LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System

Loránd Jakab, Albert Cabellos-Aparicio, Florin Coras, Damien Saucez, and Olivier Bonaventure

***Abstract*—During the last years, some operators have expressed concerns about the continued growth of the BGP routing tables in the default-free zone. Several proposed solutions for this issue are centered around the idea of separating the network node's identifier from its topological location. Among the existing proposals, the Locator/ID Separation Protocol (LISP) has seen important development and implementation effort. LISP relies on a mapping system to provide bindings between locators and identifiers. The mapping system is a critical protocol component, and its design is still an open issue. In this paper we present a new mapping system: LISP-TREE. It is based on DNS and has a similar hierarchical topology: blocks of identifiers are assigned to the levels of the hierarchy by following the current IP address allocation policies. We also present measurement-driven simulations of mapping systems' performance, assuming a deployment of LISP in the current Internet.**

***Index Terms*—Routing scalability, locator/identifier split, mapping system.**

I. Introduction

An important problem of today's Internet is the continued growth of the BGP routing tables in the default-free zone (DFZ) [1]. Besides the increasing number of Autonomous Systems, other factors contribute to this growth, including multihoming and traffic engineering [2]. This issue was ranked in 2006 at the Internet Architecture Board Workshop on Routing and Addressing [2] as *"the most important problem facing the Internet today"*.

Many of the proposed solutions to address this issue are centered around the idea of separating the network node's identity from its topological location. Among the existing proposals [3], [4], [5], [6] the Locator/Identifier Separation Protocol (LISP) [7] has seen important development and implementation effort. LISP considers two different types of addresses: *Endpoint Identifiers (EIDs)* and *Routing Locators (RLOCs)*. EIDs are allocated to sites in a provider-independent manner, but they are not advertised in the global Internet. The global BGP routing tables will eventually only contain the RLOCs; it would then be possible for these to be assigned in such a way that transit network providers can highly aggregate them, and help scale the BGP routing tables. Finally, a mapping system must be used to map the EIDs on the RLOCs that allow reaching them.

To date, several mapping systems have been proposed [8], [9], [10], [11]. LISP+ALT [8] is a hierarchical architecture that uses BGP to pass information, including requests for mappings among the nodes of the mapping system; it relies on aggregation to scale. LISP+ALT uses an overlay BGP network where each AS announces its EIDs through the overlay. On the other hand, LISP-DHT [9] is a P2P-based solution that stores bindings between EIDs and RLOCs on a Chord-like overlay [12], where each border router acts as a node of the overlay.

In this paper we propose a new mapping system: LISP-TREE. It is based on the widely used Domain Name System (DNS), with a similar hierarchical organization. Blocks of EIDs are assigned to the layers of the naming hierarchy by following the current allocation rules for IP addresses. The root of the naming hierarchy is maintained by the Regional EID Registries, which allocate EID blocks to local registries. These in turn maintain delegation information for the owners of the provider independent EID prefixes. Levels can be dynamically added to the hierarchy. LISP-TREE nodes can use existing DNS implementations, and benefit from the long operational experience, but to avoid interference with the current domain name system, LISP-TREE should be deployed on a *physically* separate infrastructure. One of the main advantages of LISP-TREE is DNS' proven scalability track record: for instance, at the time of this writing, the *.com* top-level domain stores roughly 77 million entries [13], while there are only around 320,000 entries in the default-free zone today [1].

In addition to its architectural qualities, the performance of mapping systems plays a key role in LISP. However, apart from a first trace-driven analysis of the mapping cache [14], no research efforts have been devoted to this goal. This paper aims to fill this gap as well, presenting a trace-driven simulation of the performance of LISP's main mapping systems. In particular we consider LISP+ALT, LISP-DHT and LISP-TREE. Our performance analysis assumes a complete deployment of LISP in today's Internet. We use topology and latency data from the iPlane [15] infrastructure and build the open source CoreSim simulator [16] on top of that. This setup provides estimations of LISP performance metrics.

The rest of the paper is structured as follows: Section II gives an overview of LISP and of the mapping systems under study. It is followed, in Section III, by a description

Loránd Jakab, Albert Cabellos-Aparicio and Florin Coras are with the Department of Computer Architecture, Universitat Politècnica de Catalunya, Barcelona, Spain (e-mail: {ljakab,acabello,fcoras}@ac.upc.edu). Damien Saucez and Olivier Bonaventure are with the Department of Computer Science and Engineering, Université catholique de Louvain, Louvain-la-Neuve, Belgium (e-mail: firstname.lastname@uclouvain.be).

This work has been partially supported by the Ministry of Innovation, Universities and Enterprise of the Generalitat of Catalonia under scholarship 2006FI-00935 and contract 2009SGR-1140, the Spanish Ministry of Science and Technology under contract TEC2009-13252, a COST TMA STSM and a Cisco URP grant. We would like to thank Noel Chiappa for his valuable suggestions that greatly improved this paper. Many thanks also to Harsha V. Madhyastha for the help with the iPlane latency lookup service and to Pere Barlet-Ros for providing the UPC traffic trace.

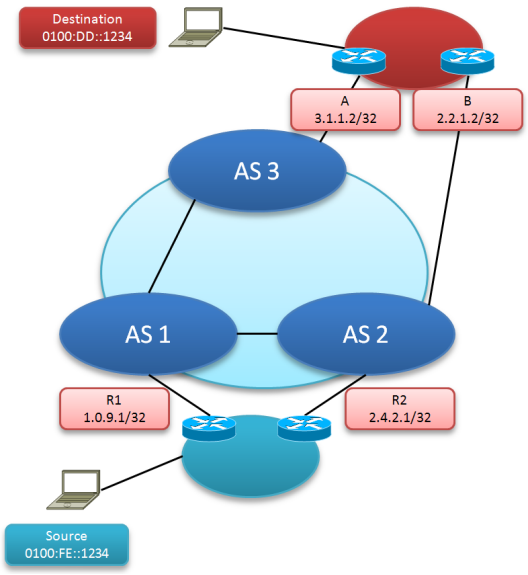


Fig. 1. Simple LISP scenario

of LISP-TREE, our proposed mapping system. Next, Section IV presents CoreSim, the simulator used to evaluate the performance of the main LISP mapping systems. The simulation results are presented and discussed in Section V. Finally Section VI includes the related work while Section VII concludes the paper.

II. LISP Background

This section describes briefly the main architectural blocks of LISP and gives an overview of the existing mapping systems.

A. The Locator/ID Separation Protocol

To forward packets between endhosts, LISP relies on mapping and encapsulation. All LISP sites communicate with the rest of the Internet through at least one pair of Ingress and Egress Tunnel Routers (resp., ITR, ETR), each globally identifiable by one or more routing locators (RLOCs). Though functionally different entities, the two types of tunnel routers could be implemented on the same device. For example, in Fig. 1, the bottom site has two xTRs: `R1` and `R2`. In this example, we use IPv6 addresses as end host identifiers (EIDs) and IPv4 addresses as RLOCs. One important feature of LISP is that it scales by not advertising the EIDs – belonging to the `0100::/8` prefix for the example – in the legacy Internet BGP routing tables, after the initial transition period.

When an endhost in the bottom LISP site in Fig. 1, e.g. `0100:FE::1234`, needs to contact a remote endhost, e.g. `0100:DD::1234` it sends a normal (IPv6 in this case) packet with the destination EID as destination address. This packet is intercepted by one of the site's ITRs (i.e., R1 or R2). To forward the packet, the ITR needs to obtain at least one of the RLOCs of the destination ETR. For this, the ITR queries the mapping system by sending a Map-Request packet. Several mapping systems [8], [9], [10], [11] are being developed for LISP (see Sec. II-B). The mapping system will respond with a Map-Reply which contains a set of RLOCs of the ETRs which the ITR can use to forward packets to the destination. A priority and a weight are associated with each RLOC to allow a LISP site to control its ingress traffic. RLOCs with the lowest priority value are preferred. Load balancing between the same priority RLOCs is achieved with the weight. A time-to-live (TTL) is also associated to these mappings, indicating their maximum lifetime in a cache. A mapping can be evicted from a cache before its TTL expires, if so required by the particular cache's eviction policy, but must be either refreshed or deleted when its maximum lifetime is reached. Furthermore, the mapping system will usually return a mapping that is valid for the entire prefix containing the destination EID. For example, in Fig. 1, the mapping system could return a mapping that associates `0100:DD/48` to `3.1.1.2` and `2.2.1.2`. Once the mapping has been received, it is installed in the mapping cache of the ITR and the packet is encapsulated and sent to the RLOC of the destination ETR. The destination ETR decapsulates the packet and forwards it to the destination endhost. Subsequent packets sent to this EID will be forwarded based on the cached mapping.

B. Mapping systems

The mapping system is a major component of the Locator/Identifier Separation Protocol as it provides the association between an identifier and its locators.

From an architectural standpoint there are two possible ways in which a mapping system could supply mapping information. It can either provide individual answers to specific requests (pull), or distribute (push) all the mappings onto listeners. In pull-based mapping systems, the ITR sends queries to the mapping system every time it needs to contact a remote EID and has no mapping for it. The mapping system then returns a mapping for a prefix that contains this EID. Pull-based mapping systems have thus similarities with today's DNS. Proposed pull-based mapping systems include LISP+ALT [8], LISP-CONS [10] and LISP-DHT [9]. In push-based mapping systems, the ITR receives and stores all the mappings for all EID prefixes even if it does not contact them. Push-based mapping systems have thus similarities with today's BGP. To the best of our knowledge, NERD [11] is the only proposed push-based mapping system.

1) LISP+ALT: The LISP Alternative Topology (LISP+ALT) [8] is a mapping system distributed over an overlay. All the participating nodes connect to their peers through static tunnels. BGP is the routing protocol chosen to maintain the routes on the overlay. Every ETR involved in the ALT topology advertises its EID prefixes making the EID routable on the overlay. Note though, that the mappings are not advertised by BGP. When an ITR needs a mapping, it sends a Map-Request to a nearby ALT router. It starts by constructing a packet with the EID, for which the mapping has to be retrieved, as the destination address, and the RLOC of the ITR as the source address. The ALT routers then forward the Map-Request on the overlay by inspecting their ALT routing tables. When the Map-Request reaches the ETR responsible for the mapping, a Map-Reply is generated and

directly sent to the ITR's RLOC, without using the ALT overlay.

2) LISP-DHT: LISP-DHT [9] is a mapping system based on a Distributed Hash Table (DHT). The LISP-DHT mapping system uses an overlay network derived from Chord [12]. Choosing this particular structured DHT over others (e.g., CAN, Pastry, Tapestry or Kademlia) was motivated by the algorithm used to map search keys to nodes containing the stored values. In a traditional Chord DHT, nodes choose their identifier randomly. In LISP-DHT, a node is associated to an EID prefix and its Chord identifier is chosen at bootstrap as the highest EID in that associated EID prefix. This enforces mapping locality that ensures that a mapping is always stored on a node chosen by the owner of the EID prefix, see [9] for details. When an ITR needs a mapping, it sends a Map-Request through the LISP-DHT overlay with its RLOC as source address. Each node routes the request according to its finger table (a table that associates a next hop to a portion of the space covered by the Chord ring). The Map-Reply is sent directly to the ITR via its RLOC.

3) LISP-CONS: The Content distribution Overlay Network Service for LISP, LISP-CONS [10], is a hierarchical content distribution system for EID-to-RLOC mappings. It is a generalization of LISP+ALT, which does not use the BGP routing protocol. On the other hand, it adds support for caching in intermediary nodes. In this paper we do not consider LISP-CONS as it does not seem to evolve anymore.

4) NERD: NERD is a flat centralized mapping database, using the push-model. Because any change requires a new version of the database to be downloaded by all ITRs, this approach is unlikely to scale to the needs of a future global LISP mapping system. The main advantage of NERD is the absence of cache misses that could degrade traffic performance.

III. LISP-TREE

This section presents the LISP-TREE mapping system. First, we provide arguments for using the protocols supporting today's Domain Name System (DNS), followed by a short overview of our proposal. Then, after the detailed description we also propose a deployment scheme for today's Internet.

A. Why a DNS-based mapping system?

On today's Internet, DNS is the system which is closest to a mapping system. More than twenty years of operational experience in implementing, testing, deploying, and operating such a system forms a solid foundation for an identifier-to-locator mapping service.

1) Scalability: Scalability is a key concern for any mapping system. The scalability of the current DNS system is due to several factors. First, caching is heavily used to reduce the number of queries that are sent by resolvers. Second, the domain names have a hierarchical structure, so the changes to the IP addresses and names of name servers responsible for domains are local to a few DNS servers and do not need to be propagated throughout the DNS. In contrast, the BGP-based LISP+ALT would require an intelligent organization of the topology, coupled with aggregation, to mitigate the risk of organic growth (i.e., due to non-technical reasons: social, economical) of the ALT routing table. Last, the combination of the previous two factors leads to a shortened path for a significant amount of queries, due to the fact that not only responses, but also intermediate nodes can be cached. Because queries can go directly from the resolver to servers lower on the hierarchy, not all cache misses have to traverse the tree.

2) Fault tolerance and troubleshooting: Network operators need to rapidly detect and fix the problems when they happen. DNS resolvers operate in iterative and recursive modes. In iterative mode, the resolver sends the queries to servers on different levels of the hierarchy, step by step, starting at the root. If one server does not reply, the resolver automatically tries another server [17]. In recursive mode, the query is forwarded from one server to another, closer to the authoritative, which sends the reply back on the same path.

In this case, the location of a failing intermediate server will not be known to the resolver, which will be unable to circumvent the failing node if no extra failure discovery protocol is deployed. DNS allows both forms of operation, but today's Internet mostly uses iterative DNS [18].

Moreover, fault tolerance is a key concern in a mapping system. Because a reply from the mapping system is necessary to allow an ITR to send packets towards a destination EID, any failure of the mapping system would block communications. In the current DNS, fault tolerance is provided mainly by replicating servers. All the important zones of the DNS hierarchy are served by two or more name servers with distinct IP addresses. Furthermore, some of these IP addresses are in fact anycast addresses that correspond to several physical servers. DNS resolvers take advantage of this redundancy by using load balancing when contacting name servers [17].

3) Security: None of the proposed LISP mapping systems have specified security features. While some existing solutions could be used to secure them (e.g., applying the Secure Inter-Domain Routing architecture to LISP+ALT [19]), they have not received enough wide-scale implementation, testing and operational experience. DNSSEC adds the required security mechanisms to allow resolvers to authenticate the replies received from DNS servers. It is readily available in most DNS implementations, with some top-level domains already deploying it. It does introduce additional complexity to the mapping system, but all add-on security mechanisms do so.

For completeness sake, security was included in this section, but a detailed security analysis is out of scope for this paper and is left for future work.

4) Impact of configuration errors: DNS configuration errors impact only the domains served by the misconfigured name server and do not propagate to the whole system. With LISP+ALT, the main source of misconfiguration would be the advertisement of incorrect EID prefixes via BGP on the overlay network. Experience with BGP shows that many misconfiguration errors could cause network operators to advertise an invalid BGP prefix with different impacts on the network [20]. All these misconfiguration errors could affect the ALT overlay as well.

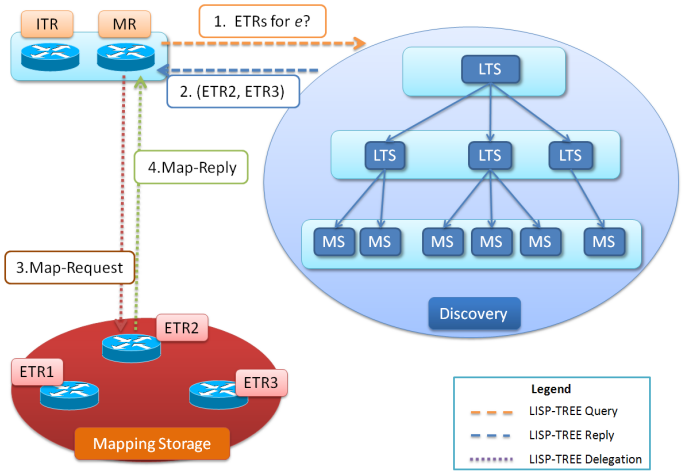


Fig. 2. Global overview of LISP-TREE. 1. The requester asks the discovery part to have the locator of some authoritative ETRs for an EID e. 2. The discovery part provides this list. 3. The Map-Resolver sends a Map-Request to one of these ETRs. 4. The ETR sends a Map-Reply back with a mapping for e.

B. LISP-TREE Overview

LISP-TREE is a hierarchical mapping system that has a clear separation of the storage of mappings and their discovery. The *mapping storage* is under the responsibility of the ETRs while the *discovery* mechanism is built on top of the DNS protocol. The role of the discovery mechanism is to provide a list of ETRs that respond authoritatively for the mappings associated to the queried EID.

Fig. 2 presents an overview of LISP-TREE. When a requester needs to obtain a mapping for an EID, it first sends a request to the discovery part that answers with a list containing the locators of the authoritative ETRs for the requested EID. The requester then sends a Map-Request to one of these ETRs and receives a Map-Reply containing a mapping for the identifier. The mappings are provided by the ETR to let them control their traffic by setting the priorities and weights.

C. LISP-TREE Model

The bindings between the EIDs and the locators are kept on the authoritative egress tunnel routers of customer domains. These ETRs manage and distribute these mappings with the aim of sinking all self owned EID-prefix mapping requests. All the mapping databases are combined to form the *Mapping Storage*. It will not be further detailed here as its functionality has already been defined in the LISP specification [7].

ETRs respond with Map-Replies only for the EID space they are authoritative for [7]. Because of this, it is the responsibility of the inquiring node, the Map-Request originator, to find the locators of the ETRs authoritative for the queried identifier. Such functionality is provided by the *discovery part* of LISP-TREE. It is implemented on top of the DNS protocol [21] and builds its tree by logically linking *LISP-TREE Servers (LTS)*. Although LISP-TREE is built over DNS, we suggest to deploy it independently to keep the separation between the DNS names and the identifier resolutions.

All the LTSes are *responsible* for an EID prefix and build a hierarchy determined by their intrinsic relationship. Therefore, an LTS responsible for EID prefix p_1 is ancestor of an LTS responsible for EID prefix p_2 if and only if $p_1 \preceq p_2$[1]. Moreover, any LTS of prefix p_1 is a parent of an LTS responsible for prefix p_2 if and only if there exists no LTS of prefix p_3 such that $p_1 \prec p_3 \prec p_2$. All these strict ordering relations are stored by parent LTS as a list of child servers. Exceptions are the lowest level servers, the leaves, which store a list of ETRs that are responsible for their associated prefix. Hence, the search for any EID e ends at a leaf of prefix p that respects $p \preceq e$.

The authoritative ETRs are registered to their responsible leaf LTSes by using the Map-Register procedure defined in [22]. In LISP terminology, the leaf LTSes are called *Map Servers (MS)* [22].

To make LISP-TREE transparent for the ITRs, *Map-Resolvers (MR)* [22] are added to the system. When a Map-Resolver receives a Map-Request from an ITR, it queries the LISP-TREE discovery part to obtain the list of the authoritative ETRs for the EID in the request. Once the MR has the corresponding list of authoritative ETRs, it sends a Map-Request to one of them and subsequently forwards the received Map-Reply to the requesting ITR. Thanks to this functionality, the ITRs do not need to know anything about LISP-TREE, they just need to send a Map-Request to an MR.

To avoid circular dependencies in the addresses, every node involved in the mapping system (i.e., MRs, LTSes, MSes and xTRs) is addressed with a locator.

D. LISP-TREE Modes

Like DNS, LISP-TREE can run in two distinct modes: (i) *recursive* and (ii) *iterative*. Fig. 3 illustrates the difference between the two and presents a requester who wants to obtain a mapping for the EID `192.0.2.20`. To simplify the figure, only the last EID octet's is shown.

1) Recursive Mode: In the recursive mode (Fig.3(a)), the MR requests a mapping for EID e from a root LTS[2]. The root LTS sends a request to one of its children responsible for e and so on until a MS is reached. The MS then generates a list of e's authoritative ETRs locators and this list is back-walked until the answer arrives the root LTS. At that stage, the root LTS sends the reply back to the MR.

2) Iterative Mode: In the iterative mode (Fig.3(b)), the MR requests a mapping for EID e from a root LTS. The LTS then sends back the locators of its children responsible for e to the MR. The MR then sends a request for the mapping to one of those children. The child does the same and answers with a list of locators of its children responsible for e, and so on until a MS is reached. The MS then generates a list of e's authoritative ETRs locators and sends it to the MR.

In both modes, when the MR receives the ETR locators list, it sends a Map-Request to one of them to get a mapping for e and eventually receives a Map-Reply. It is possible to optimize

[1] $p \preceq q$ if and only if prefix p is shorter (less specific) than q

[2] A root LTS is an LTS that serves the root of the tree

for latency in the last step, by allowing the MS to do DNS-to-LISP protocol translation and have it send the Map-Request to the ETR. We did not consider this scenario in order to keep the architectural separation and with that the good troubleshooting properties of LISP-TREE.

E. LISP-TREE Deployment Scenario

A key factor for the long-term success of a mapping system is its ability to scale. However, the effective deployment of a mapping system also largely depends on its ability to fit with a business model accepted by the community willing to use it.

We therefore propose a model that relies on the the *delegation* principle which is very common in the Internet. A globally approved entity (e.g., IANA, IETF...) defines the minimum set of rules and delegates the operations to independent service providers that can, at their turn, delegate to other providers and fix extended rules. People wishing to gain access to the system are free to contact any provider and sign agreements with them. The providers are differentiated by the extra services they propose.

The EID prefix delegation follows an approach similar to the current Internet prefix allocation. The IANA divides the EID space in large blocks and assigns them to five different *Regional EID Registries (RER)* that could be the current RIRs. The RERs are responsible for the allocation of prefixes to the *Local EID Registries (LERs)* and large networks. The LERs can then provide EID prefixes to some customers or LERs below, which, at their turn, can delegate the prefixes to other LERs or to customers. In this scenario, we consider that the EID space is independent from the currently assigned IP space. For example, it could be a subset of the IPv6 address space. This separation means that the EID space does not have to support all the legacy decomposition of prefixes observed on the Internet.

In this scenario, the tree has at least three levels. The first level (i.e., the root) is composed of LTSes maintained by the RERs. These servers replicate the same information: the list of all LERs responsible for all the large blocks of EIDs. The number of such blocks is limited (maximum 255 in an `/8` decomposition perspective) meaning that the total state to maintain at the root LTSes is limited even if every prefix is maintained by tens of different servers. Level 2 of the tree is composed of the LTSes maintained by the LERs and the lowest level encompasses the map servers responsible for customer ETRs. State at level 2 depends on the deaggregation of the large block of EIDs and implicitly on the number of subscribed customer map servers. To avoid having to support a large number of deaggregated prefixes, an LTS can partially deaggregate its EID block (e.g., divide it by two) and delegate such sub-blocks. In other words, the state to maintain at an LTS can be controlled by adapting the depth of the tree.

It is important to note that the EID prefixes are independent of any given ISP, and that additional levels in the hierarchy can be defined if needed. It is worth to note that this raises the problem of registrar lock-in: once an organization gets an allocation, it cannot move the allocated prefix to a different registrar. It is also important to remark that any LTS (including the MSes) can be replicated and maintained by independent server operators and implementations. Therefore, to limit the impact of the lock-in, we would recommend the registrars to ensure enough diversity in LTS operators for their blocks. No significant lock-in is observed in DNS today thanks to this way of deploying TLDs. The load can also be controlled among the different replicas by deploying them in anycast. Finally, the use of caches, like in the DNS, can reduce the number of LTSes involved in every query.

IV. Simulation Model

LISP and LISP+ALT are discussed in the LISP working group of the IETF and an implementation is being developed for Cisco platforms. A world-wide testbed of more than 50 tunnel routers (at the time of this writing) relying on the LISP+ALT mapping system is also deployed[3]. In addition, an open source implementation of the LISP tunnel router functionality is developed by the OpenLISP project [23] for the FreeBSD operating system. However, while these implementations help to validate and gain operational experience with the proposed protocols, they do not allow to estimate the behavior of LISP at a very large scale. We developed *CoreSim* [16] for this purpose, an open source Internet-scale LISP deployment simulator[4]. CoreSim combines a packet trace and Internet latency data to simulate the behavior of an ITR and the mapping system.

CoreSim works on top of a topology built from measurements performed by the iPlane infrastructure[5], which provides Internet topology information and the latency between arbitrary IP addresses. The simulator reports mapping lookup latency, the load imposed on each node of the mapping system and cache performance statistics.

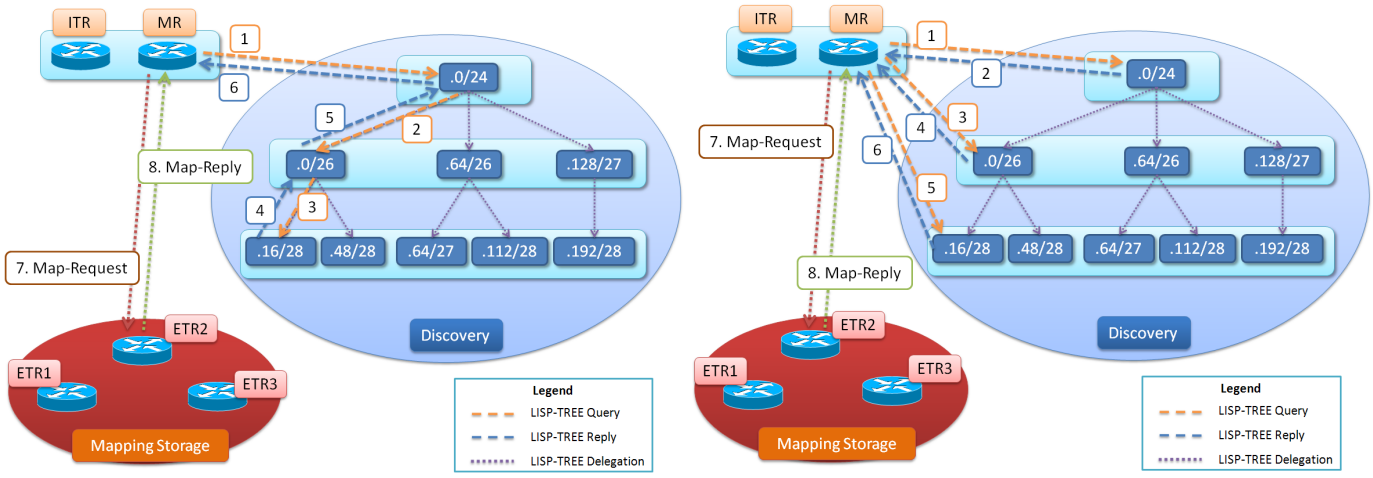
CoreSim is composed of two main building blocks (see Fig. 4): the first one, `ITR`, simulates an ITR with its associated operations (sending Map-Requests, caching Map-Replies and forwarding packets), while the second, `MS`, simulates the mapping system (path taken and mapping latency).

A packet trace file is used as input data for the ITR block. The position of the simulated ITR in the topology built by CoreSim (see next section) is determined by the real world location of link where the trace was captured.

A. Topology Model

The first element to model the behavior of mapping systems in a large scale network is the network itself. For CoreSim, we chose to use the current Internet as the reference topology. The topology used in the simulator is composed of 14,340 points of presence (PoPs), one per autonomous system (AS), about which iPlane [24] knows inter-domain peering information. For the simulations we assume that each AS deploys one and only one LISP tunnel router (xTR) and that its xTR is located on its most connected PoP.

[3]See http://www.lisp4.net/ for more details.
[4]Available from http://lisp.cba.upc.edu/
[5]See http://iplane.cs.washington.edu/



(a) Recursive mode: queries in the discovery parts are progressively transmitted to a MS. The MS answer then back-walk the tree up to the root that sends the answer to the MR.

(b) Iterative mode: queries are moving back and forth from the MR and the LTSes, starting at a root LTS until a MR is reached. The MS then provides the answer to the MR.

Fig. 3. LISP-TREE works with different mode: (a) recursive and (b) iterative.

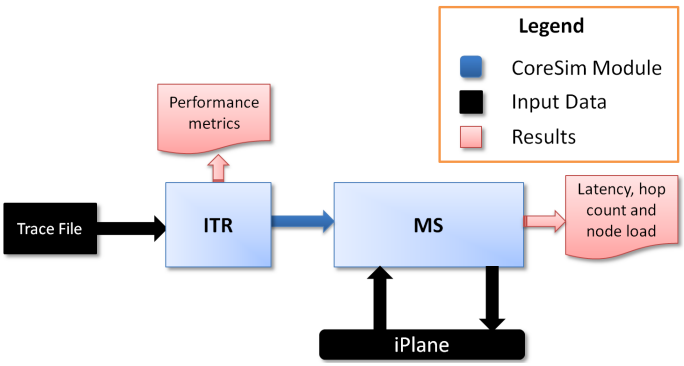


Fig. 4. CoreSim architecture

Once the xTRs are selected, they are assigned several EID prefixes. Those prefixes are the IP prefixes currently advertised in the default-free zone (DFZ) by this AS. Since the causes of prefixes deaggregation in BGP is not always clear, we removed from the iPlane dataset the more specific prefixes. These prefixes are mostly advertised for traffic engineering purposes [2] and would not be advertised with LISP as it natively supports traffic engineering in the mappings. A total number of $112,233$ prefixes are assigned based on originating AS to their respective xTR[6].

CoreSim relies on the iPlane Internet latency lookup service that returns the measured latency between arbitrary IP addresses to simulate the time required to obtain a mapping in the simulated mapping system. Unfortunately, iPlane does not provide all possible delay pairs. Because of this, for 35% of the lookups, we use a latency estimator that correlates the geographical distance between IP addresses as reported by the MaxMind database[7] with the latencies based on an iPlane training dataset (see details in [25]). This approach was found only slightly less accurate than other more complex algorithms in [26] using measurement data from over 3.5 million globally well distributed nodes.

An important assumption made by CoreSim is that there is no churn in the topology during a simulation run, meaning that the delay between two nodes is constant, and that nodes never fail.

B. ITR Model

CoreSim studies the behavior of only one ITR at a time, therefore, out of all xTRs, one is selected as the ITR under study (see Fig. 4). The PoP for the selected ITR is determined by manual configuration, based on the point of capture of the traffic trace that is fed to the simulator. The ITR caches mappings (i.e., resolved Map-Requests) and evicts entries after 3 minutes of inactivity.

C. Mapping System Models

In CoreSim, a LISP Mapping System is modeled as an IP overlay. The overlay is mainly composed of nodes of the topology module, but also includes some mapping system specific ones. As expected the organization of the overlay depends on the considered mapping system.

The simulator routes Map-Requests from the ITR to the node authoritative for the mapping (ETR). The path of the query and the latency of each hop are recorded in order to infer statistics and metrics of interest.

1) LISP+ALT model: The LISP+ALT draft [8] envisages a hierarchical topology built with GRE tunnels but does not recommend any organization for the overlay. Therefore, among all the possible organizations, a hierarchical overlay structure with strong EID prefix aggregation for advertisements has been chosen.

Based on discussions on the LISP mailing list, a three-level hierarchy was decided (see Figure 5). In this hierarchy, the bottom leaf nodes are ALT routers belonging to a certain domain. The upper two levels are dedicated ALT routers, which may

[6]Data is from March 2009

[7]Available from http://www.maxmind.com/.

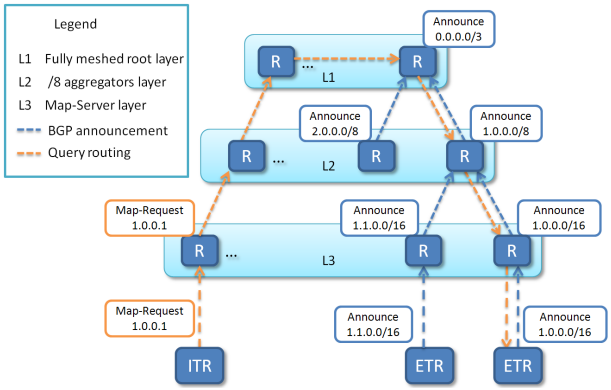


Fig. 5. LISP+ALT architecture

be offered as a commercial service by providers or registries. Each of these nodes is responsible for certain aggregated prefixes, and connects to all lower level nodes which advertise sub-prefixes included in those prefixes. The hierarchy consists of 16 root (first) level ALT routers, responsible for the eight /3 prefixes, and 256 second level ALT routers, each responsible for a /8 prefix. For each /3 prefix two ALT routers at different locations are used and each lower level ALT router connects to the one with the lowest latency. All these 16 routers are connected to each other with a fully meshed topology. Please note that LISP+ALT can support other topologies and for instance include intra-level links.

Map-Requests are routed via the shortest path through the hierarchy starting from the bottom layer where the ITR is connected to the ALT topology.

2) LISP-DHT model: The simulator considers the 112,233 filtered prefixes from iPlane and builds a static Chord ring of the same size, using a trace-based approach to compute the finger table on each node. The last EID of each prefix is used as the ChordID of the LISP-DHT nodes that is responsible for this EID prefix. EID lookups proceed according to the standard Chord protocol, which we implemented in CoreSim.

3) LISP-TREE model: CoreSim considers only one ITR at a time, and because the MR always selects the closest root LTS, the one selected by the MR is always the same. Therefore, the simulator considers only one root LTS for the tree. This server connects to the 256 level 2 LTSes that are each responsible for one /8 EID prefix. In turn, these servers know about all the third level LTSes that are responsible for the prefixes included in their /8 prefix. These third level servers are the map servers the ETRs subscribe to. The simulator assumes that an ETR registers to a single MS, and that MS is located in the same PoP as the ETR. Since the simulator assigns them to the same PoP, the resulting latency between them is 0. Further, because the latency introduced by DNSSEC is two orders of magnitude lower than typical mapping lookup latencies [27], it is considered negligible and is not accounted for in the simulator. This deployment scenario is an instantiation of the one presented in Sec. III-E which is congruent with the current Internet.

Additional implementation details on the simulator can be found in [28].

V. Mapping System Comparison

Section III described in detail the advantages of our proposed mapping system from an architectural point of view. This section complements that with a qualitative analysis, comparing several performance metrics of three mapping systems: lookup latency, hop count, node load and the amount of state stored in mapping system nodes. Low lookup latency improves user experience for new flows, while node load and state affect the scalability of the system. We begin by describing the packet traces that support our evaluation.

A. Experimental Datasets

In order to evaluate the performance of the mapping systems presented above we used traffic traces collected at the border routers of two university campuses, because border routers are the most likely place to deploy a LISP tunnel router. The first trace was captured at Université catholique de Louvain (UCL) in NetFlow format, and the second is a packet trace from Universitat Politècnica de Catalunya (UPC).

1) UCL: The UCL campus is connected to the Internet with a 1 Gbps link via the Belgian national research network (Belnet). This trace consists of a one day full NetFlow trace collected on March 23, 2009. For this paper, only the outgoing traffic is considered, representing 752 GB of traffic and 1,200 million packets for an average bandwidth of 69 Mbps. A total number of 4.3 million different IP addresses in 58,204 different BGP prefixes have been contacted by 8,769 different UCL hosts during the 24 hours of the trace. The UCL campus is accessible to more than 26,000 users.

NetFlow generates transport layer flow traces, where each entry is defined as a five-tuple consisting of the source and destination addresses and ports, and the transport layer protocol. The simulator however requires packet traces. This is why the NetFlow trace collected at UCL has been converted into a packet trace: for each flow, we generated the number of packets specified in the NetFlow record, distributed evenly across the flow duration and the size of the flow. Throughout the rest of the paper, the term *UCL trace* corresponds to the packet trace obtained from the NetFlow trace collected at UCL.

2) UPC: The second unidirectional trace we used was captured at the 2 Gbps link connecting several campus networks of UPC to the Catalan Research Network (CESCA) with the help of the CoMo infrastructure [29]. It consists of the egress traffic on May 26, 2009 between 08:00-11:49 local time, and contains about 1,200 million packets accounting for 463 GB of traffic with an average bandwidth of 289 Mbps. 4.3 million distinct destination IP addresses from 56,387 BGP prefixes were observed in the trace. UPC Campus has more than 36,000 users.

B. Cache Miss Rate

The average packet rates at the UCL and UPC border routers are 13 Kpkts/s and 87 Kpkts/s, respectively. However, due to the nature of the traffic, a mapping is already cached by the ITR for most of the packets, with only 0.31% and

0.1% of cache misses for the mentioned vantage points. A cache miss occurs when no mapping is known for an EID to encapsulate. On cache miss, a Map-Request is sent to obtain a mapping, resulting in 2,350,000 Map-Requests sent for UCL and 560,000 for UPC during the 24h and 4h periods of the traces, which are shorter than the default TTL value (1 day) for mappings. As a result, all cache evictions were due to lack of activity to a particular prefix for 3 minutes, rather than expired TTL. These values for Map-Requests have to be compared with the total of 1,200 million packets observed for both traces. The difference between UCL and UPC can be explained by the higher average packet rate of the UPC trace, which keeps the cache more active, resulting in less timed out entries.

During our simulations the maximum number of mapping cache entries reached was 22,993 and 15,011 for the two traces. This is an order of magnitude less than routes in the DFZ. For a detailed mapping cache study, please refer to [14].

It is important to note that the results are obtained for a cache initially empty. Therefore, the miss rate is very important at the beginning of the experiment and thus influences the number of requests. The number of Map-Requests would have been dramatically smaller if we had considered the system at the steady state.

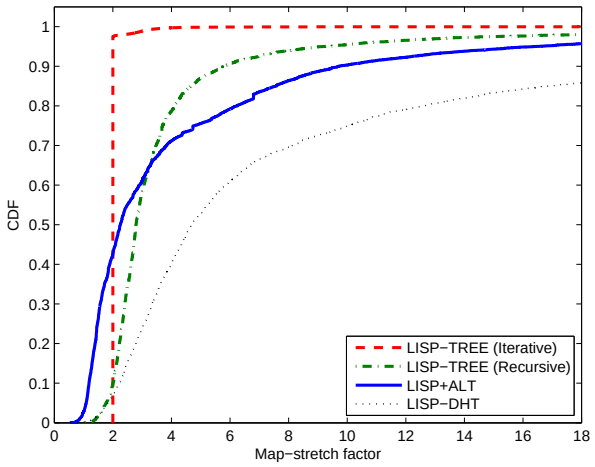
C. Mapping Lookup Latency

The mapping lookup latency is particularly important in mapping systems as it represents the time required to receive a Map-Reply after sending a Map-Request. When an ITR waits for a mapping for an EID, no packet can be sent to this EID. If this delay is too long, the traffic can be severely affected.

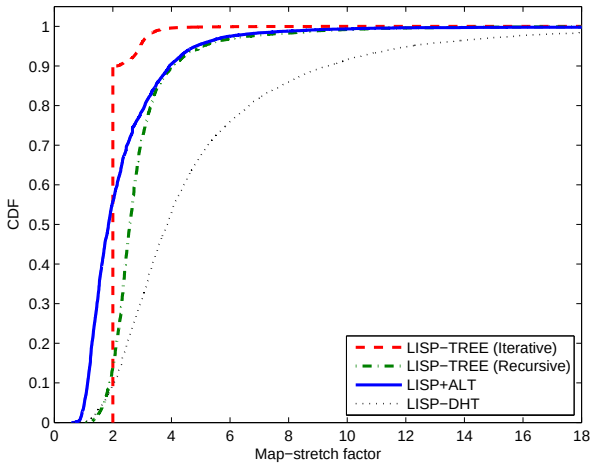
For instance, one of the Internet's most popular resources, the World Wide Web is continuously evolving, delivering web pages that are increasingly interactive. Content on these pages is often from different servers, or even different providers, and presents a multi-level dependency graph [30]. This is already a challenging environment for some applications, and the introduction of a new level of indirection has the potential to introduce additional latencies. However the ATLAS study [31] shows that content is more and more comming from datacenters and CDNs.

To compare the considered mapping systems, we define the *map-stretch factor* of a lookup as the ratio between the total time required for performing it and the round-trip delay between the ITR and ETR. A low map-stretch indicates that the mapping system introduces a low delay overhead.

Fig. 6 presents the cumulative distribution function of the map-stretch factor obtained for both the UCL and UPC traces. In most of the cases (over 90%) the iterative LISP-TREE presents a stretch factor of 2. After the initial cache warm up, there is no need to contact the root and level 2 servers, their responses are already cached. Still, the discovery phase cannot be completely avoided and the level 3 MS have to be contacted for the RLOCs of the ETRs. These are not cached in the MR, because it would have a negative impact on the mapping dynamics, limiting the possibilities for supporting fast end-host mobility.



(a) UCL



(b) UPC

Fig. 6. CDF of map-stretch ratio. Use of caching makes LISP-TREE have a constant value for the majority of the lookups.

Recursive LISP-TREE is slower than iterative in over 90% of the cases. The caching limitation mentioned in the previous paragraph has particularly negative consequences for this operating mode: since the only response arriving to the MR is the list of RLOCs for the authoritative ETRs, no caching at all can be done. The 10% of the cases when recursive is better than iterative can be attributed to the path via the tree being faster than from the MR to the MS, resulting in map-stretch ratio values below 2.

Concerning LISP+ALT, its latency performance is similar to LISP-TREE iterative. This is because in LISP+ALT, the queries are routed through to the overlay topology, which is composed by core routers. According to our assumptions (see Section IV), these nodes are located in well connected PoPs. However, in iterative LISP-TREE the query flows in almost all the cases between the ITR and ETR, which may not be so well-connected. When comparing LISP+ALT and LISP-TREE recursive, we can see that LISP+ALT performs slightly better. In the recursive mode of LISP-TREE queries are also

forwarded through a topology of well-connected nodes, but as we will see, they have to follow a much longer path.

As expected, LISP-DHT has the highest latency because of the longer path taken by the queries routed through the P2P topology. This is a well known problem in P2P networks, and research to tackle this issue is ongoing [32]. However, it is worth to note that LISP-DHT uses a particular Chord ring, where the peers do not choose their identifiers randomly, but as the highest EID in the EID prefix. This enforces mapping locality that ensures that a mapping is always stored on a node chosen by the owner of the EID prefix. As a consequence, it may be difficult for LISP-DHT to benefit from existing optimization techniques proposed in the literature.

Average lookup latencies were close to half a second for all mapping systems except LISP-DHT, which had values slightly larger than one second.

Fig. 7 helps to better understand the mapping latency difference between the mapping systems. It presents a CDF of the number of hops passed by over which a request and reply.

The iterative version of LISP-TREE has the lowest hopcount values, which can be explained, just like for the latency, by caching in the Map-Resolver. Recursive LISP-TREE not helped by caching, and queries have to traverse the full path in each case, for a maximum of 8 hops (Fig. 3(a)).

The topology chosen for LISP+ALT in the simulator (Fig. 5) limits the maximum number of hops to 6, but in 95% of the cases, this maximum number of hops is observed. In order to have a shorter path, the destination EID would have to be in one of the `/8` prefixes that doesn't have a more specific part announced separately. As we will see in the next section, this also increases the load on the nodes composing the LISP+ALT mapping system. In fact, Fig. 5 shows that all queries are forwarded through the root layer. This may result in scalability problems.

LISP-DHT has a much higher hop count with a maximum of 17 hops. This not only increases the lookup latency, it means that more nodes are involved in the resolution of a mapping than in the case of the other mapping systems, increasing the overall load on the system and the probability of failure.

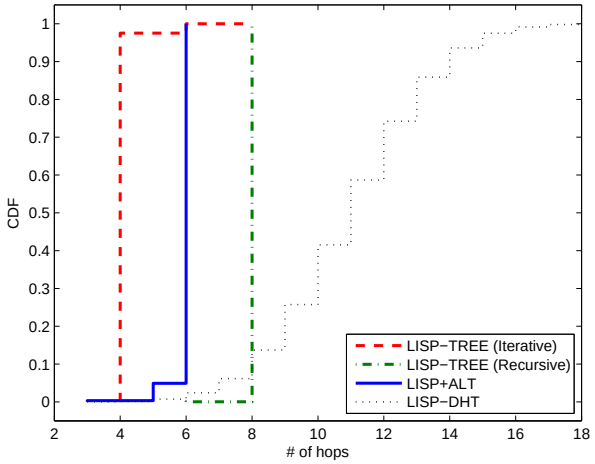
Summarizing, these results reveal significant differences among the mapping systems under consideration. LISP-TREE and LISP+ALT both use a hierarchical model, where nodes on the query path tend to be congruent with the topology. In contrast, the Chord overlay used to route queries in LISP-DHT does not follow the underlying physical topology. Further, iterative LISP-TREE allows caching and this reduces its mapping latency.

The latency introduced by the mapping system in case of cache misses will likely have a negative impact on the higher layer protocols, in particular on congestion control in the transport or application layer. These issues deserve a dedicated study, and are left for future work.

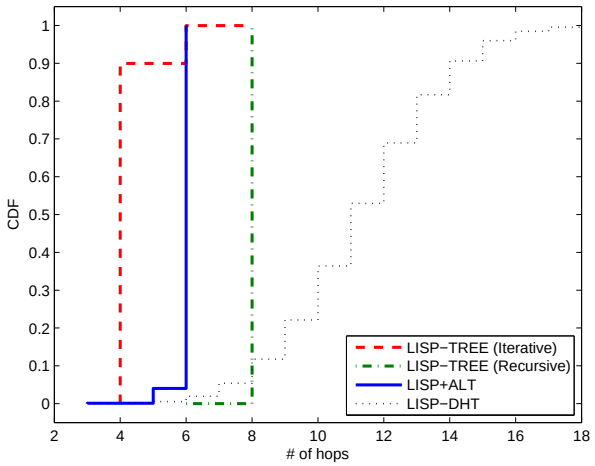
D. Node Load

We define the *node load* as the total number of Map-Request messages processed by nodes of the mapping system during the full run of the trace. For a more thorough analysis, we differentiate between the load caused by messages forwarded by the node (*transit load*) and the load due to the messages for which the node is the final destination (*mapping load*). Mapping load mainly depends on the observed traffic (distribution of destination EIDs) and is mostly the same for all studied mapping systems. On the other hand, transit load is mapping system specific and depends on how a request is sent to the holder of the mapping. Table I summarizes the load statistics of the UCL trace.

The root LTS is only contacted 158 times in the case of



(a) UCL



(b) UPC

Fig. 7. CDF of hop count. For hierarchical mapping systems it is almost constant, for LISP-DHT we have a wide array of different values.

TABLE I
NODE LOAD IN LEVELS 1 AND 2

	Level 1		Level 2	
Mapping System	**Avg.**	**Max.**	**Avg.**	**Max.**
LISP-TREE (Itr.)	158	158	372	2,354
LISP-TREE (Rec.)	2,351,815	2,351,815	14,941	70,520
LISP+ALT	655,600	2,348,695	29,776	2,356,404
LISP-DHT	147,860	1,257,642	258	2,365,797

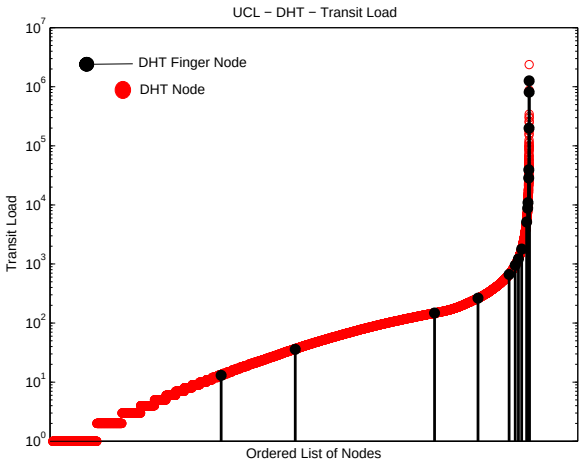


Fig. 8. LISP-DHT load distribution. Fingers of the ITR, represented with the vertical lines, cluster on the right, having a considerable load.

iterative LISP-TREE, that is, the number of times necessary to look up the locators of the level 2 nodes responsible for the contacted /8 prefixes. Since these locators are cached by the Map-Resolver, they are only requested once. The load on the level 2 servers is also very low, compared to the other mapping systems, where there is no caching on intermediate nodes. In recursive LISP-TREE all Map-Requests have to pass through the root node, and then get distributed to the lower level nodes, according to the destination prefix. In LISP+ALT, level 1 consists of 7 logical nodes for the seven /3 covering the routable unicast prefixes. The nodes responsible for IP space with high prefix density sustain a considerably higher load.

For LISP-DHT, level 1 in the table refers to the transit load of the ITR's fingers, while level 2 to the transit load of all other nodes. Figure 8 shows the transit load in LISP-DHT. Vertical lines represent fingers of the ITR at UCL, which was originating the Map-Requests. From the $112,233$ nodes participating in the DHT, only 74% have routed or received Map-Request messages and are depicted in the figure. We can observe a sharp surge after a load value of 1000, that accounts for about 1.8% of the total number of nodes on the DHT. As expected we find many of the ITR's fingers among these hotspots. Of the $2,365,797$ Map-Requests initiated, more than half pass via the last finger and one third via the second last finger. Among the top ten most loaded nodes 7 are not fingers.

Upon further inspection it was discovered that the node with the highest load was the last finger of the ITR. Due to the way Chord lookups work, this node is responsible for half of the key space on the Chord ring, which explains the high load. Further investigation revealed that there is one node which is last finger for 5.6% of the LISP-DHT participants: the one responsible for the prefix 4.0.0.0/8. This is the first prefix from the IPv4 space present in the iPlane dataset. Since Chord is organized as a ring (key space wraps around), this node becomes responsible for the EID space of classes D and E as well. Because this EID space is not represented in the overlay, the result is a disproportional load. The UPC trace produced similar load distribution results.

LISP-DHT's transit traffic distribution characteristics may be desirable for peer-to-peer networks, but are a major disadvantage for a mapping system. Since the transit route is defined only by the Chord routing algorithm, two issues arise: lack of path control may lead to choke points at nodes belonging to small sites, and exposure of mapping traffic to unknown third parties (potentially leading eavesdropping and denial-of-service attacks). This makes LISP-DHT a poor choice as a mapping system.

Due to these architectural differences, the mapping systems under study exhibit different transit load characteristics. Indeed, in the case of LISP-DHT all participating nodes are both transit and destination nodes, while the hierarchical mapping systems have dedicated transit nodes on levels 1 and 2. The iterative version of LISP-TREE has a significantly lower load in those dedicated transit nodes, because of the caching done in the Map-Resolver.

LISP+ALT needs to route all packets through the root nodes, producing a potential hot spot. LISP-TREE on the other hand avoids using the root nodes most of the time, because it is able to cache intermediate nodes. Apart from the obvious scalability advantages, this improves reliability as well, since a total failure of the root infrastructure would still allow partial functioning of LISP-TREE, while no resolutions would be possible in LISP+ALT.

E. Operational considerations

The mapping requests are transmitted through the mapping system towards the queried ETR. To achieve this, every node involved in the topology has to store some *state* about its neighbors. The amount of state needed on each node is directly related to the mapping system technology. For example, in an horizontally organized system such as LISP-DHT, all nodes have the same amount of state (32 entries). On the contrary, the amount of state needed on a node in a hierarchical mapping system depends on its position in the hierarchy.

LISP-TREE and LISP+ALT are both hierarchical topologies, and in this paper have been deployed according to the same EID prefix distribution, and thus have the same amount of state. The root nodes refer to all the disjoint /8 prefixes, which amounts to a maximum of 256 entries when all are allocated. The number of nodes that are referred to by level 2 nodes depends on how the prefixes are allocated. Fig. 9 shows the distribution of the state kept at each level 2 node (both LISP-TREE and LISP+ALT). Finally, the leaves store a small amount of entries, equal to the number of announced prefixes.

Fig. 9 shows that 15% of the nodes have more than 1000 children and the most connected node has 6072 children. For LISP-TREE this is not an issue, as the nodes only need to keep a database that relates a prefix with the list of locators. It is well known that currently deployed DNS servers scale to much more than thousands of records [13]. However, in the case of LISP+ALT, a BGP session has to be maintained for each child, as well as a tunnel between the two nodes. The costs in terms of configuration, memory, and processing are much

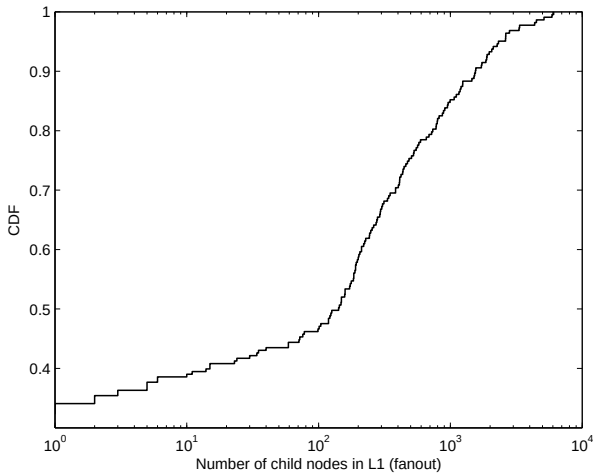


Fig. 9. The cumulative distribution of the amount of state in the level 2 LISP-TREE and LISP+ALT nodes

higher than for LISP-TREE. A way to alleviate this would be to increase the depth of the tree and thus reduce the number of children of any one node. Unfortunately, this solution stretches the path length within the tree and is likely to increase the mapping lookup latency. Another solution could be to follow an organic deployment, but in that case, the mapping system would eventually have scalability issues because of the lack of aggregability.

VI. Related Work

On today's Internet, the DNS [21] is the system which is closest to a mapping system. Several researchers have proposed to extend it to store identity-to-location bindings, and provide mobility support in the Internet (e.g., [33], [34], [35]). In this paper we extend the DNS to support the LISP mapping system. We also propose to follow the same address allocation policy as in today's Internet. Further, by means of measurement-driven simulations, we show its feasibily along with performance and operational benefits.

Concerning the mapping systems, several architectures have been proposed for LISP [8], [10], [11], [9], but to our knowledge none of these systems have been evaluated in detail. Iannone and Bonaventure have analysed in [14] the mapping cache used by ITRs. Using trace-driven simulation, they showed that the number of entries in this cache grows with the TTL of the cache and that the mapping system should provide mappings for EID prefixes and not individual EIDs. CoreSim also models the mapping cache on the ITR, but furthermore it models the entire mapping system and provides detailed information about its performance in terms of delay and load on mapping nodes. Other researchers [36], [37] have analysed the utilisation of caches or similar techniques to reduce the size of FIB tables on routers.

Luo et al. proposed in [38] another LISP mapping system that relies on the CAN DHT. This mapping system provides mappings for individual EIDs instead of EID prefixes as [8], [10], [11], [9]. Using such flat EIDs is unlikely to scale. A trace-driven evaluation of this mapping system is provided in [38]. The evaluation mainly focuses on the size of the mapping cache and the number of hops through the CAN DHT. CoreSim models delays and is not dedicated to a single mapping system.

VII. Conclusion

This paper proposes a new mapping system: LISP-TREE, and a measurement-driven simulation of the main LISP mapping systems. LISP-TREE is based on the Domain Name System and is built on a hierarchical topology. From an operational point of view the main advantage of LISP-TREE over LISP+ALT and LISP-DHT is that it enables clients to cache information about intermediate nodes in the resolution hierarchy, and direct communication with them. This avoids resolution request traffic concentration at the root, and as a result iterative LISP-TREE has much better scaling properties. Further, LISP+ALT's scalability also depends on enforcing an intelligent organization that increases aggregation. Unfortunately, the current BGP system shows that there is a risk of an organic growth of LISP+ALT, one which does not achieve aggregation. Neither of the LISP-TREE variants displays this weakness, since their organization is inherently hierarchical (and thus inherently aggregable).

Secondly, the hierarchical organization of LISP-TREE also reduces the possibility for a configuration error which could interfere with the operation of the network (unlike the situation with the current BGP). Existing DNS security extensions can also help produce a high degree of robustness, both against misconfiguration, and deliberate attack. The direct communication with intermediate nodes in iterative LISP-TREE also helps to quickly locate problems when they occur, resulting in better operational characteristics.

And thirdly, in LISP+ALT and LISP-DHT, since mapping requests must be transmitted through the overlay network, a significant share of requests can see substantially increased latencies. Our simulation results clearly show, and quantify, this effect. Also, our simulations show that the nodes composing the LISP+ALT and LISP-TREE networks can have thousands of neighbors. This is not an issue for LISP-TREE, but may be problematic for LISP+ALT nodes, since handling that number of simultaneous BGP sessions could be difficult.

References

[1] G. Huston, "Growth of the BGP table - 1994 to present."
[2] 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. [Online]. Available: http://www.ietf.org/rfc/rfc4984.txt
[3] C. Vogt, "Six/One router: A scalable and backwards compatible solution for provider-independent addressing," in *MobiArch*, Aug. 2008.
[4] M. Menth, M. Hartmann, and D. Klein, "Global locator, local locator, and identifier split (GLI-split)," University of Würzburg, Tech. Rep. 470, Apr. 2010.
[5] D. J. et al, "APT: A Practical Transit Mapping Service," draft-jen-apt-01.txt, Internet Engineering Task Force, Nov. 2007, work in progress.
[6] A. Feldmann, L. Cittadini, W. Muhlbauer, R. Bush, and O. Maennel, "Hair: Hierarchical architecture for internet routing," in *Re-Architectuing the Internet (ReArch '09)*, L. Eggert and T. Wolf, Eds., 2009.
[7] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID Separation Protocol (LISP)," draft-ietf-lisp-05, Internet Engineering Task Force, Sep. 2009, work in progress.

[8] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "LISP Alternative Topology (LISP+ALT)," draft-ietf-lisp-alt-01, Internet Engineering Task Force, May 2009, work in progress.

[9] L. Mathy and L. Iannone, "LISP-DHT: Towards a DHT to map identifiers onto locators," in *ReArch*, Dec. 2008.

[10] S. Brim, N. Chiappa, D. Farinacci, V. Fuller, D. Lewis, and D. Meyer, "LISP-CONS: A Content distribution Overlay Network Service for LISP," draft-meyer-lisp-cons-04, Internet Engineering Task Force, Apr. 2008, work in progress.

[11] E. Lear, "NERD: A Not-so-novel EID to RLOC Database," draft-lear-lisp-nerd-04, Internet Engineering Task Force, Apr. 2008, work in progress.

[12] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[13] VB.com, "Domains Counter - Domain Timeline since 1985 by VB.com," 2009.

[14] L. Iannone and O. Bonaventure, "On the cost of caching Locator/ID mappings," in *Proceedings of the 3rd International Conference on emerging Networking EXperiments and Technologies (CoNEXT'07)*. ACM, Dec. 2007, pp. 1–12.

[15] H. V. Madhyastha, T. Anderson, A. Krishnamurthy, N. Spring, and A. Venkataramani, "A structural approach to latency prediction," in *IMC*, Oct. 2006.

[16] F. Coras, L. Jakab, A. Cabellos, J. Domingo-Pascual, and V. Dobrota, "Coresim: A simulator for evaluationg locator/id separation protocol mapping systems," in *Trilogy Future Internet Summer School poster session*, 2009.

[17] P. B. Danzig, K. Obraczka, and A. Kumar, "An analysis of wide-area name server traffic: a study of the internet Domain Name System," *SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 4, pp. 281–292, 1992.

[18] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 589–603, Oct. 2002.

[19] K. Butler, T. Farley, P. Mcdaniel, and J. Rexford, "A Survey of BGP Security Issues and Solutions," *IEEE/ACM Transactions on Networking*, vol. 98, no. 1, pp. 100–122, Jan. 2010.

[20] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding bgp misconfiguration," in *SIGCOMM*, Aug. 2002, pp. 3–16.

[21] P. V. Mockapetris and K. J. Dunlap, "Development of the domain name system," *SIGCOMM Comput. Commun. Rev.*, vol. 25, no. 1, pp. 112–122, 1995.

[22] D. Farinacci and V. Fuller, "LISP Map Server," draft-fuller-lisp-ms-00, Internet Engineering Task Force, Mar. 2009, work in progress.

[23] L. Iannone, D. Saucez, and O. Bonaventure, "OpenLISP Implementation Report," draft-iannone-openlisp-implementation-01, Internet Engineering Task Force, Feb. 2008, work in progress.

[24] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *USENIX OSDI*, Nov. 2006.

[25] A. Cabellos-Aparicio, D. Saucez, O. Bonaventure, and J. Domingo-Pascual, "Validation of a LISP simulator," UPC, Tech. Rep. UPC-DAC-RR-CBA-2009-8, 2009. [Online]. Available: http://gsi.ac.upc.edu/reports/2009/46/TR.pdf

[26] S. Agarwal and J. R. Lorch, "Matchmaking for online games and other latency-sensitive P2P systems," in *SIGCOMM*, Aug. 2009, pp. 315–326.

[27] D. Wessels and G. Sisson, "Root zone augmentation and impact analysis," DNS-OARC, Tech. Rep., 2009.

[28] F. Coras, "CoreSim: A simulator for evaluating LISP mapping systems," Master's thesis, Technical University of Cluj-Napoca, Jun. 2009.

[29] P. Barlet-Ros, G. Iannaccone, J. Sanjuàs-Cuxart, D. Amores-López, and J. Solé-Pareta, "Load shedding in network monitoring applications," in *USENIX Annual Technical Conference*, 2007.

[30] D. Wischik, "Short messages," in *Royal Society workshop on networks: modelling and control*, Sep. 2007.

[31] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, and M. Karir, "Atlas internet observatory 2009 annual report," Arbor Networks, the University of Michigan and Merit Network, Tech. Rep., 2009.

[32] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: Provider portal for applications," in *SIGCOMM*, Aug. 2008.

[33] A. C. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, Aug. 2000.

[34] R. Atkinson, S. Bhatti, and S. Hailes, "Mobility through naming: Impact on DNS," in *MobiArch*, Aug. 2008.

[35] A. R. S. A. M. F. S. Ahmed, "Performance of DNS as location manager," in *IEEE International Conference on Electro Information Technology*. IEEE, 2005.

[36] C. Kim, M. Caesar, A. Gerber, and J. Rexford, "Revisiting route caching: The world should be flat," in *PAM*, Apr. 2009.

[37] H. Ballani, P. Francis, T. Cao, and J. Wang, "Making routers last longer with viaggre," in *USENIX NSDI*, Apr. 2009.

[38] H. Luo, Y. Qin, and H. Zhang, "A DHT-based identifier-to-locator mapping approach for a scalable Internet," *IEEE Transactions on Parallel and Distributed Systems*, Feb. 2009.