# Control-Plane Implementation report

## 1. Introduction

Open Lisp control plane implements the control-plane function of LISP (Locator/Identifier Separation Protocol). The basic idea of LISP is to replace IP addresses with two new types of numbers:

- Routing Locators (RLOCs), which are topologically assigned to network attachment points and used for routing and forwarding of packets through the network;
- Endpoint Identifiers (EIDs) which are assigned independently from the network topology, are used for numbering devices, and are aggregated along administrative boundaries [IETF-LISP-Draft].

During the packet exchange between the two hosts, at the beginning, the source issues a normal IP package with source and destination IP are EIDs. When the package reach one of border routers of its domain (ITR), ITR lookup for EID-to-RLOC mapping and then it append a new LISP header to the package (encapsulation processing) before forward it. In the new header (outer header), source and destination IP are RLOCs. The new package is routed over core network until it reaches the border router of destination host (ETR). At this time, ETR strips the outer header (decapsulation processing) and deliver the package to the destination host. The eventual reply of the host follows the same steps.

Lisp control-plane implements the function of mapping between the two spaces numbering. Our purpose is the provision a control-plane support in FreeBSD. It implements two functions, which are map-register and map-reply. The first one is register xTR's EID-to-RLOC mapping database with map-server. The second function allows xTR response with map-request. In this document, we will describe the architecture of our implement.

For terms used in this document (e.g., xTR, MS, MR), please refer to the original LISP specification.

## 2. Configure file

To implement control-plane, the xTR (IRT or ERT) need to know about mapping between the EIDs and RLOCs that it manages, which map-servers to send register message. That information is kept in configure file. The configuration file includes one section for map-server and one or more sections for EID-to-RLOC database. Each sections start by a control keyword: @MAPSERVER for map-server section and @EID for EID-to-RLOC section.

Map-server section contains a list of map-server (ip address or domain name) with Key which xTR use to authenticate.

E.g:

```
@MAPSERVER
193.162.145.50                                  lip6-fr-xtr
l3-london-mr-ms.rloc.lisp4.net                  lip6-fr-xtr
2001:67c:21b4:109::b                            lip6-fr-xtr
```

Each EID-to-RLOC section include mapping between one EID and RLOC(s). The first line of section contains EID's information: EID-prefix, Subnetmask, Time-To-Live (TTL) and Flag. Flag is used to indicate if EID-prefix is registered with map-server or not. Other lines are RLOC's information: RLOC address, Priority, Weight and Local flag. Please refer to the original LISP specification [section 6] for more detail about terms.

**E.g:**

```
@EID
#Eid-Prefix         Subnetmask   TTL    Flag
153.16.38.0         25           60     1
#RLOC               Priority     Weight       Local flag
132.227.62.242      2            100          1
132.227.62.243      1            100          0
2001:660:240::242   5            100          1
2001:660:240::243   3            100          0
```

@EID

| #Eid-Prefix | Subnetmask | TTL | Flag |
|---|---|---|---|
| 2610:D0:2121:: | 48 | 60 | 1 |

| #RLOC | Priority | Weight | Local flag |
|---|---|---|---|
| 132.227.62.242 | 2 | 100 | 1 |
| 132.227.62.243 | 1 | 100 | 0 |
| 2001:660:240::242 | 5 | 100 | 1 |
| 2001:660:240::243 | 3 | 100 | 0 |

## 3. Thread

At the beginning, this implementation includes three threads. One is register-thread, it periodically sends map-register message to map-servers. Two other ones are ipv4-reply-thread and ipv6-reply-thread; they receive map-request message and response with map-reply message. One works with IPv4 and the other works with IPv6. Note that two reply-thread share two sockets (one for ipv4 and one for ipv6) because it must accept Encapsulated Control Message (ECM), which contain map-request message. ECM maybe has inner header in IPv4 and outer header in IPv6 or vice versa.

For the messages format and how thread working, please refer to the original LISP specification [section 6].

## 4. Debug

During running, the program prints debug information to console. With control-plane message, the debug information exists in two forms: raw package format (in hexa) and human readable format.

**E.g:**

Raw package:

```
Map-Register Packet
==========================
0x3000000459f066a12bab660c00010014a3e543f64e202605dbf4e7c2a2215fdc8083b6260
000003c04181000000000019910260004 64ff000005000184e33ef20264ff000001000184e33ef30564
ff00000500022001066033022 82c0240000000002420364ff00000100022001066033022 82c0240000
```

```
00000024300000003c0419100000000001991026000264ff000005000184e33ef20164ff000001000184
e33ef30564ff00000500022001066033022 82c024000000000024203 64ff00000100022001066033022
82c0240000000000243000000 3c0419100000000001991026800164ff000005000184e33ef20264ff00
0001000184e33ef30564ff00000500022001066033022 82c024000000000024203 64ff0000010002200
106603302282c0240000000000243000000 3c0430100000000002261000d0212100000000000000000
000264ff000005000184e33ef20164ff000001000184e33ef30564ff00000500022001066033022 82c0
2400000000002420364ff00000100022001066033022 82c0240000000000243
```

Human package

```
lisp_type               = 3
rloc_probe              = 0
want_map_notify         = 0
record_count            = 4
lisp_nonce              = 0x59f066a1-0x2bab660c
key_id                  = 1
key_len                 = 20
auth_data               =0xa3e543f64e202605dbf4e7c2a2215fdc8083b626
#Recodr0
-------------
record_ttl                = 60
loc_count                 = 4
eid_mask_len              = 24
action                    = 0
auth_bit                  = 0xffffffff
lisp_map_version          = 0x0000-0x00000000
eid_afi                   = 1
eid_prefix                = 153.16.38.0
RLOC 0: priority 4     weight 100    mpriority 255     mweight 0     rloc_local 1
rloc_probe 0    reach_bit 1  loc_afi 1        locator = 132.227.62.242
RLOC 1:                 priority 2      weight 100      mpriority 255   mweight 0
rloc_local 0    rloc_probe 0    reach_bit 1  loc_afi 1        locator =
132.227.62.243
```

## 5. Install

Please refer to the INSTALL file in tarball source code.

## 6. Comment and bug

All comments and bugs report are welcome!