LISP-DHT: Towards a DHT to map identifiers onto locators

Laurent Mathy
Lancaster University
Lancaster, UK
laurent@comp.lancs.ac.uk

Luigi Iannone*
TU Berlin / Deutsche Telekom Laboratories
Berlin, Germany
luigi@net.t-labs.tu-berlin.de

ABSTRACT

Recent activities in the IRTF (Internet Research Task Force), and in particular in the Routing Research Group (RRG), focus on defining a new Internet architecture, in order to solve scalability issues related to interdomain routing. The research community has agreed that the separation of the end-systems' addressing space (the identifiers) and the routing locators' space will alleviate the routing burden of the Default Free Zone. Nevertheless, such approach, adding a new level of indirection, implies the need of storing and distributing mappings between identifiers and routing locators. In this paper we present LISP-DHT, a mapping distribution system based on Distributed Hash Tables (DHTs). LISP-DHT is designed to take full advantage of the DHT architecture in order to build an efficient and secured mapping lookup system while preserving the locality of the mapping. The paper describes the overall architecture of LISP-DHT, explaining its main points and how it works.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network communications; C.2.6 [**Internetworking**]: Routers; C.2.5 [**Local and Wide-Area Network**]: Internet; C.2.4 [**Distributed Systems**]: Network operating systems

General Terms

Algorithms, Management, Design.

Keywords

Locator/ID separation, LISP, Internet Architecture, Routing, Addressing, DHT.

*This work was performed while Luigi Iannone was at Université catholique de Louvain, Louvain-la-Neuve, Belgium.

ReArch'08, December 9, 2008, Madrid, SPAIN

1. INTRODUCTION

The locator/ID separation paradigm is being discussed in the IRTF (Internet Research Task Force) Routing Research Group (RRG) in order to solve scalability issues that today's Internet is facing ([11], [9], [10]). Works like [12] clearly show that several benefits can be achieved with such an approach, not only in alleviating the routing burden of the Default Free Zone (DFZ).

The Locator/ID Separation Protocol (LISP [5]) is one of the solutions being discussed within the RRG in order to provide the support for this new Internet architecture. A key problem faced by LISP, is that a mapping system will be required to distribute mappings between identifiers and locators in a scalable way. In the LISP specification, for what is called LISP 3 variant, the use of Distributed Hash Tables (DHTs) as mappings distribution system is suggested. In spite of this, none of the mapping distribution systems proposed so far for LISP ([8], [3], [4], [2]) has a DHT-based approach.

LISP-DHT, our proposal, fills this gap by using a DHT lookup infrastructure in order to efficiently retrieve mappings. DHTs exhibit several very interesting properties, such as self-configuration, self-maintenance, scalability and robustness that are clearly desirable for an identifier-to-locators resolution service.

Given the importance of the mapping service for the reachability of the hosts inside a domain, we expect that domains will have the following requirements for their mapping service:

1. A domain must be able to control the server that provides the authoritative mappings for the identifiers allocated to its hosts. Since mappings contain important information not only concerning reachability, but also traffic engineering, a domain needs to have full control over them, thus delegation outside the owner domain is not possible.
2. A domain requires one (or more) redundant mapping server(s), which the domain must be able to control. The purpose, for the domain, is to be always able to provide its mappings, in order to guarantee its own reachability.

These two requirements are similar to the requirements

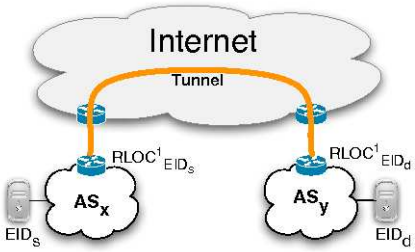

Figure 1: LISP Overview.

of today's network operators concerning their DNS (Domain Name Service) servers. Since classical DHTs tend to randomize which node is responsible for a key-value pair (in our case a key-value pair would be a mapping), they do not directly meet the above requirements. LISP-DHT is based on a modified version of Chord [13], so to retain all its useful properties, while helping to meet the two above-mentioned requirements. Thus, LISP-DHT is able to preserve the locality of the mapping, *i.e.*, the mapping is always stored on the LISP-DHT nodes of the owner of the mapping.

The paper is organized as follows. In Section 2 we give a short overview of the LISP protocol. In Section 3, a brief summary of the Chord DHT is presented. Section 4 describes the principals used in LISP-DHT to preserve mapping locality allowing domain owners to retain control of their mappings. Section 5 describes how the fundamental separation of role between mapping servers and mapping resolvers (i.e. routers) can be enforced in LISP-DHT, while Section 6 addresses LISP-DHT reliability issues. A revision of the currently proposed LISP-related mappings distribution systems is presented in Section 7, before discussing the differences in Section 8 and concluding the paper in Section 9.

2. LISP OVERVIEW

The Locator/ID Separation Protocol (LISP [5]) is based on a simple IP-over-UDP tunneling approach, implemented typically on border routers which act as *Routing LOCators* (RLOCs) for the end-systems of the local domain. End-systems still send and receive packets using IP addresses, which in the LISP terminology are called Endpoint IDentifiers (EIDs). Remark that since in a local domain there may be several border routers, EIDs can be associated to several RLOCs.

The basic idea of LISP is to tunnel packets in the core Internet from the RLOC of the source EID to the RLOC of the destination EID, as depicted in Figure 1. During end-to-end packet exchange between two hosts, the source host EID_s first issues a normal IP packet that is normally routed in the source domain to reach one of its border routers for tunneling. The border router, or Ingress Tunnel Router (ITR), performs the EID-to-RLOC lookup in its local cache, or queries the mapping system if no mapping is available in the cache. The result of the lookup is the RLOC of the destination host EID_d which consist in a border router of EID_d's domain acting as Egress Tunnel Router (ETR). The ITR prepends a new LISP header to the packet before forwarding it, while the ETR strips this header on reception, before delivering the packet to EID_d. The eventual reply of EID_d follows the same rules. Remark that only the first packet may trigger a query to the mapping system. Indeed, LISP uses a local caching mechanism to reduce the frequency of lookup and latency [5].

3. CHORD OVERVIEW

Chord [13] is a DHT using for each node a unique k-bits identifier (called ChordID) and where the whole space of ChordIDs is organized as a ring. A node owns all the keys that precede it on the ring, up to, but excluding, the previous node, and any request sent to a ChordID is routed to the node that owns that ID.

For consistent routing and operation, every Chord node must maintain correct successor and predecessor node pointers. For performance, nodes also keep a finger table (a list of nodes that are further and further away – to be more precise, the i^{th} finger in the table is the node that holds (*i.e.*, succeeds) ChordID $(n + 2^{(i-1)})$, where n is the node's ChordID). These fingers are not necessary to ensure correct operation of the protocol, however, they help in reducing lookup latency (to $O(\log N)$, where N is the number of nodes on the DHT), which can be a main concern in the context of locator/id separation. Reliability and consistency is also increased if a node keeps pointers to several consecutive successors and predecessors on the ring.

To join the DHT ring, a new node only has to know another node on the ring, and use this existing node to initialize its finger table (the first entry of the finger table is the successor to a node) and predecessor pointer, by simply asking that node to look up the corresponding keys (IDs) on the ring. The joining node also needs to then update its predecessor's successor pointer, its successor's predecessor pointer, as well as the finger pointers of other nodes that should now use the joining node in their fingers (these can be easily computed by the joining node). Then a key transfer can take place.

4. MAPPINGS AUTHORITATIVE OWNERSHIP

Chord nodes are supposed to choose their ChordID randomly. Also, keys associated with values should also be random (they should be hash values of the associated value). In the context of the EID-to-RLOC mapping resolution application considered in this document, such randomization is in contradiction with the kind of control expected by domains on their mappings.

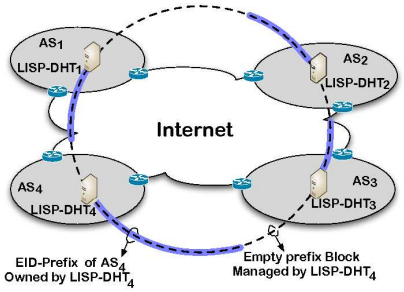

Figure 2: Example of LISP-DHT infrastructure.

To meet the first requirement expressed in Section 1, we propose to use EIDs as ChordIDs on the Chord ring, and have domains create a Chord instance for each of their EID-prefix. The ChordID of each Chord instance would be the highest numerical EID inside EID-prefix.[1] Such a deterministic approach allows to ensure that the domain owns and manages the parts of the Chord ring that contains its identifiers (the domain can also actually own "unallocated" EIDs adjacent to EID-prefix) as depicted in figure 2.

Of course, the value associated with a key (EID-Prefix) on the Chord ring would be a mapping whose granularity can vary from an EID-Prefix and the corresponding list of RLOCs, down to a single EID and a list of RLOCs. It is mainly because of the straightforward use of EIDs as ChordIDs and the associated ownership guarantee that we choose to base LISP-DHT on Chord. Also, all Chord's routing information (used by Chord itself) should associate the (EID, RLOC) pair of a Chord node (*i.e.*, mapping server) with its corresponding ChordID, so nodes on the Chord ring can communicate with each other.

Credentials may be needed before insertion of a new node onto the ring. This could be needed, for instance, to prevent a rogue node from trying to hijack the mapping for some EID-prefix it does not own. This could be achieved by having the appropriate Internet Registries issue EID owners with a signed "certificate of ownership" for each prefix. The Secure Inter-Domain (SIDR) working group is working on X.509 certificates for IP address prefixes [7] that are very close to the certificates that would be required by LISP-DHT. Such a certificate can be logically defined in the following manner:

M_{own} = (EID-prefix, owner, expiry);
C_{own} = (M_{own}, Reg, Epriv(Reg, H(M_{own})));

Where M_{own} represents a message containing the EID-Prefix and a description of its owner, Reg is a description of the Internet Registry responsible for allocating the EID-Prefix, expiry is the expiry date of the certificate, H(M_{own}) is a hash of message M_{own}, and Epriv(Entity, D) is the encryption, using Entity's private key (a registry in our case) of data D (the hash of M_{own} in our case). As the owner of an EID-Prefix will often be an organization, this organization must issue the authoritative mapping for its EID-Prefix with a *certificate of authority*:

M_{auth} = (ChordID, server, C_{own}, expiry);
C_{auth} = (M_{auth}, Epriv(owner, M_{auth}));

where ChordID is the Chord node ID (last address in the EID-prefix) and server is the (EID, RLOC) pair identifying the server. By including this certificate of authority in all Chord messages that trigger an update of routing information in other Chord nodes, a mapping server willing to join the ring can prove its right to be inserted in the corresponding place on the Chord ring. Note that for the system to be trusted, this Public Key Infrastructure (PKI) should be supported by the use of digital certificates and that any description of entities that may need to be authenticated should include a link to that entity's digital certificate.

Finally, unless a re-allocation of EID-prefixes is taking place, no key transfer should happen, as a joining mapping server is simply claiming the part of the Chord space for which it owns the mapping.

5. STEALTH CHORD FOR LISP-DHT

The users of mappings (*e.g.*, ITR) will not necessarily also serve as mapping servers. Indeed, it is actually very likely that network providers will deploy a server infrastructure dedicated to mapping resolution, relieving the routers from the burden of answering mapping requests (this mirrors the DNS system where servers and resolvers play different roles). This situation, however, poses a dilemma: to send a request onto a DHT, a node must have joined the DHT, while every node on the DHT must relay DHT messages, manage part of the key space, and answer requests for keys in this space. On the one hand, letting routers join the DHT is clearly not an option, as they would fragment and capture part of the mapping space for which they would have no authority, nor sufficient knowledge, to answer requests. On the other hand, using a router's local mapping server (*i.e.*, the mapping server that manages the mappings for addresses local to the router's network) as a proxy onto the DHT poses a reliability issue. Indeed, if the local mapping server becomes unavailable, the corresponding ITRs would become unable to find

[1]To achieve good performance, Chord requires that all Chord nodes be roughly uniformly distributed around the ring. For simplicity, and without loss of generality, we assume that EIDs have their own entire address space directly mapped onto a full ring. If this is not the case, for instance if EIDs were to be allocated as a block of a wider address space, then the ChordID space could be chosen to be equivalent to the varying suffix of the EID addresses, to preserve the desired property. Other EID space to Chord space mapping are also possible, but these are not explored in the present paper.

new mappings (even when those mappings could reside on multiple mapping servers that are distant and alive).

To solve this problem, we propose to use the concept of Stealth DHT [1]. Stealth DHT separates the nodes on the DHT ring into two distinct types:

1. the service nodes, which behave like fully-fledged DHT nodes,
2. the stealth nodes, which can inject messages onto the DHT, but are never used for routing or key management.

Stealth DHT works on the observation that the joining procedure of a DHT is composed of two distinct phases:

1. a state gathering phase at the end of which the joining node will have received enough information (*e.g.*, routing, etc) to take part in the DHT;
2. an announcement phase through which the joining node advertises its presence to other nodes in the DHT and acquires part of the key space.

Service nodes complete both of these phases, while stealth nodes only complete the state gathering phase. In Chord, this would mean that stealth nodes would acquire successor and predecessor pointers, as well as finger tables, but they would never appear as predecessor, successor or in finger tables of any other nodes (service and stealth nodes alike). In other words, the stealth nodes, which in our case are ITRs/ETRs, do have all the necessary information to inject lookup messages directly into the DHT. This is a much more efficient solution than having ITRs/ETRs outside the DHT sending lookup messages only to the local DHT node, thus introducing a single point of failure. A stealth node will never receive any DHT requests or be used for relaying messages. As a result, they cannot store keys (*i.e.* EID-RLOC mappings) nor reply to any request sent into the DHT, and receive replies to their requests, directly from the destination mapping server (through the DHT infrastructure).

In an EID-to-RLOC mapping scenario, the role of the respective types of nodes can be enforced by the use of credentials. Without appropriate proof of identity (*e.g.* certificate of authority or of authority transfer) a node would only be allowed to complete the state gathering phase and join as a stealth node.

6. LISP-DHT ROBUSTNESS

For increased reliability (*cf.*, Section 1), several copies of a mapping should be somehow present on the Chord ring, in case the authoritative server for that mapping fails. The classical approach to this problem on DHTs is to replicate key-value pairs on nodes neighboring the node that owns them. However, such an approach would violate the first domain's requirement for mappings: the domain, who can't choose its neighbors on the ring, would loose control of who is managing its mappings.

As an alternative solution meeting the two main domain requirements (*i.e.*, control and redundancy), we propose to build redundancy into the DHT structure itself. We propose to let several Chord entities share a same ChordID and say that such entities form a redundancy group. In other words, all routing information, such as predecessor, successor and finger table information, must now be ready to accommodate and manage several chord entities per ChordID. These entities will be differentiated by their (EID, RLOC) address pairs, used to send IP packets.

When sending a message to a ChordID, a node can choose any of the Chord entities in the redundancy group to be the message recipient. Some form of topology awareness could, and should, be used to choose the nodes from a redundancy group in proximity order. If finer redundancy control is required, state information can even be associated with every member of the redundancy group to indicate when and how the member should be used. For instance, a surrogate state could indicate that the node will route and answer requests as ChordID at any time, while a back-up state could indicate that the node will only route and answer requests as ChordID when all the surrogate nodes have left the DHT. The first node of a redundancy group joins the Chord space as a "classic" Chord node (see Section 4 and Section 5). However, the join requests from subsequent redundancy group members trigger a demand for predecessor (resp. successor) update at the node that succeeds (resp. preceeds) the joining ChordID on the ring. Indeed, these nodes are aware that some redundant members are already present and should then send to the joining node a list of (some) other known nodes in the redundancy group (and the corresponding certificates proving the authority of the nodes (if need be), see C_{auth} above and C_{tran} below), along with the required routing information. The successor and predecessor of the joining node will add information about this node in their Chord routing information, but will refrain from deleting their existing information (existing Chord routing information is only changed when the joining node is the first to join at the specified ChordID).

In general, to facilitate management of structural information, members of a redundancy group should implement a distributed monitoring and structural information data exchange protocol, based on gossiping for instance. Through such a system, they not only cooperatively monitor each other's availability, but also propagate known changes to the redundancy group (such as the addition or removal of a member) and to the routing information (such as the addition of a new successor). The key-value pairs (in our case EID-to-RLOC mappings) are exchanged from the authoritative server for the EID block to members of its redundancy group by an out-of-band method.

In cases like EID-to-RLOC mappings, where tight control may be needed over which nodes can join as a

member in a redundancy group, a mechanism for node credentials will be required. Building on the concept of "certificate of authority" in Section 4, the authoritative server for an EID-prefix can issue members of its redundancy group with "certificate of authority transfer":

M_{tran} = (ChordID, member, C_{auth}, expiry);

C_{tran} = (M_{tran}, Epriv(server, H(M_{tran})));

where member is a description of the redundancy group member (*i.e.*, the <EID, RLOC> pair), and server the description of the authoritative server.

7. LISP MAPPING SYSTEMS

As stated in Section 1, there are already several mapping distribution systems and mapping query infrastructures proposed for LISP.[2] Here we review these systems highlighting their main features.

7.1 LISP-NERD

The simplest mapping distribution system proposed insofar is NERD (Not-so-novel EID RLOC Database [8]). NERD is based on a monolithic database, on each xTR[3], refreshed at regular intervals of time and containing all available mappings, which are assumed to be published by a centralized authority. This means that LISP-NERD follows a *push* distribution model, since it proactively "pushes" all available mappings toward all existing xTRs. On the one hand, this offers the advantage of reducing the signaling overhead, since LISP-NERD uses a HTTP-based incremental updates approach. On the other hand, LISP router needs to store all existing mappings, even the ones that are never used, putting a heavy limitation on the scalability of such an approach, since the database may grow to very large sizes. Furthermore, the bootstrap operation can be very long in time, since the whole database needs to be downloaded.

7.2 LISP-CONS

The *Content distribution Overlay Network Service* for LISP (or LISP-CONS [2]) operates on a distributed database hierarchically organized in a tree-like fashion. LISP-CONS is a hybrid push/pull approach. The EID-Prefixes (not the whole mapping) are "pushed" toward the root of the hierarchy. While propagating toward the root EID-Prefixes are aggregated. When a mapping is needed, this is "pulled" from the LISP-CONS hierarchy by sending a query. The message is recursively forwarded up into the hierarchy until a node knows who the owner of the mapping is, then the request is forwarded to the latter. The owner will reply directly. This design is similar to the DNS system, however, unlike DNS, LISP-CONS maintains a full mesh at each level in order to improve its responsiveness.

[2]Non-LISP solutions are out of the scope of this paper.

[3]By xTR we indicate a node that can be an ITR, an ETR, or both.

7.3 LISP-EMACS

The *EID Mappings Multicast Across Cooperating Systems* for LISP (or LISP-EMACS [3]) proposes to use the PIM [6] multicast protocol in order to build a set of trees through which distributing the mappings. An xTR joins an appropriate multicast group based on the owned EID-Prefix. Once joined, other xTR can pull the mapping from it by simply sending a request on the multicast group. LISP-EMACS proposes to forward packets, for which no mapping is available on the local cache, directly on the LISP-EMACS infrastructure. This will introduce some added delay for those packets, but avoids packet drops. The main issue with the LISP-EMACS approach is the complexity in putting in place and maintaining the multicast infrastructure and coordinate lookups on the several different trees build for the different sets of EID blocks.

7.4 LISP-ALT

The LISP *ALternative Topology* (or LISP-ALT [4]) proposes to use an overlay (the alternative topology) based on GRE tunnels among BGP routers that advertise EID-Prefixes. The basic approach is very similar to LISP-CONS, EID-Prefixes are announced on the overlay in a broadcast fashion, *i.e.*, they are pushed toward every node of the overlay, while performing aggregation when possible. Such a solution may raise scalability concerns since, like in NERD, each node of the overlay stores also information that is never used. In case of a missing mapping in the local cache, LISP routers can query the overlay and (similarly to LISP-EMACS) ask the overlay to deliver the packet on its behalf.

8. DISCUSSION

Among the above described proposals, none is based on DHTs. The one that is closer, in the design, to LISP-DHT is LISP-ALT, since both rely on an overlay. Nevertheless, LISP-ALT is totally unstructured and based on manual configuration of BGP, while LISP-DHT has the same ring structure of Chord, on which it is based. Thus, LISP-DHT maintains the properties of self-organization and robustness of Chord, offering several additional benefits when compared to LISP-ALT. LISP-DHT is a complete distributed database, where each node stores only its own mappings, thus not raising any scalability concern, while efficient lookups are possible thanks to the DHT infrastructure.

Table 1 summarizes the main features of all the existing mapping systems proposals, including LISP-DHT. The only proposal based on a full *push* approach is LISP-NERD, which pushes the whole mapping database toward all existing xTRs. LISP-CONS and LISP-ALT use a hybrid approach; the EID-Prefixes are pushed in the system, while the mapping is pulled on demand (*cf.*, third column Table 1). The fact that only EID-Prefixes

System	Distribution Model	Propagated Information	Aggregation	Sensitive to Churn
LISP-NERD	Push	Entire Mapping	No	No (updates follow a fixed time schedule)
LISP-CONS	Hybrid Push/Pull	EID-Prefix	Yes	Yes
LISP-EMACS	Pull	-	No	No
LISP-ALT	Hybrid Push/Pull	EID-Prefix	Yes	Yes
LISP-DHT	Pull	-	No	No

Table 1: Summary of existing Mappings Distribution/Lookup Systems.

are announced on the distribution infrastructure allows performing prefix aggregation, thus improving scalability and reducing the amount of signaling overhead. LISP-DHT and LISP-EMACS both use a full *pull* approach. Their infrastructure is built in such a way that the mapping request is automatically forwarded to the owner.[4] This means that no specific advertisement is performed (*cf.*, third column Table 1). This has the main advantage of not raising any churn issue. Other approaches, even partially based on push model, can suffer of such a problem since some updates can percolate globally. Further, since there is no specific information advertised on the distribution infrastructure, aggregation cannot be performed in the context of LISP-DHT and LISP-EMACS. This does not cause any issue in the case of LISP-DHT, since its DHT infrastructure is designed to be scalable. On the contrary, LISP-EMACS is based on a complex multi-tree infrastructure that can raise several issues to scalability and more than other approaches can suffer from churn problems.

Compared to all other approaches, LISP-DHT seems a promising solution, offering high scalability and self-management by design, thanks to the DHT paradigm. Further, the infrastructure is insensitive to churn, providing an efficient lookup mechanism.

9. CONCLUSION AND FUTURE WORK

Recent work in the research community has focused on the Locator/ID separation paradigm to improve the scalability of the current Internet. A critical task in such a context is the mapping system, necessary to maintain the binding between locators and identifiers. The present paper proposes a DHT-based mapping lookup system for the LISP protocol. To the best of our knowledge, this is the first work introducing DHTs in the context of LISP Locator/ID separation. On the one hand, the advantage of LISP-DHT is that robustness and scalability is naturally guaranteed by the DHT infrastructure. On the other hand, LISP-DHT has been designed to leave the control of the mapping to the domain that owns it. Future steps on LISP-DHT are the evaluation of the protocol and its performance in terms of reliability and effectiveness.

[4]Actually, for LISP-EMACS, the message is delivered to all nodes of the multicast group on which the query has been issued, but only the owner of the mapping will reply.

Acknowledgments

The research results presented herein have received support from Trilogy (http://www.trilogy-project.org), a research project (ICT-216372) partially funded by the European Community under its Seventh Framework Programme, and from a Cisco UFRP. The views expressed here are those of the author(s) only. Neither the European Commission, nor Cisco, are liable for any use that may be made of the information in this document.

Authors wish to thank O. Bonaventure, P. Francois, and D. Farinacci for their helpful comments on the work.

10. REFERENCES

[1] A. Brampton, A. MacQuire, I. A. Rai, N. J. P. Race, and L. Mathy. Stealth distributed hash table: unleashing the real potential of peer-to-peer. *Proceedings of the 2005 ACM conference on Emerging network experiment and technology (CoNEXT)*, 2005.

[2] S. Brim, D. Farinacci, V. Fuller, D. Lewis, and D. Meyer. LISP-CONS: A Content distribution Overlay Network Service for LISP. *draft-meyer-lisp-cons-04.txt*, IETF Network Working Group, April 2008.

[3] S. Brim, D. Farinacci, D. Meyer, and J. Curran. EID Mappings Multicast Across Cooperating Systems for LISP. *draft-curran-lisp-emacs-00.txt*, IETF Network Working Group, November 2007.

[4] D. Farinacci, V. Fuller, and D. Meyer. LISP Alternative Topology (LISP-ALT). *draft-fuller-lisp-alt-03.txt*, IETF Network Working Group, October 2008.

[5] D. Farinacci, V. Fuller, D. Oran, and D. Meyer. Locator/ID Separation Protocol (LISP). *draft-farinacci-lisp-09.txt*, IETF Network Working Group, October 2008.

[6] M. Handley, I. Kouvelas, T. Speakman, and L. Vicisano. Bidirectional Protocol Independent Multicast (BIDIR-PIM). RFC 5015, October 2007.

[7] G. Huston, G. Michaelson, and R. Loomans. A Profile for X.509 PKIX Resource Certificates. *draft-ietf-sidr-res-certs-14.txt*, IETF Network Working Group, October 2008.

[8] E. Lear. NERD: A Not-so-novel EID to RLOC Database. *draft-lear-lisp-nerd-04.txt*, IETF Network Working Group, April 2008.

[9] T. Li. Design Goals for Scalable Internet Routing. *draft-irtf-rrg-design-goals-01.txt*, IETF Network Working Group, July 2007.

[10] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang. IPv4 Address Allocation and the BGP Routing Table Evolution. *Computer Communication Reviews*, 35(1):71–80, 2005.

[11] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984, IETF Network Working Group, September 2007.

[12] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure. Evaluating the Benefits of the Locator/Identifier Separation. *Proc. 2nd ACM SIGCOMM Workshop on Mobility in the Evolving Internet Architecture (MobiArch)*, August 2007.

[13] I. Stoica, R. Robert, D. Liben-Nowell, D. David, M. Frans, F. Frank, and H. Hari. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003.