Implementing OpenLISP with LISP+ALT

Attilla de Groot
attilla.degroot@os3.nl
attilla@attilla.nl

April 14, 2009

Abstract

Due to the exponential growth of the BGP routing table in the "default-free-zone" the Locator ID Separation Protocol (LISP) is being developed. OpenLISP is one of the implementations of this protocol, however it does not have the function to route locator and endpoint id addresses. This document describes how a lisp daemon should interact with OpenLISP, GRE and Quagga to use LISP+ALT as a control plane.

Contents

1 Introduction

Over the last ten years the Internet has grown exponentially. Not only the number of connected hosts has increased, but also the number of routable prefixes. This has lead to the discussion of creating a more scalable solution to reduce the routing table size in the "default-free-zone" (DFZ). The growth of the routing table and a possible solution are explained in Section 2.

There have been several protocols suggested in the Routing Research Group that provide a locator id split; the Locator ID Separation Protocol [1] (LISP) is one of them. The locator ID Separation protocol is a mapping and encapsulation implementation of the locator id split. To realise a locator id split a table with mappings between locator and end points is needed. The LISP+ALT system [2] is an overlay network based on already available protocols that provides a mapping system. In Section 3 the LISP protocol and the LISP+ALT system is explained.

At the time of writing there are two implementations of the lisp protocol. One is a propriatary implementation on the Cisco NX-OS [3] operating system. The other implementation is OpenLISP [4] developed by the Belgium university of Louvain. The implementation of OpenLISP is explained in section 4.

There is currently no integration between a LISP+ALT network and the OpenLISP implementation. Since the system is based on already available protocols this lead to the following research question:

> *Is it possible to let the OpenLISP implementation interoperate with a LISP+ALT network by using existing open source software?*

In the next sections the practical deployment of OpenLISP in the LISP+ALT network is studied. Some current impediments with the OpenLISP implementation are identified and recommendations for further research are presented.

2 Scalability of Internet Routing

The growth of the routing table in the "default-free-zone" (DFZ) has lead to discussions in the Routing Research Group [5] on a more scalable routing system. It is commonly agreed upon that splitting the routing and identification space (IP-adresses) into a locator and endpoint IDs is the way to provide scalability. In the Internet Research Task Force [5] several drafts have been submitted with protocols to provide such a locator ID split. The LISP protocol provides this through a map and encapsulation solution.

2.1 BGP and the Default-Free-Zone

Currently, the BGP [6] protocol is used for inter-domain routing. Every domain can be seen in BGP terms as an autonomous system (AS). Every AS has one or more IP subnets which are announced between BGP speakers that have a peering relationship. A simplified representation can be seen in Figure 1.

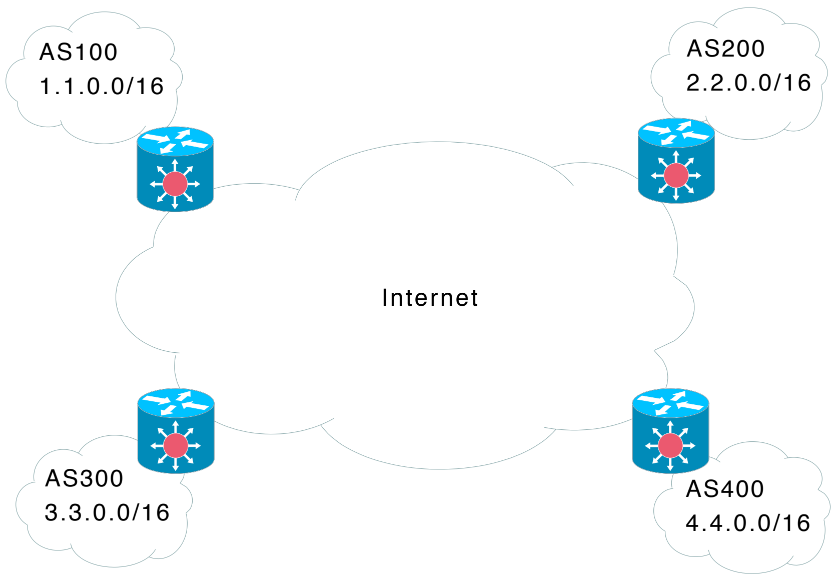

Figure 1: BGP Routing

Assuming that every AS has one BGP speaker and AS400 has a peering with AS200 and AS300 the routing table of the AS400 BGP router would look like table **??**.

Over the last 15 years the Internet has grown to over 30.000 autonomous systems and the use of classless inter-domain routing [7] has lead to a routing table with almost 300.000 entries as can be seen in figure 2. This is

Network	Next Hop	Path
3.3.0.0/16	3.3.0.1	400 300
2.2.0.0/16	2.2.0.1	400 200
1.1.0.0/16	3.3.0.1	400 300 100
	2.2.0.1	400 200 100

Table 1: BGP FIB for AS400

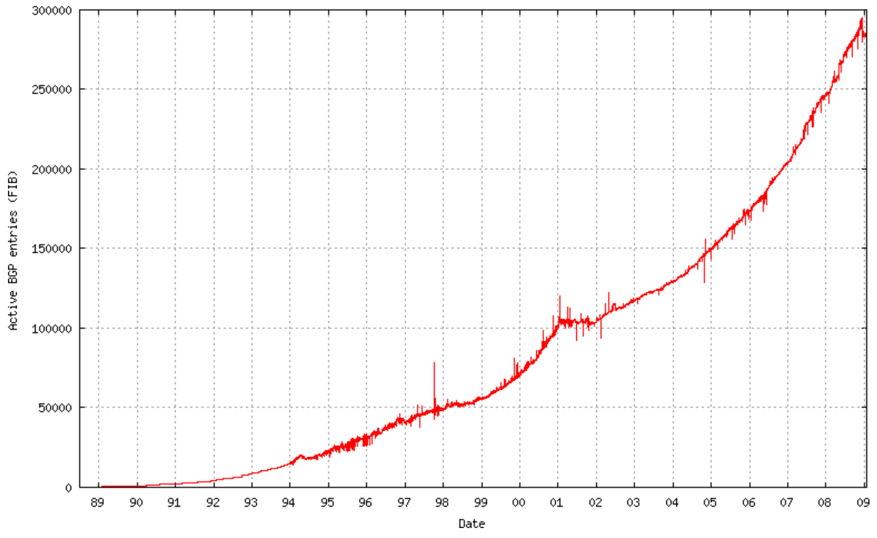

Figure 2: Growth of the BGP Table - 1994 to Present [8]

caused by more provider independent address space that has been given out over the last few years and less provider aggregation. Another factor of table growth is multihoming, where two providers announce the same subnet.

BGP was not ment to be used for traffic engineering. However routing updates are abused to control traffic flows in the network. This causes a more dynamic routing table and more entries if subnets are separately announced and thus more growth of the Forwarding Information Base (FIB).

The FIB is finally loaded into high speed memory, because for every IP packet a lookup has to be done. This kind of memory is costly in design and manufacturing. Having a large FIB increases the operational cost of having a router in de DFZ.

2.2 Splitting location and ID

In current (inter)network routing an IP-address is used as the location and identifier. Discussions in the RRG have lead to the conlusion that splitting location and identifier is a commonly accepted solution to limit the growth of the routing table. If such a system is used in Internet routing this will lead to a situation where every AS has one or a few locators that represent multiple subnets. The routing table will then only contain locator addresses. The ids for the endpoints will be stored in a seperate mapping table.

As result of a locator ID split we will have a more stable DFZ, because no routing updates are needed if an endpoint prefix is moved, fragmentation of the identity space is not an issue because of the RLOC to end ID mapping, and in a multihoming situation only one mapping for the endpoint prefix is needed.

To implement a location/ID split addressing, headers, and protocols need to be adjusted. Several proposals have been made for this implementation [9, 10]. The Locator ID separation protocol is a map and encapsulation implementation of the location/ID split approach.

3 Locator ID Separation Protocol

The Locator ID Separation Protocol (LISP) is an implementation of the encapsulation solution for the locator id split. The development of the LISP protocol continued at the IETF after the Internet Architecture Board [11] routing and adressing workshop in 2006 [12]. At the time of writing an IETF workgroup is being setup for further development of the protocol.

Although the size of the routing table is one of the main problems LISP should solve, it also adresses traffic engineering. By using priority and weight from the authoritative side you can control your ingress traffic and load balance your traffic over multiple LISP devices.

The LISP protocol can be devided into two sections: packet encapsulation and mapping. The data plane takes care of the encapsulation proces and problems that may occur in this process. As is explained in Section 2.2 a seperate mapping table is needed. The protocol provides for mechanisms to fill and update the mapping table.

3.1 Encapsulation

To reduce the routing table an original IP packet is encapsulated with two new headers as seen in Figure 3. The LISP header includes the locator reachability information and a nonce. This header is then encapsulated with another ip header that is used to route the packet. This is send to the well-known LISP UDP port 4341. Since the encapsulation process uses standard IP headers, nothing in the protocol excludes the use of IPV6 either as locator IDs or as endpoint IDs.

To prepend a LISP header and outer ip header two new devices are introduced into the network.

- An Ingress Tunnel Router (ITR) that adds a LISP header, encapsulates it with another IP-header and sends it to the destination router.
- An Egress Tunnel Router (ETR) that removes the extra headers and sends the packet to the final destination.

Both functions can be combined in one device and is named an XTR.

As with every encapsulation protocol the MTU might be a problem. Since lisp adds additional headers the packet becomes larger and could exceed the maximum MTU in the path. According to the draft most traffic paths can accommodate a path MTU of at least 4470 bytes. It has to be determined in a deployment pilot if and what problems are caused by adding the extra headers. At this moment there is no plan to add a mechanism for fragmenting packets at an ITR.

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     / |Version|  IHL  |Type of Service|          Total Length         |
    /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   |         Identification        |Flags|      Fragment Offset    |
   |   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   OH  |  Time to Live | Protocol = 17 |         Header Checksum       |
   |   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   |                    Source Routing Locator                     |
    \  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     \ |                 Destination Routing Locator                   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     / |       Source Port = xxxx      |       Dest Port = 4341        |
   UDP +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     \ |           UDP Length          |        UDP Checksum           |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   L / |S|                 Locator Reach Bits                          |
   I   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   S \ |                           Nonce                               |
   P   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     / |Version|  IHL  |Type of Service|          Total Length         |
    /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   |         Identification        |Flags|      Fragment Offset    |
   |   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   IH  |  Time to Live |    Protocol   |         Header Checksum       |
   |   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   |                           Source EID                          |
    \  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     \ |                         Destination EID                       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: LISP headers

3.2 Mapping information

Before packets can be encapsulated a mapping table has to be filled with locator ID to end point mappings. The protocol draft proposes two new messages to fill a mapping database: map-request and map-reply.

When an ITR receives a packet with a destination IP for which it has no mapping, it sends out a map-request to the authoratative ETR. The map-request also includes the EID prefix information of the source ITR, because it is likely that information has to be send back to the orginating AS. When an ETR is also functioning as an ITR it can choose to cache the information so it does not have to send a seperate map-request.

The map-request messages has several additional fields that are used to fill the mapping table.

- Nonce
  The Nonce is used for route returnability. An ITR wil only accept a map-reply from an ETR that replies with the same Nonce. This nonce is not used as a security technique.

- Locator Reach Bits
  With the locator reach bits can be determined whether an ETR is reachable.
- Record Count
  A map-request can hold multiple records. The record count gives the number of records in the map-request.

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |S|                   Locator Reach Bits                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                             Nonce                             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Type=1 |A|R|            Reserved               | Record Count  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Source-EID-AFI        |             ITR-AFI           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                   Source EID Address  ...                     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                Originating ITR RLOC Address ...               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  / |   Reserved    | EID mask-len  |        EID-prefix-AFI         |
Rec +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  \ |                       EID-prefix  ...                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                   Map-Reply Record  ...                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Mapping Protocol Data                     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Map Request

If an ETR receives a map-request, it responds with a map-reply message that includes the mapping for the end ID prefix. The map-request also has some additional fields for managing the mapping table.

- TTL
  The TTL value is used to determine how long the record should be stored.
- Priority
- Weight
  With the priority and weight field ingress traffic can be controlled. Priority detemines the priority, weight loadbalances the traffic over multiple ETRs.

With the scheme of map-requests and map-replies there is a problem with the first packet that is send. If there is no mapping available the packet will be dropped by the system. To prevent dropping of packets a data probe

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |x|                   Locator Reach Bits                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                            Nonce                              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Type=2 |              Reserved                 | Record Count  |
+-> +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   |                          Record  TTL                          |
|   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
R   | Locator Count | EID mask-len  |A|         Reserved            |
e   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
c   |           Reserved            |            EID-AFI            |
o   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
r   |                          EID-prefix                           |
d   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  /|    Priority   |    Weight     |  M Priority   |   M Weight    |
| L +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| o |           Unused Flags      |R|           Loc-AFI             |
| c +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  \|                            Locator                            |
+-> +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Mapping Protocol Data                     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Map Reply

can be created. A data probe has the same structure as a standard encapsulated packet, see Figure 3, however the destination address of the inner headers is copied to the outer headers. When a packet is received by an ETR and it detects the same destination address in the inner and outer header it knows that it is a data probe and will reply with a map-reply packet and still deliver the orginal packet to the endpoint. There is an ongoing discussion in the RRG if a data probe is needed for the protocol.

3.2.1 Life of a Packet in the LISP Protocol

In the previous section the details of the LISP protocol are explained. But how is a packet actually send over the network? For example we have a network setup like Figure 6 where host 1.1.1.1 and host 2.2.2.2 are communicating using the lisp protocol.

1. Host 1.1.1.1 wil send a TCP packet to host 2.2.2.2 over its local network. Since it is a destination outside its own subnet it will send it to its default router. This maybe the ITR itself or a internal gateway protocol router, eventually the packet will get routed outside its own AS and thus needs to cross the ITR.

2. The ITR will prepend a LISP header and adds a second IP header. If there is a known mapping for the end ID the ITR will use the locator as destination id. If the mapping is unknown, it will create a dataprobe by using the original destination address as destination in the second

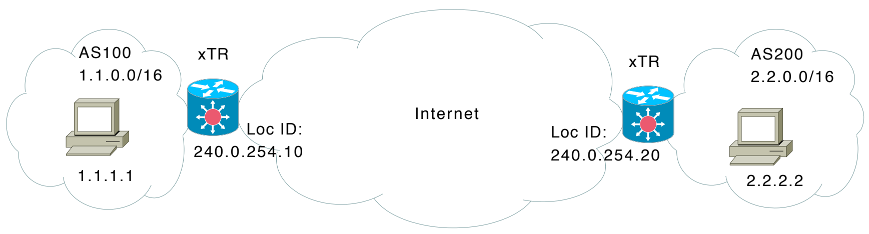

Figure 6: LISP Network

header. The ITR will use data probes untill the EID prefix is available in the mapping database.

3. If the packet is a data probe, the ITR will route the packet over the control plane (if lisp+alt 3.3.1 is used). If it is a normal packet it will route the packet over the data plane.
4. In either case the ETR will receive the packet and strips the extra headers. If the packet is a data probe, it will send a map-reply message to the ITR. The ETR will then forward the original packet to the endpoint.
5. The ITR will receive a map-reply if the packet was a data probe and will install the mapping in the database.
6. For subsquent packets to the same EID prefix the locator address wil be used in the second IP header and the packet will be routed directly to the locator.

This scheme assumes the use of a data probe. If data probes are not used LISP will send a map-request when a mapping is not available and will drop packets untill the map-reply information is installed in the database.

3.3 Mapping system

As explained in Section 2 one of the goals of the locator id split is reducing the size of the routing table. To send encapsulated packets to an authoritative ETR there is a mapping table needed. There have been several solutions suggested [2, 13, 14, 15, 16, 17] for a system that provides these mappings.

3.3.1 LISP+ALT

The LISP+ALT network is a gre overlay network with bgp to route the endpoint prefixes. An end-point network is connected to the core ALT network by GRE tunnels. This leads to a topology as seen in Figure 7. Instead having the full table itself, an ITR will forward either a data probe or map-request to the end-point of the gre tunnel and the packet will be routed to the authoritative ETR. The map-reply messages are not send over the ALT network, because the ETR knows the source address of the ITR that can be routed normally over the internet.

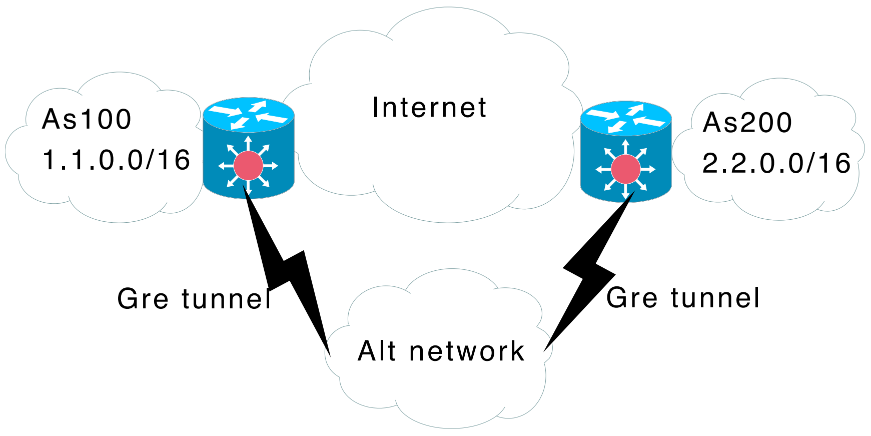

Figure 7: ALT network

3.3.2 Routing in LISP+ALT

The ALT network is composed of core routers with multiple end-points which are interconnected using GRE tunnels.This gives the possibility to build a more hierarical bgp structure and aggregate more routes. An ITR announces its own EID prefix into the ALT network. This means that only the core of the alt network holds the full, more aggregated, BGP table. The ITR contains the mapping for active sessions and can be run on less dimensioned hardware, which will reduce the operational cost.

As said, an XTR has a connection to the "normal" internet, where only packets with locator addresses are routed and a gre connection to the ALT network. This setup means that an XTR needs to run two bgp instances (one for the "normal" internet and one for the ALT network). The locator address space will be provided from seperated IP space and since only these addresses are advertised the routing table will be much smaller. In the ALT network it will be sufficient that a default route is advertised to an XTR.

In Section 3.3.1 is explained that only an ITR sends packets onto the ALT network. A logical configuration for an ITR would be to have a default route to the ALT network and more specific routes to the "normal" internet. This will will automatically lead to the situation that data probes are send onto the ALT network and be delivered at the authoritative ETR. In the lisp draft however is stated [1, This is performed by using the RLOC as the destination address for Map-Request message], this is only desired if a map-request is not send upon a cache miss. If there is a cache miss, the XTR does not know the rloc address for the EID prefix. Only in case of the first map-request the destination address of the original packet should be used. Only in this case the map-request will be routed to the authoritative ETR. This statement in the draft will be adjusted in the next version.

3.3.3 LISP4

At this moment Cisco has build a LISP+ALT network (lisp4) for testing purposes, the layout of this network can be seen in Figure 8. Cisco workes closely with Regional Internet Registries (RIR) to position the lisp core routers, since they are the most logical choice of aggregating the numbering space. However a RIR such as Ripe does not have any operational responsibilities on the internet now. Further research is needed to determine if this is an acceptable position for a RIR.

In the layout NLnet Labs is already an endpoint for the ALT network, connected to the RIPE core.

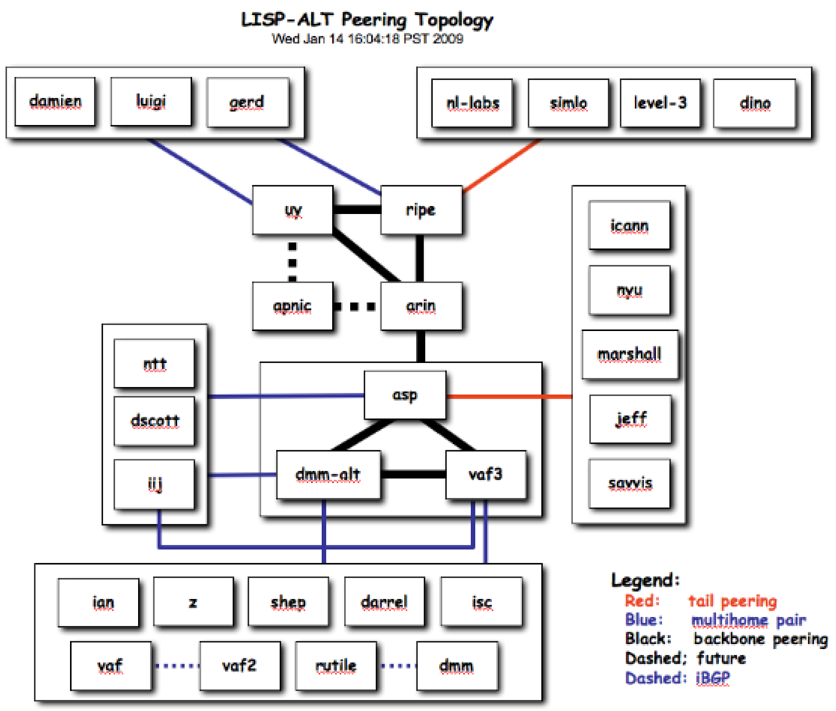

Figure 8: LISP4 network

4 OpenLISP

The LISP development is mainly driven by developers from Cisco. Despite the fact that most of the software from Cisco is proprietary closed source, their approach to the LISP protocol development is open. All the documentation concerning the standard is published in the IETF, except the NX-OS implementation details. OpenLISP is an open source implementation for FreeBSD based on the LISP draft (version 07) and is currently the only implementation next to the NX-OS implementation of Cisco.

As the LISP protocol itself, the OpenLISP implementation can be devided in two parts and are described in two section respectively. The first section discusses the encapsulation implementation, the second the implementation of the mapping table.

4.1 Packet encapsulation

The packet encapsulation of LISP is implemented as a patch on the IP protocol stack of the FreeBSD kernel. The design of the protocol is intended to be simple.

Although the design provides the possibility for IPV6 implementation as orginal packets or usage in the outer header, the current implementation of OpenLISP doesn't include support for IPv6

4.1.1 Incoming packets

For incoming packets the adapted ip stack implements a kind of loop where the outer header and LISP header are removed, as seen in Figure 10

If a LISP packet is received by the ip_input module and recognized as a LISP packets by the well-known udp port 4341, it is forwarded to the lisp_input module. Non-lisp packets are inserted into the transport layer as they normally would. The lisp_input module uses the information in the LISP header and reinserts the original packet into the ip_input module. This allows the module to handle the packet as it normally would, by routing it to an end-point or forwarding the packet for local delivery.

4.1.2 Outgoing packets

The implementation of the IP stack for outgoing packets follows a similar approuch as for the incoming packets.

First a normal IP packet is created. This packet can be from the local system itself or from the ip_forward module if kernel routing is enabled. If there is a LISP mapping available for the destination address the LISP header is added and inserted again into the ip_output module where the locator ID is used as the destination address in the IP header. In the case

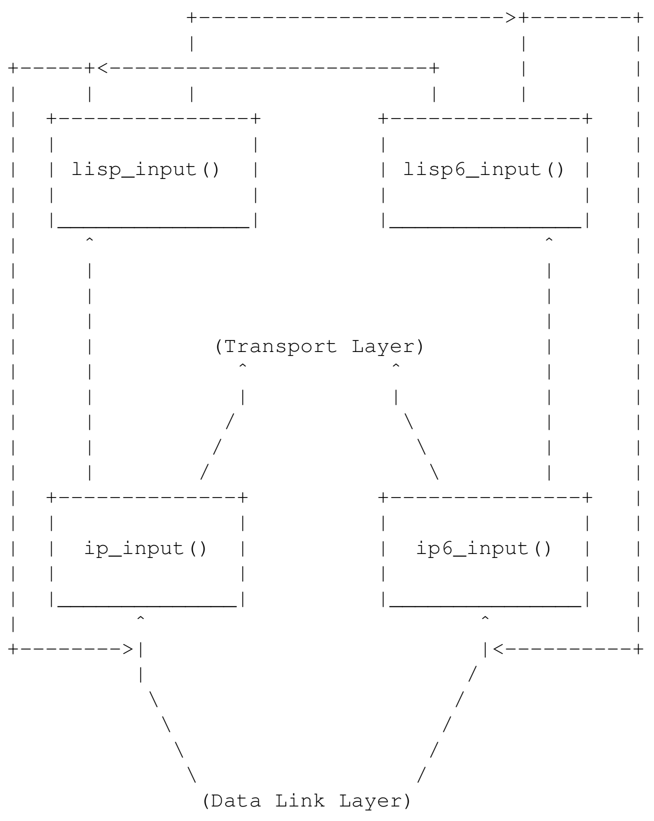

Figure 9: Protocol Stack Modifications for incoming packets

that there is no mapping for the destination, a cache miss is send over the API in order to generate a map-request packet and contact the authoritative ETR for the prefix mapping. Because there is no mapping the packet will be routed normally, however in a LISP implementation this will mean that the packet is dropped. OpenLISP does not have an implementation of the data probe mechanism as specified in the draft.

4.1.3 MTU Management

As is described in Section 3.1 the MTU may cause problems when encapsulating packets. OpenLISP doesn't implement a packet fragmentation system. It uses the MTU setting of the interface of the RLOC and adds it to the mapping database entry of the RLOC. When a packet is received to encapsulated, the final MTU size is calculated. When it exceeds the MTU of the interface an icmp "too big" message is send and the packet is dropped. Because of this implementation, a packet will never exceed the maximum interface MTU value. Problems will be not different then with normal packets.

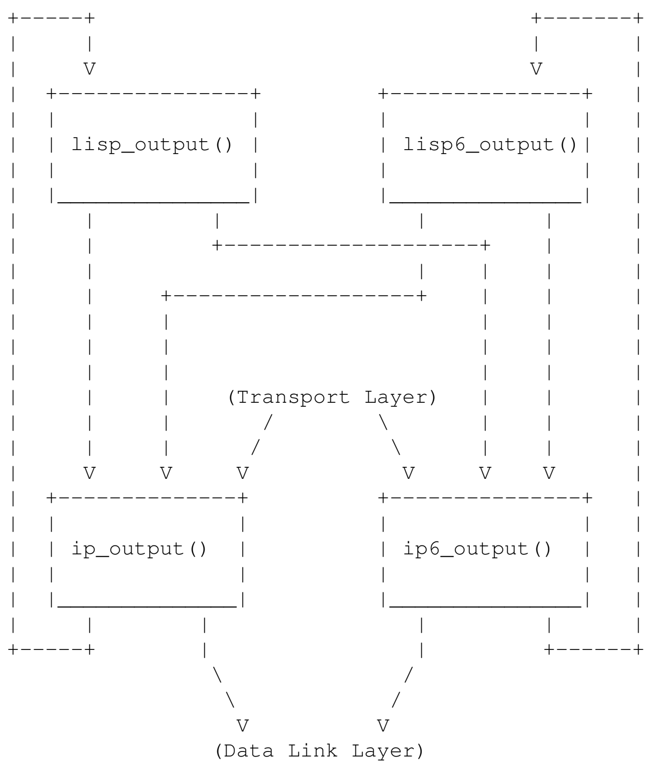

Figure 10: Protocol Stack Modifications for outgoing packets

4.2 Mappingtable implementation

OpenLISP defines two seperate mapping tables. One table is a cache table for short lived mappings, the other is defined for locally available EID prefixes. The mappings in the tables contain the necesary metrics, such as weight, priority and the MTU value as explained in Section 4.1.3.

The OpenLISP implementation provides an api to read and write to the tables. Also two commandline tools are available to manually manage the tables.

4.2.1 Command line tools

The map tool provides a way to add, remove, and view mappings in the database manually, as can be seen in the following examples. The man (1) map explains usage details.

In the first example a local mapping is added. If LISP encapsulated packets are received at the locator address (145.100.104.13), the headers will be removed and the packet will be forwarded to its orginal destination. To configure a local mapping one of the interfaces should have an IP-address configured from the EID block.

Map view
```
[airwolf@os3-lisp ~]$map get -inet 145.100.104.192
    Mapping for EID:   145.100.104.192
       EID:   145.100.104.192
       EID mask:   255.255.255.224
       RLOC Addr : inet 145.100.104.13 P 255 W 100 Flags i MTU 1500
       flags: <UP,LOCAL,STATIC,DONE>
```

Map write
```
[airwolf@os3-lisp ~]$ sudo map add -local -inet 145.100.104.193/27
-inet 145.100.104.13

add DB 145.100.104.194
```

With the mapstat tool (based on the netstat utility) LISP statistics can be viewed. These stats are from one of the OpenLISP test machines that send encapsulated ICMP packets.

Mapstat
```
[airwolf@os3-lisp ~]\$ mapstat -sf inet -p lisp
lisp:
        6945 datagrams received
        0 with incomplete header
        0 with bad encap header
        0 with bad data length field
        6945 delivered
        4856 datagrams output
        0 dropped on output
        4856 sent
```

4.2.2 API

The OpenLISP implementation is developed in line with the unix philosophy to let a daemon in user space control the mapping table. To provide access to the mapping table for a daemon, mapping sockets are created. The mapping sockets are based on the well known routing sockets.

If a process has opened a mapping socket it can read en write through three operations:

- MAPM_ADD
  Used to add mappings to the table, and read result from the kernel.
- MAPM_DELETE
  Used to delete mappings from the table, same as add.
- MAPM_GET
  Is used to retrieve information from the mapping table. A specific EID should be given in the query.

Next to the operations the daemon can perform on the mapping table, the kernel also sends messages. As explained in Section 4.1.2 before the encapulation is done the mapping from the database is retrieved. However when there is no mapping available, a map-request has to be send by a daemon. The OpenLISP kernel patch can trigger three different messages. If there is no mapping available a MAPM_MISS is send. This message should be a trigger for a daemon to send a map-request message to the locator.

The current implementation of OpenLISP is based on the 07 draft of LISP. Although it is in the unix spirit to let a user space daemon control the mapping information, there is currently no daemon available that can accomplish this.

5 Implementing OpenLISP with LISP+ALT

In the previous sections we explained the LISP protocol, the LISP+ALT network and the OpenLISP implementation. At this moment there has been no interoperability between the OpenLISP implementation and a LISP+ALT network.

To let OpenLISP interoperate with LISP+ALT we can devide the implentation in four parts.

- Gre
  As is explained in Section 3.3.1, the ALT network is an overlay network based on the GRE protocol. This means that a LISP XTR should be able to create a GRE tunnel to one or more ALT network core routers.

- Bgp
  Over the GRE overlay network BGP is used to aggregate the endpoint prefixes. A XTR has to announce its assigned eid prefix into the ALT core over BGP. The XTR should therefore have the bgp protocol available.

- Encapsulation
  The primary function of an XTR is to encapsulate and decapsulate packets. The OpenLISP implementation provides these functionalities.

- Mapping information
  To let OpenLISP encapsulate packets mapping information has to be exchanged over the ALT network. The OpenLISP implementation has to send the map control packets over the ALT network.

In the next sections will be explained how these parts can be implemented with current open source software or, in case they do not exist, should be build. For implementation testing the LISP4 (Section 3.3.3) network is used. Ripe has provided NLnetLabs with an eid prefix to build a tail-peering.

5.1 Gre

To build a GRE connection to the ALT network an implementation of the protocol is needed. The GRE protocol is a simple protocol [18] that encapsulates the orginal packet with a second ip header.

Since OpenLISP is currently only available for FreeBSD, this operating system is used as the testing platform. FreeBSD has a native implementation [19] of the GRE protocol. This only has to be configured with the details provided by Ripe to connect to the endpoint. In most implementations GRE can be configured as an extra interface as is the case in FreeBSD.

To configure a GRE interface the ifconfig command can be used. Since Ripe provided the details this was a simple case of filling in the right values.

FreeBSD gre configuration

```
[root@phobos ~]# ifconfig gre0 240.0.254.169 240.0.254.168 netmask
255.255.255.252 tunnel 213.154.224.103 193.0.0.170

[root@phobos ~]# ifconfig gre0
gre0: flags=9051<UP,POINTOPOINT,RUNNING,LINK0,MULTICAST> metric 0 mtu 1476
        tunnel inet 213.154.224.103 --> 193.0.0.170
        inet 240.0.254.169 --> 240.0.254.168 netmask 0xfffffffc

[airwolf@phobos ~] ping 240.0.254.168
PING 240.0.254.204 (240.0.254.168): 56 data bytes
64 bytes from 240.0.254.168: icmp_seq=0 ttl=255 time=6.185 ms
64 bytes from 240.0.254.168: icmp_seq=1 ttl=255 time=6.197 ms
--- 240.0.254.204 ping statistics ---
2 packets transmitted, 2 packets received, 0.0\% packet loss
round-trip min/avg/max/stddev = 6.185/6.191/6.197/0.006 ms
```

5.2 Bgp

Since we now have a GRE tunnel to the ALT network, the assigned EID prefix has to be announced to the Ripe core router. FreeBSD doesn't have native support for BGP, but there are several others available. A frequently used BGP implementation is the BGP daemon in the Quagga routing suite. This suite supports multiple routing protocols and allows the protocols to adjust the routing table through a routing management daemon.

As explained in Section 3.3.1 only the core ALT routers hold the full routing table, tail peerings such as the one from NLnetLabs should only announce their prefixes into the network. This leads to a simple configuration, because only a bgp neiggbour and the prefix announcement has to be setup. In a live situation it would be advisable to also use filtering to prevent spurious routes to be installed.

All the Quagga daemons provide a Cisco like interface which is accessible through telnet. To announce the prefix only the router-id, prefix and the Ripe neighbour have to be provided.

Quagga bgpd configuration

```
router bgp 2147483677
 bgp router-id 153.16.36.254
 network 153.16.36.0/24
 neighbor 240.0.254.204 remote-as 2147483671
```

This should be enough to have a Peering into the LISP4 ALT network. However the BGP daemon soon after the connection was made.

Quagga bgpd crash
```
2009/01/13 10:27:27 BGP: BGPd 0.99.11 starting: vty <at> 2605,
bgp@<all>:179
2009/01/13 10:33:30 BGP: 240.0.254.204 unrecognized capability code:
67 - ignored
2009/01/13 10:33:31 BGP: Assertion 'len < str_size' failed in file
bgp_aspath.c, line 619, function aspath_make_str_count
2009/01/13 10:33:31 BGP: No backtrace available on this platform.
Abort trap: 6
```

In the LISP4 network asn32 AS numbers are used. The Quagga bgpd does have support for this in the unstable version, however it seemed that the daemon has a fixed maximum length of AS numbers (6 digits). Since the LISP4 network uses 10 digits AS numbers, the daemon crashes if there are long AS numbers in the BGP as_path.

During the reseach project a quick fix was done and the bug reported on a Quagga mailinglist [20].

5.3 Encapsulation

An XTR should be able to encapsulate and decapsulate the packets since that is its main function. OpenLISP provides this function if it can find a mapping for the endpoint ID. Since OpenLISP sends packets to the locator ID it is routed over the "normal" internet instead of the ALT network. So for the OpenLISP implementation no special configuration is needed.

However there is one situation when OpenLISP should send packets onto the ALT network. In case when there is no mapping for the endpoint ID OpenLISP should encapsulate the packet with a LISP header and outer IP header. The destination address of the inner header should be copied to the outer header, making the packet a data probe. This packet should be send onto the ALT network.

In the current implementation of OpenLISP the packet is discarded when there is no mapping in the database (and the system is not able to normally route the packet). In order to have a full implementation of the LISP draft a data probe mechanism should be implemented.

5.4 Mapping Information

To decrease to size of the routing table an XTR should send map-requests over the ALT network. As explained in Section 4.2.2, OpenLISP is only a kernel patch with an API. This means that there is a seperate daemon necessary to send map-requests over the ALT network.

At time of writing a mapping daemon still has to be developed, developing a daemon is out of the scope of this research project. This means that interoperability between OpenLISP and a LISP+ALT network is not yet possible. It is however that this is the only "part" missing for succesfull integration. This section will describe the requirements and functional design for the mapping daemon that is needed.

5.4.1 Requirements

For basic interoperability a mapping daemon with the following requirements is needed:

- Communication with OpenLISP kernel api socket.
  - Write mappings to database through MAPM_ADD operation.
  - Read mappings from database through MAPM_READ operation.
  - Retrieve information from database through MAPM_GET operation.
  - Listen on mapping socket for MAPM_MISS, MAPM_BADREACH, or MAPM_REACH operation.
- LISP mapping messages.
  - Sending map-request, triggered by socket operation.
  - Listening on well-known udp port 4342
  - Sending map-reply message with information from database, triggered by a received map-request message.

5.4.2 Functional Design

A functional design for the mapping daemon can be seen in Figure 11. This design provides functionality for three scenarios.

- If the XTR has to encapsulate a packet where it has no mapping for, a map-request is send. The API uses the MAPM_MISS operation that triggers the daemon to send the packet. This packet is send over the gre tunnel into the core ALT network.
- If a map-request from the ALT network is received a map-reply is send. The map reply has to be filled with information from the mapping database and can only be send if the XTR is authoritative for the requested prefix.
- If the XTR receives a data probe it should let the api send a message to send a map-reply. As explained in Section 5.3 data probes are not yet supported. Since the daemon already communicates with the api and is capable to send map-replies, it won't be a problem to let the daemon respond on a data probe.

These scenarios mean that the daemon could function both as an ITR and ETR. In the draft the devices are still seperated. The daemon should be able to function as just an ITR or ETR if set by a management function.

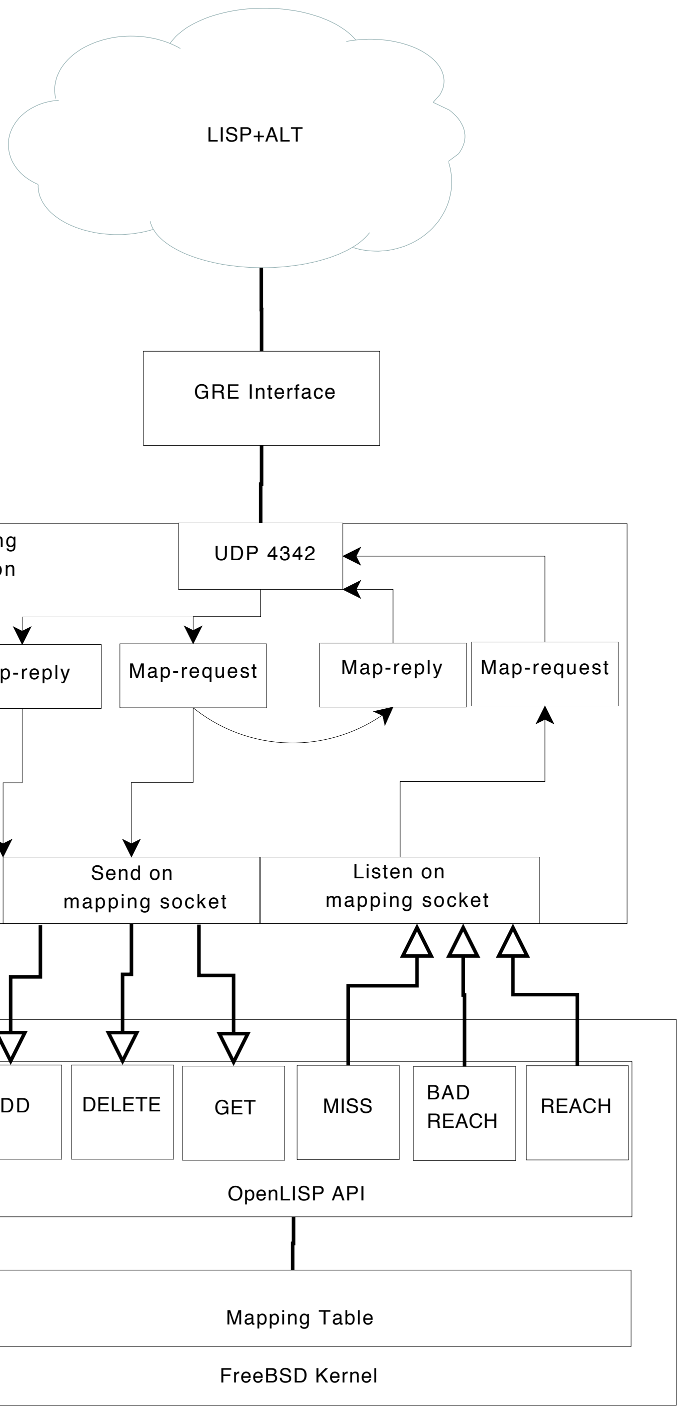

Figure 11: Functional mapping design

6 Conclusion

Reviewing out research question from Section 1.

Is it possible to let the OpenLISP implementation interoperate with a LISP+ALT *network by using existing open source software?*

At this time can be concluded that it is not possible to let the OpenLISP implementation interoperate with a LISP+ALT network. The OpenLISP implementation is based on the LISP draft, however it is developed according to unix principals where a daemon in userspace is needed to fill and update the mapping table. In the current version of OpenLISP such a daemon is not included and there is also no third party daemon available. In Section 5.4 requirements and a functional design are described for such a daemon.

To let OpenLISP interoperate with a LISP+ALT network also the GRE and BGP protocols have to be available. They are implemented according to the standards and as described in Section 5 no considerable problems were encountered when the FreeBSD GRE or Quagga BGP implementations were tested with the LISP4 NETWORK.

The current OpenLISP encapsulation does not need any extra features to interact with the LISP+ALT network. However it does not have support for the data probe mechanism, while it is specified in current LISP draft. It is still debated if it is needed for the protocol, but if OpenLISP would support the data probe it could be tested against other implementations and in combination with a mapping daemon.

6.1 Further research

This research is done in a four week project and has lead to the following future research questions.

- Development of mapping daemon
  The absence of the mapping daemon is the only part missing to let the OpenLISP implementation interoperate with the LISP+ALT network ans should be developed to keep the development of the lisp protocol and mapping system multiplatform. NLnet Labs intends to develop this daemon in the near future.

- Security in OpenLISP and LISP+ALT implementation
  The preliminary LISP threat analysis [21] shows that if the mapping database can be manipulated several attacks are possible. The LISP protocol was build to provide scalability it does not rely on a PKI infrastructure. Building a scalable authentication, based on open standards and if possible open source software would be a good addition to the LISP protocol.

- Location of core LISP+ALT routers
  In the LISP4 network core routers are placed at RIR'S to provide efficient aggregation. An interesting research would be if the RIR'S are indeed the best location to manage and host the core routers. For example organisations such as the SIDN already have operational responsibility and could still provide aggregation.

References

[1] D. Farinacci *et al.*, Locator/id separation protocol (lisp), 2008, http://tools.ietf.org/html/draft-farinacci-lisp-11.

[2] V. Fuller *et al.*, Lisp alternative topology (lisp+alt), 2008, http://tools.ietf.org/html/draft-fuller-lisp-alt-03.

[3] C. Systems, Nx-os, http://www.cisco.com/en/US/products/ps9372/index.html.

[4] L. Iannone *et al.*, Openlisp implementation report, 2008, http://tools.ietf.org/html/draft-iannone-openlisp-implementation-01.

[5] Routing research group, http://www.irtf.org/charter?gtype=rg&group=rrg.

[6] Y. Rekhter and T. Li, A border gateway protocol 4 (bgp-4), 1995, http://www.ietf.org/rfc/rfc1771.txt.

[7] V. Fuller *et al.*, Classless inter-domain routing (cidr), 1993, http://www.faqs.org/rfcs/rfc1519.html.

[8] 2009, http://bgp.potaroo.net/as2.0/bgp-active.html.

[9] Shim6 working group, http://tools.ietf.org/wg/shim6/.

[10] M. O'Dell, An alternate addressing architecture for ipv6, 1997, http://tools.ietf.org/html/draft-ietf-ipngwg-gseaddr-00.

[11] Internet architecture board, http://www.iab.org.

[12] D. Meyer *et al.*, Report from the iab workshop on routing and addressing, 2006, http://tools.ietf.org/html/draft-iab-raws-report-02.

[13] L. Mathy, L. Iannone, and O. Bonaventure, Towards a dht to map identifiers onto locators, 2008, http://inl.info.ucl.ac.be/system/files/draft-mathy-lisp-dht-00.txt.

[14] J. Curran *et al.*, Eid mappings multicast across cooperating systems for lisp, 2007, http://tools.ietf.org/html/draft-curran-lisp-emacs-00.

[15] D. Meyer *et al.*, A content distribution overlay network service for lisp, 2008, http://tools.ietf.org/html/draft-meyer-lisp-cons-04.

[16] E. Lear *et al.*, A not-so-novel eid to rloc database, 2008, http://tools.ietf.org/html/draft-lear-lisp-nerd-04.

[17] D. Jen *et al.*, A practical transit mapping service, 2007, http://tools.ietf.org/html/draft-jen-apt-01.

[18] D. Meyer *et al.*, Generic routing encapsulation (gre), 2000, http://tools.ietf.org/html/rfc2784.

[19] Freebsd gre, http://www.FreeBSD.org/cgi/man.cgi?query=gre.

[20] A. de Groot, Bgpd crash on long asn32 in aspath, 2009, http://article.gmane.org/gmane.network.quagga.user/9956/.

[21] M. Bagnulo, Preliminary lisp threat analysis, 2007, http://tools.ietf.org/html/draft-bagnulo-lisp-threat-01.