA
USA

Email: darlewis@cisco.com

Fabio Maino
Cisco Systems
170 Tasman Drive
San Jose, CA
USA

Email: fmaino@cisco.com

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com