Designing a Deployable Future Internet: the Locator/Identifier Separation Protocol (LISP) case

Damien Saucez - Luigi Iannone - Olivier Bonaventure - Dino Farinacci

***Abstract*—The Internet has been created for interconnecting few hundreds networks, but is now close to one billion hosts, grouped in 40,000 Autonomous Systems, using more than 400,000 prefixes. Such a situation raises scalability issues that have driven both academia and industry to review the current Internet Architecture in the light of the Locator/Identifier Split paradigm. In particular, the Internet Engineering Task Force (IETF) has adopted and is actively designing and developing the Locator/ID Separation Protocol (LISP). However, changing the routing and addressing architecture of the Internet in an incrementally deployable manner Several constraints impact such a design. We use LISP as reference to describe the different design choices necessary to achieve deployability, which is the ultimate goal of any new Future Internet architecture. Furthermore, we showcase several alternate usages of LISP, which go beyond improving the Internet scalability.**

I. Introduction

In the last years both academia and industry have worked toward new Internet architecture proposals, due to the awareness that the current architecture is facing unforeseen scalability issues [1], concerning the restless increase of the BGP routing tables [2], addressing, mobility, multihoming, and inter-domain traffic engineering. The general consensus is that splitting the locator and identifier roles of IP addresses solves these issues and is necessary for the Future Internet architecture [3]. However, in practice, several constraints have to be taken into account in order to design a viable solution that can be incrementally deployed, without disrupting the existing communication infrastructure, whilst providing benefits, hence incentives, for early adopters [4].

An instance of such a paradigm is the Locator/ID Separation Protocol (LISP). LISP was first proposed by Cisco in the IRTF (Internet Research Task Force) and is now under development in the IETF (Internet Engineering Task Force). Aiming at being incrementally deployable, LISP has evolved from its initial design in order to accommodate the constraints that the current Internet imposes, but still offering an effective solution for the scalability issues. For these reasons, we use LISP as reference to explore the design trade-offs that the design of any new Locator/ID Split based architecture has to consider.

Our goal is not to convince the reader about the merits of LISP, or the general Locator/ID Split paradigm, and neither to provide numerical results of its performance. Such aspects have been tackled in different research works and IETF discussions, motivating the necessity of such a solution [1], [3], [5], exploring scalability aspects [6], [7], [8], [9], and possible adoption paths [4]. Rather, this paper aims at exploring the design space and the constraints that shape such a space. To this end, this paper is three-fold and is distributed in its three main parts.

In the first part of the paper, we describe the main design goals for any deployable solution that separates locators and identifiers (Section II), aiming at understanding, but also defining, what are the most important deployability constraints.

In the second part of the paper we provide a concrete example of a protocol that achieves such goals. Hence, we overview how LISP works (Section III), focusing on the basic knowledge necessary to understand the different trade-offs of its design. It is not in the scope of this paper to provide a full description of LISP in every detail, rather we focus on its main architectural aspects, exploring the different design choices, their rationale with respect to the design goals, and their implications.

The success of a protocol is defined by its use in contexts for which it has not been designed for, rather than its use for what it has been designed for [10]. Hence, in the last part of the paper (Section IV), we explore the use of LISP as a framework to support either relatively new technology (*e.g.*, virtualization, IPv6 transition) or known problems that lack of a final solution (*e.g.*, multihoming, mobility). Such use-cases go beyond the original LISP design and are very important for early adopters.

II. Design Goals of a Deployable Future Internet

The technical merits that a protocol/technology may have, does not guarantee widespread Internet adoption [11], unless there is a clear deployment path that also provides benefits for early adopters.

The first and foremost goal while designing a new protocol is that it has to be *incrementally deployable*. While sometimes this is interpreted as backward compatibility, it is not exactly the same thing. A solution can be incrementally deployed when there is no need for different Internet stakeholders to agree on deployments and a flag day to switch over the new technology, nor there is a need of a minimum critical mass of adopters for the solution to correctly work. Rather, every independent stakeholder can decide to deploy the protocol for its own purpose, obtaining initial benefits. This leads to the issue of guaranteeing interoperability with the existing architecture, in order to allow end-to-end communication between early adopters and legacy systems.

Another important goal while designing new protocols is their capability of *core network transit and middlebox traversal*. In other words their packets have to be able to transit through the core Internet like any other packet, without requiring any special handling procedure. Similarly, packets must be able to traverse any type of middleboxes, which are more and more deployed in the Internet, without being discarded. In both cases the aim is to maintain full Internet connectivity.

An important question that arises when exploring the design space of any protocol is how to *limit the number of affected systems*. How many systems in the Internet need to be upgraded in order to deploy the new solution? Can this number be *limited and as small as possible*? There is a big difference between requiring the upgrade of every end-system to the new solution and being able to achieve the same (or equivalent) result just modifying a few boxes (*e.g.*, border routers). On the one hand, this has direct impact on the deployment cost, on the other hand, it might create tussles between different stakeholders (*e.g.*, end-user waiting for their service provider to upgrade, while the latter is waiting for the former.

For every solution touching the addressing and routing Internet architecture it is important to consider the *namespace* on which to base the new architecture, especially when separating the locator and identifier semantic in different namespaces. After more than 30 years of evolution, the Internet hardware and software are extremely efficient in dealing with IP addresses. Introducing a namespace with a totally different syntax increases deployment costs, reduces efficiency, and slows down packet processing because new hardware and software (usually less efficient) has to be developed.

Since the Locator/ID Split paradigm is based on two different namespaces, mappings between the two are necessary in order to guarantee end-to-end communication. This implies the need of a new service able to provide those mappings. Such a *Mapping System* infrastructure however needs to avoid any form of technology lock-in. Meaning that it has to be *open* and ready to future evolution.

We can summarize the design goals as follows:

1) Incremental deployability
2) Core network transit and middlebox traversal
3) Limited number of affected boxes
4) Non-disruptive namespace
5) Lock-in free mapping system solution

To make these goals less abstract, in the following we describe the operation and main features of LISP, highlighting at each step how these goals have shaped its design.

III. Achieving the Design Goals: the LISP Case

Routing and addressing architectures based on separating the identity of end-systems from their location in the Internet topology was already discussed in the mid-90s [12]. However, it is only in the last years that real proposals, including LISP, have been designed, under the pressure of the concerns about Internet scalability. LISP, in particular, falls in the map-and-encap class of solutions, since it relies on three simple principles: *address role separation*, *encapsulation*, and *mapping* [13], [14]. Hereafter, we describe these three principles, highlighting how they fulfill the provided design goals. We then provide a description of how LISP works.

A. Splitting the IP Addressing Space

In LISP, address role separation is achieved by splitting the semantic of IP addresses in two categories: the *Routing LOCators (RLOCs)* and the *Endpoint IDentifiers (EIDs)*. RLOCs are assigned to border routers from the *RLOC Space* of Internet Service Providers. EIDs are assigned in blocks extracted from the *EID Space* to stub networks. A stub network, also called edge network, is a network that only carries traffic from and to itself, *e.g.*, enterprise or campus network. A visual example of what the Internet looks like when LISP is deployed, which highlights as well where RLOCs and EIDs are used, is depicted in Figure 1. Re-using the IP address space allows LISP to fulfill goals 1) and 4). Indeed, it is just a slight, non-disruptive, change in the IP address semantic, that is totally inter-operable with any IP-based device. Furthermore, this is the premise to fulfill goal 2) since LISP is still based on the IP technology.

While simple, the IP address semantic change is also at the very core of how LISP solves the scalability problem of today's Internet. With LISP only the routes towards the RLOCs are announced in the BGP routing infrastructure. On the contrary, in today's' Internet, EIDs (*i.e.*, IP prefixes of stub networks) are also announced. These prefixes are largely responsible of the Routing Information Base bloat [3].

B. Core-Edge Separation Through Encapsulation

With LISP, routers in the core of the Internet are not aware anymore about EIDs addresses. Therefore, a tunneling approach is used in order to forward packets between edge networks. The tunneling is done by encapsulating the original packet in an UDP segment that includes a LISP-specific header. While the inner (original) IP header uses EID addresses, the outer IP header uses addresses from the RLOC space. The UDP header sets the destination port to the well-know value 4341, so to uniquely identify LISP-encapsulated packets. The LISP-specific header contains flags and other information related to traffic engineering and RLOCs' reachability.[1]

The encapsulation is performed by border routers of the packets' source site, the so-called *Ingress Tunnel Routers* (ITRs), while the decapsulation operation is carried out by border routers of packet's destination site, the so called *Egress Tunnel Routers* (ETRs). Note that forwarding inside the source and destination sites is performed using the EIDs addresses of the original (un-encapsulated) packet. Such design leads to the very nice property that only these tunnel border routers (generally referred to as xTRs) have to support LISP. Indeed inside the local domain routing and forwarding is done as usual, with the peculiarity that IP addresses are from the EID space. Similarly, in the core Internet, routing and forwarding is done as usual, with the peculiarity that LISP-encapsulated packets use IP addresses from the RLOC space. In this way goal 3) is achieved, since only a very limited number of boxes (compared to total number of devices that form the Internet)

[1]As already mentioned, it is out of the scope of this paper to provide a complete and detailed description of LISP. The interested reader can find this information in the original LISP specification [13].

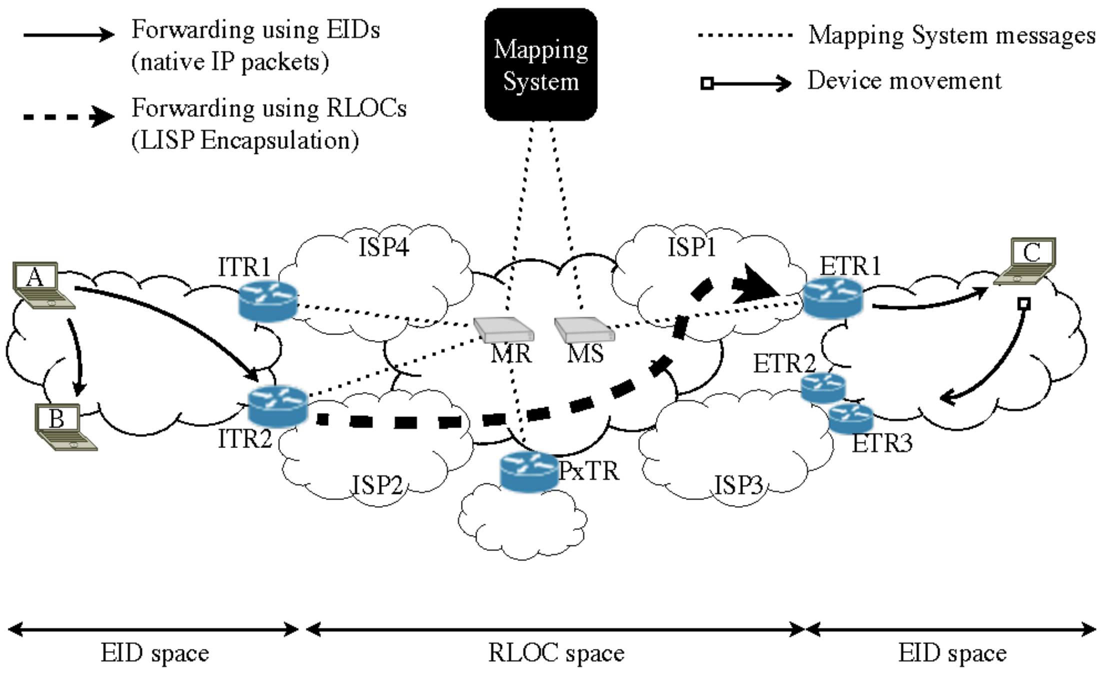

Fig. 1. LISP operation

need to be modified. Even more, there is no need for hardware upgrades; LISP functionality can be achieved by a simple software upgrade.

Once forwarded in the core Internet, LISP-encapsulated packets look like any other UDP packets, achieving goal $2)$. Indeed, UDP packets transit in the core and traverse middle-boxes. This point is very important as middle boxes are likely to refuse, because of security reasons, to process datagrams with unknown features such as protocol numbers, IP options, or addressing syntax.

C. *Mapping IDs to Locators*

Due to the usage of tunneling, and in order to actually achieve end-to-end communication, it is necessary to *bind* EIDs with RLOCs, basically to know the locators (RLOCs) that allow to reach any end-system (EIDs). Such bindings are the so-called *mappings*, and are distributed by a Mapping System. Differently from the current routing infrastructure where routing information is distributed by BGP to all routers, the LISP Mapping System works like the DNS. It uses an on-demand approach where the Mapping System is queried for specific EIDs, returning all related mappings.

LISP defines a general front-end toward the Mapping Systems consisting of two types of servers: *Map-Servers* (MS) and *Map-Resolvers* (MR) [15]. Map-Servers are used to make the Mapping System aware of the existence of mappings. In particular ETRs send *Map-Register* messages in order to register their mappings in the Mapping System. Upon successful registration, the MS replies with a *Map-Notify* message to confirm the operation to the ETR. Map-Resolvers are instead queried by the ITRs, by sending *Map-Request* messages, in order to retrieve mappings. Depending on whether or not the Map-Resolver works in proxy-mode, either the MR or the destination ETR will eventually reply with a *Map-Reply* message containing the requested mapping.

This mapping system front-end is the cornerstone of achieving goal $5)$. Indeed, this allows to nicely separate the data-plane (*i.e.*, packets' encapsulation and forwarding) from the control-plane (*i.e.*, mapping registration and distribution). In this way the mapping system infrastructure has an open architecture allowing easily changing mapping system or make multiple instantiations of different mapping systems to work in parallel, avoiding technology lock-in. For example, the first mapping system was built on top of BGP [16] but has been seamlessly replaced by a DNS-inspired solution [17], [9].

D. *The Big Picture*

To better clarify how address role separation, encapsulation, and mapping principles all work together in LISP, let us analyze in Figure 1 how the end-system A can send a packet to end-system C. Initially, A sends a normal IP packet using A's EID as source address and C's EID as destination address. The packet is forwarded until it reaches one of the ITRs, let us suppose it reaches $ITR2$. $ITR2$, maintains in its LISP-Database the *local* mappings, *i.e.*, the mappings for the EIDs that are in the local domain. $ITR2$ thus selects a source RLOC for the encapsulation. To obtain the RLOC to be used as destination address to reach C, $ITR2$ sends a Map-Request message to MR. Assuming that the latter is working in proxy-mode, it will send back a Map-Reply containing the requested mapping. At this point $ITR2$ encapsulates the packet and forwards it through the core Internet. The mapping for C is stored in the LISP-Cache in order to speed up encapsulation and forwarding for subsequent packets.[2] Once the encapsulated packet reaches one of C's ETRs, *e.g.*, $ETR1$, the packet is decapsulated and finally forwarded to C.

As previously pointed out, this approach works smoothly with routers that are not LISP-aware, respecting all of the five goals presented in Section II. In particular, end-systems and the core Internet do not need whatsoever change. The only case to pay attention to is when LISP sites need to communicate with non-LISP sites and vice-versa. This is however already included in the original LISP design [18] which provides an incrementally deployable solution from every point of view.

IV. Beyond the Design Goals: alternative LISP use-cases

Previous sections presented how LISP design goals for a better Internet scalability have influenced the final design of the protocol and its operation. This section showcases how LISP goes beyond Internet scalability, providing an elegant solution to several networking problems not directly related to the Internet scalability. These representing important benefits for early adopters. The proposed use-cases are not meant to be new, but rather to provide ground information on the benefits of Locator/ID split in different contexts. More than simply providing improved multihoming and traffic engineering capabilities to IP networks, LISP offers new perspectives for known scenarios. The map-and-encap nature of LISP makes it a flexible tool and a perfect candidate for supporting the transition to IPv6 (*cf.* Section IV-A), network virtualization (*cf.* Section IV-B), seamless virtual machine mobility in datacenters (*cf.* Section IV-C), and even device mobility in the Internet (*cf.* Section IV-D).

A. LISP for IPv6 Transition

The LISP encapsulation mechanism is designed to support any combination of locators and identifiers address family. It is possible to bind IPv6 EIDs with IPv4 RLOCs and vice-versa. This allows to transport IPv6 packets over an IPv4 network (or IPv4 packets over an IPv6 network), thus enabling the use of LISP as an IPv6 transition mechanism.

A not so uncommon example is the case of the network infrastructure of a datacenter being IPv4-only while dual-stack front-end load balancers are used. In this scenario, LISP can be used to provide IPv6 access to servers even though the network and the servers only support IPv4. Indeed, assuming that the datacenter's Internet Service Provider (ISP) offers IPv6 connectivity, the datacenter only needs to deploy one (or more) xTR(s) at its border with the ISP and one (or more) xTR(s) directly connected to the load balancers. The xTR(s) at the ISP's border tunnels IPv6 packets over IPv4 to the xTR(s) directly attached to the load balancer. The load balancer's xTRs decapsulate the packets and forward them to the load balancers, which act as proxies, translating each IPv6 packet into an IPv4. IPv4 packets are then sent to the appropriate servers. Similarly, when the server response arrives at the load balancer, the packet is translated back into an IPv6 packet and forwarded to its xTR(s), which in turn will tunnel it back, over the IPv4-only infrastructure, to the xTR(s) connected to the ISP. The packet is then decapsulated and forwarded to the ISP natively in IPv6.

Let us consider that Figure 1 represents a datacenter network instead of the Internet and that clouds represent subnetworks instead of ASes. Let us consider end-system C being a load balancer, connected to a pool of servers, and the domain it belongs to being IPv4-only. Let us also consider end-system A being part of an IPv6-only domain. When packets from A enter the datacenter via $ITR2$, they are encapsulated in IPv4 LISP packets and forwarded on the IPv4 datacenter infrastructure network, until they reach the xTR just in front of C, where they are decapsulated and forwarded in IPv6 to the load balancer C. The latter translates the packets into IPv4 and forwards them to the pool of servers.

B. LISP for Network Virtualization

Network virtualization enables operating several logical networks over one physical network. Provider-based *Virtual Private Networks (VPN)* are a way of virtualizing networks, often using BGP/MPLS VPNs to offer VPN services to enterprises [19]. Nevertheless, BGP/MPLS VPN is complex to configure and maintain. LISP, with its map-and-encap mechanism, can be used to replace BGP/MPLS VPN, with the encapsulation part of LISP playing the role of MPLS and the mapping part the one of BGP.

To support network virtualization, it is necessary to tag the packets so that routers can determine to which VPN instance a packet belongs. Labels in MPLS provide this functionality. LISP supports such tagging of packets and mappings with the *Instance ID* (typically a 802.1Q VLAN tag) in the LISP-specific header. Every packet or mapping having the same Instance ID belongs to the same VPN. When a packet arrives at an ITR, this discriminates the packet to determine its virtual network. The ITR then sends a Map-Request, for the destination EID, tagged with the Instance ID of the related virtual network. The Mapping System returns the mapping associated to the destination EID, as it appears in the LISP instance identified by the Instance ID. The packet is then encapsulated using the retrieved mapping and tagged as well with the Instance ID. The Instance ID is not used when forwarding the encapsulated packet in the core Internet. Only when the packet is decapsulated by an ETR, the Instance ID is used to determine the network where to forward the decapsulated packet (*e.g.*, its corresponding VLAN).

[2]LISP does not specify what to do with the first packet for which no mapping is available, current implementations silently drop it.

Actually, beyond the IP-over-IP VPN, the *LISP Canonical Address Format (LCAF)* [20] allows to represent and use any address family over LISP (*e.g.*, Ethernet). The coupling of Instance ID with LCAF extents the range of network virtualization scenarios, enabling virtualization of layer two networks. One can then imagine deploying an Ethernet network over IP to, for example, leverage multipath and overcoming the limitations of the Spanning Tree Protocol.

C. LISP for Virtual Machine Mobility in Datacenters

LISP enables seamless mobility of *virtual machines* (VM) in datacenter networks. To this end, IP addresses of the routers in the backbone of the datacenter form the RLOC addressing space, while IP prefixes of the subnet where virtual machines are placed form the EID addressing space. Routers at the edge between the backbone and the subnets run LISP and operate as xTRs. Differently from the Internet wide LISP deployment, in this case every xTR is assigned a multicast address, with xTRs that potentially host the same set of virtual machines belong to the same multicast group. Such setup has the advantage that no change is necessary on the virtual machines, only routers at the edge of the subnets must be upgraded to support LISP. As last piece, the datacenter maintains a local Mapping System accessed through a Map-Server and a Map-Resolver.

For the sake of illustration, let us consider that Figure 1 represents a datacenter network. When a VM (*e.g.*, C) moves from one subnet to another, the xTRs in the new subnet (*i.e.*, $ETR2$ and $ETR3$) discover the presence of a new virtual machine by continuously monitoring the subnet. Then, the xTR registers this change by sending a Map-Register message to the Map-Server (*i.e.*, MS), binding the EID address of C (*i.e.*, the newly arrived VM) to the RLOC addresses of the xTRs (*i.e.*, $ETR2$ and $ETR3$). The Map-Server replies with a Map-Notify message. As a multicast address is used for the registration, the Map-Notify is received by all the xTRs of the same group, including the xTRs where the VM was previously located (*i.e.*, $ETR1$, $ETR2$, and $ETR3$). When an xTR receives such a notification, it determines whether or not it was one of the previous locators. If it is the case, the xTR proactively informs the xTRs communicating with it that the mapping has changed (*e.g.*, $ETR1$ notifies $ITR2$ because there is traffic between A and C). This operation is performed by sending a Map-Request with the SMR (Solicit-Map-Request) bit set, causing the xTR to update its locally stored mapping. If an xTR receives packets for a VM that moved, it can re-encapsulate the packet to send it to the appropriate xTR. In parallel, the xTR sends again a Map-Request with the SMR bit set, to notify the xTR that originated the packet that the mapping has changed.

To allow packets to be exchanged between the VM and the rest of the Internet, specific xTRs are placed at the edge of the datacenter, playing the role of proxy between the non-LISP Internet and the LISP datacenter. When a packet arrives from the Internet, it is encapsulated and sent to an xTR serving the VM identified by its IP address. When a packet from a virtual machine is sent to the Internet, the xTR of the VM encapsulates the packet and sends it to an xTR at the edge of the datacenter. This xTR decapsulates the packet and forwards it to the Internet.

As for the VPN case, the use of LCAF extends even more the LISP mobility support features, since it is possible to support VM's mobility based on their layer two (MAC) address instead of their IP address.

D. LISP for Device Mobility in the Internet

This section, overviews how LISP can provide node mobility without the help of the network it is visiting (differently from current IP mobility solutions). In LISP Mobile Node (LISP-MN [21]), the device itself implements a lightweight version of LISP. Every mobile node receives an EID address from its home network and keeps this EID independently of its location. Mobile nodes also receive addresses that belong to the foreign network they are visiting. These addresses are used as RLOCs. A new mapping is registered to the Map-Server of its home network by the mobile node each time it moves and changes RLOCs (*i.e.*, visited network). The mappings bind the EID of the mobile device to the RLOCs received from the visited network. The Map Server does not need to advertise the EID address to the mapping system as it belongs to the less specific prefix it already advertises. The mobility is thus transparent for the mapping system, ensuring scalability. To avoid triangular routing and the necessity of a home agent, the mobile device can send Map-Request with the SMR bit set to the nodes it is communicating with. The other end of the communication then updates its mapping so to send the packets directly to the foreign network locator. Similarly, if a node wants to contact the mobile node, it just needs to retrieve the mapping from the Map-Server of the home network (through a normal Map-Request to the mapping system), then it can directly send packets using the foreign network locator.

The mobile node's network stack is extended to implement LISP such that only the EID address is visible to applications. When a packet is sent to the mobile node, it is LISP encapsulated, the locator being the address in the foreign network and the identifier the EID of the node. When the device receives an encapsulated packet, it decapsulates it and delivers the inner packet to the higher layers in the protocol stack. Packets sent by the mobile node are encapsulated in order to traverse the foreign network, as this last may block packets that do not have a source address from the network. When one of the two nodes involved in the communication does not implement LISP, the packets are processed by a proxy xTR [18], as for any transition between non-LISP and LISP sites.

V. Conclusion

In more than forty years of history, the Internet has evolved from a research project to an every day commodity service. This tremendous growth forces the current routing and addressing infrastructure to face unforeseen scalability issues. While, on the one hand, new paradigms like Locator/ID Split, are able to increase the scalability of the Internet, on the other hand, their adoption is hindered by the strict deployment constraints imposed by the ossification of the current architecture.

Differently from other proposals, LISP (Locator/ID Separation Protocol) has been developed following specific design goals, addressing the main constraints and other incidental requirements. In this paper we have first explained these design goals that can be used as guidelines for the design of any new technology for the Future Internet. Then we described the design of LISP and its functioning, emphasizing how the design goals have shaped its development. By re-using as much as possible the current IP technology, trying as well to reduce the impact on the existing infrastructure, LISP offers all the benefits of the Locator/ID Split paradigm while being incrementally deployable. Indeed, there exists already a worldwide deployment, the `www.lisp4.net` network, which includes partners from both the academia and the industry. Furthermore, we explored multiple uses of LISP, in scenarios beyond the original set of use-cases, showing how LISP can be used for IPv6 transition, network virtualization, and mobility.

REFERENCES

[1] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing," RFC 4984 (Informational), Internet Engineering Task Force, Sep. 2007.

[2] BGP Routing Table Analysis Report. [Online]. Available: http://bgp.potaroo.net

[3] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure, "Evaluating the benefits of the locator/identifier separation," in *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving Internet Architecture (MobiArch'07)*, Aug. 2007.

[4] L. Iannone and T. Levä, "Modeling the Economics of Loc/ID Split for the Future Internet," in *Towards the Future Internet – Emerging Trends from the European Research*. IOS Press, May 2010.

[5] T. Li, "Recommendation for a Routing Architecture," RFC 6115 (Informational), Internet Engineering Task Force, Feb. 2011.

[6] D. Saucez, B. Donnet, L. Iannone, and O. Bonaventure, "Interdomain Traffic Engineering in a Locator/Identifier Separat ion Context," in *Proc. of Internet Network Management Workshop (INM'08)*, Oct. 2008.

[7] L. Iannone and O. Bonaventure, "On the Cost of Caching Locator/ID Mappings," in *Proceedings of the 2007 ACM CoNEXT conference (CoNEXT'07)*, Dec. 2007.

[8] J. Kim, L. Iannone, and A. Feldmann, "Deep Dive into the LISP Cache and What ISPs Should Know about It," in *IFIP International Conference on Networking (Networking'11)*, May 2011.

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

[10] D. Thaler and B. Aboba, "What Makes For a Successful Protocol?" RFC 5218 (Informational), Internet Engineering Task Force, Jul. 2008.

[11] M. Handley, "Why the internet only just works," *BT Technology Journal*, vol. 24, pp. 119–129, Jul. 2006.

[12] R. Hinden, "New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG," RFC 1955 (Informational), Internet Engineering Task Force, Jun. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1955.txt

[13] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID Separation Protocol (LISP)," draft-ietf-lisp-23.txt, Internet Engineering Task Force, May 2012, Work in Progress.

[14] D. Meyer, "The locator identifier separation protocol (LISP)," *Internet Protocol Journal*, vol. 11, no. 1, pp. 23–36, Mar. 2008.

[15] D. Farinacci and V. Fuller, "LISP Map Server Interface," draft-ietf-lisp-ms-16.txt, Internet Engineering Task Force, Mar. 2012, Work in Progress.

[16] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "LISP Alternative Topology (LISP+ALT)," draft-ietf-lisp-alt-10.txt, Internet Engineering Task Force, Dec. 2011, work in progress.

[17] V. Fuller, D. Lewis, and D. Farinacci, "LISP Delegated Database Tree," draft-fuller-lisp-ddt-01.txt, Internet Engineering Task Force, Mar. 2012, work in progress.

[18] D. Lewis, D. Meyer, D. Farinacci, and V. Fuller, "Interworking LISP with IPv4 and IPv6," Internet Engineering Task Force, Internet Draft (Work in Progress) draft-ietf-lisp-interworking-06.txt, Mar. 2012.

[19] E. Rosen and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)," RFC 4364 (Proposed Standard), Internet Engineering Task Force, Feb. 2006, updated by RFCs 4577, 4684, 5462.

[20] D. Farinacci, D. Meyer, and J. Snijders, "LISP Canonical Address Format," draft-farinacci-lisp-lcaf-07.txt, Internet Engineering Task Force, Mar. 2012, Work in Progress.

[21] D. Farinacci, D. Lewis, D. Meyer, and C. White, "LISP Mobile Node," draft-meyer-lisp-mn-07.txt, Internet Engineering Task Force, Apr. 2012, Work in Progress.