A Deep Dive into the LISP Cache
and what ISPs should know about it

Juhoon Kim[⋆], Luigi Iannone, and Anja Feldmann

Deutsche Telekom Laboratories – Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany

Abstract. Due to scalability issues that the current Internet is facing, the research community has re-discovered the Locator/ID Split paradigm. As the name suggests, this paradigm is based on the idea of separating the identity from the location of end-systems, in order to increase the scalability of the Internet architecture. One of the most successful proposals, currently under discussion at the IETF, is LISP (Locator/ID Separation Protocol). A critical component of LISP, from a performance and resources consumption perspective, as well as from a security point of view, is the LISP Cache. The LISP Cache is meant to temporarily store mappings, i.e., the bindings between identifiers and locations, in order to provide routers with the knowledge of *where* to forward packets. This paper presents a thorough analysis of such a component, based on real packet-level traces. Furthermore, the implications of policies to increase the level of security of LISP are also analyzed. Our results prove that even a timeout as short as 60 seconds provides high hit ratio and that the impact of using security policies is small.

Keywords: Locator/ID Separation; Next Generation Internet; Addressing and Routing Architectures; Measurements;

1 Introduction

In the last years, there has been an increasing interest in the Locator/ID Split paradigm, which is recognized to be a strong candidate to become a fundamental building block of the Next Generation Internet. Differently from the current architecture, where one single namespace, namely the IP address space, is used for both indentifying and locating end-systems, in the Locator/ID Split paradigm two different namespaces are used: the *ID space* and the *Locator space*, to respectively identify and locate end-systems. Such a separation aims at solving the scalability issues that the current Internet is facing [19], mainly concerning the continuously increasing BGP routing table [1], but also concerning addressing, mobility, multi-homing [22], and inter-domain traffic engineering [25].

The main effort to tackle these issues has started three years ago, when the Routing Research Group (RRG) has been rechartered to explore the possibility

[⋆] Corresponding Author: Juhoon Kim (`jkim@net.t-labs.tu-berlin.de`).

to enhance the Internet architecture by introducing some form of Locator/ID Split. Among the numerous proposals introduced since then [16], the most successful so far are ILNP (Identifier/Locator Network Protocol [5]), which has been adopted by the RRG, and LISP (Locator/ID Separation Protocol [7]), which has been chartered as Working Group in the IETF and has been already deployed in an international test-bed [2]. On the one hand, all the proposals improve the Internet architecture, in a way or another, by some form of Locator/ID Split. On the other hand, they introduce the necessity to distribute and store bindings between IDs and Locators (i.e., mappings), which is the key critical component that can make the difference. Hence, the importance of evaluating what is the *cost* (i.e., memory consumption, overhead, ...) of storing and distributing these mappings. In addition, the way they are managed does also have an impact on the robustness of the proposed architecture with respect to security threats.

Focusing on LISP as reference protocol, in this paper we present a thorough analysis of the mapping cache. In a previous work, Iannone et al. analyzed a Netflow traffic trace from a small/medium sized university campus [11]. We build our work on the same methodology, but develop a new emulator in order to go beyond previous results and provide a deeper analysis. For this purpose, we use two 24-hour packet-level traces from a large European ISP, taken at different periods of the year. Further, Saucez et al. [24], in their security analysis of the LISP protocol, highlighted that a number of attacks can be avoided by slightly modifying the policy that LISP uses to encapsulate and decapsulate packets. Hence, we evaluate what are the implications, from a scalability point of view, of running LISP in such a more secure manner, which we call *symmetric LISP*.

The contributions of this paper are many-fold. We thoroughly analyze the behavior of a LISP Cache for traffic of a large number of DSL customers. Our results show that it is possible to maintain a high hit ratio with relatively small cache sizes, showing that there is no use in large timeouts. Compared to [11], we show how the LISP Cache has good scalability properties. We analyze what and how much would change if instead of running vanilla LISP, i.e., as defined in the specifications, symmetric LISP is used in order to increase security. Our results are important for ISPs in order to identify and quantify the resources needed to deploy LISP, with respect to the level of resiliency that they want to guarantee.

The remainder of this paper is structured as follows. In Section 2 we present the related work. In Section 3 we provide an overview of LISP in order to highlight how the LISP Cache works and the motivation for symmetric LISP. Section 4 describes how we collected and analyzed the traces used to obtain the results presented in Section 5. Finally, we draw our conclusions in Section 6.

2 Related Work

The concept of Locator/ID Split was already discussed, but never widely explored, in the late 90s ([23], [10]); rather used to solve specific issues, like cryptographic security with HIP (Host Identity Protocol [20]) or multi-homing for IPv6 with Shim6 [21]. The situation has changed after the rechartering of the RRG, with quite a number of proposals being discussed, either based on tunneling (e.g., LISP [7], HAIR [8]) or on address rewriting (e.g., ILNP [5], Six/One [26]). Despite the plethora of proposals, very few works have tackled the evaluation of such a critical component as the cache.

Kim et al. [27] studied route caching based on Netflow data, showing how caching is feasible also for large ISPs. However, they used a generic approach, not focused on LISP, and a limited cache size. Iannone et al. [11] have performed an initial study, without considering security issues, based on a single Netflow trace of a small/medium sized university campus. We adopt a similar methodology to provide comparable results, which we discuss in Section 6. Nevertheless, because we base our results on two 24-hour packet level traces captured in different periods we are able to provide a deeper analysis, including security aspects.

In the work of Zhang et al. [9], the authors propose a simple LISP Cache model with bounded size, using a Least Recently Used (LRU) policy to replace stale entries.[1] Their evaluation is based on a 24-hour trace of outbound traffic of one link of the CERNET backbone, the China Education and Research Network. In our analysis we take a different approach, assuming that the cache is not limited in size. This is a reasonable assumption since, as we will show later, even for large ISPs there is no need to use large caches and by tuning the expiration timer, caches do not grow so large as to hit memory limits of current routers.

Jakab et al. [14] propose a LISP simulator able to emulate the behavior of different mapping systems. However, they focused on the evaluation of the latency of their own mapping system solution, namely LISP-TREE, and compared it to other approaches but neglecting the analysis of the LISP Cache.

3 LISP Overview

The main idea of the Locator/ID Separation Protocol (LISP [7]) is to split the IP addressing space in two orthogonal spaces, one used to identify the end-hosts, and one used to locate them in the Internet topology. By re-using IP addresses, LISP is incrementally deployable and meant to be used on the border routers of stub domains, like in the scenario presented in Figure 1. Stub domains use internally an IP prefix, called EID-Prefix as for End-system IDentifier, which is part of the ID space. This prefix does not need to be globally routable, since it is used only for routing in the local domain. On the contrary, the core of the Internet, known as Default Free Zone (DFZ), will use globally routable IP addresses that are part of the locator space. In particular, the IP addresses used by stub domains' border routers on their upstream interfaces (toward the provider) are part of such a space and represent the Routing LOCators (RLOCs), since they allow to *locate* EID in the Internet. The binding between an EID-Prefix and the set of RLOCs that locate it is a *mapping*. EID-Prefixes are no longer announced in the DFZ, allowing a size reduction of BGP's routing table. Evaluating such kind of benefits is beyond the scope of this paper. Further information can be found in the work of Quoitin et al. [22] and Iannone et al. [12].

[1] In the rest of the paper we will use the term *LISP Cache* and *cache* interchangeably.

In the context of LISP, end-hosts generate normal IP packets using EIDs as source and destination addresses, where the destination EID is obtained, for instance, through a DNS query. LISP then tunnels those packets from the border router of the source domain to the border router of the destination domain, using the RLOCs as source and destination addresses in the outer header. For tunneling, LISP uses an IP-over-UDP approach, with the addition of a LISP-specific header between the outer UDP header and the inner (original) IP header.[2] In the LISP terminology, the border router in the source domain performing the encapsulation is called Ingress Tunnel Router (ITR), while the one in the destination domain, performing the decapsulation, is called Egress Tunnel Router (ETR). In general, they are just named xTRs. To perform tunneling operations routers use two data stores, namely the LISP Database and the LISP Cache.

The LISP Database stores mappings that bind the local EID-Prefixes (i.e., inside the local domain) to a set of RLOCs belonging to the xTRs deployed in the domain. The purpose of the LISP Database is two-fold. For outgoing packets, if a mapping exists for the source EID it means that the packet has to be LISP encapsulated and the source RLOC is selected from the set of RLOCs associated to the source EID-Prefix. For incoming packets, if a mapping exists for the destination EID, then the packets are decapsulated. The LISP Database is statically configured on each xTR. Its size is directly proportional to the number of the EID-Prefixes and xTRs that are part of the local domain. Due to its static nature and limited size, the LISP Database does not present any scalability issue and is not further analyzed in this paper.

The LISP Cache temporarily stores the mappings for EID-Prefixes that are not part of the local domain. This is necessary to correctly encapsulate outgoing packets, in particular to select the RLOC to be used as destination address in the outer header. Mappings are stored only for the time that are used to encapsulate packets, otherwise, after a timeout, they are purged from the cache. While critical for the dimensioning of the system, the LISP specification does not provide any value for this timeout, leaving the choice (or responsibility) to the implementers and system administrators. It should be clear that since the entries in the cache can expire, they are also entered in an on-demand fashion. This makes the cache so critical, since its content, size, and efficiency is totally traffic driven. In particular, the first outgoing packet, destined to an EID for which there is no mapping in the cache, triggers a cache-miss. Such an event, in turn, triggers a query message that the ITR sends to the Mapping Distribution System.[3] The latter is a lookup infrastructure designed to retrieve the mapping for the destination EID of the packet that triggered the query. This is the same principle of DNS, which allows to retrieve the IP address(es) of a server from its

[2] The LISP header contains information about RLOCs' reachability and traffic engineering. Details on this header can be found in the protocol specification ([7], [13]).

[3] Note that, in case of cache-miss, the packet that triggered it cannot be encasulated, since there is no mapping available. The LISP specifications do not explicitily describe what to do with the packet, however, it is out of the scope of the present work to evaluate this particular aspect.

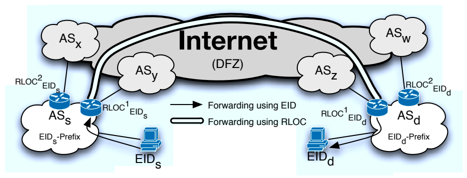

Fig. 1. Example of LISP deployment and communication.

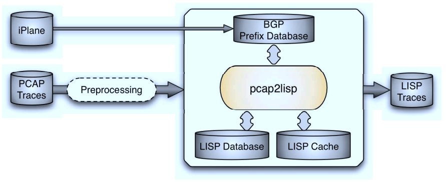

Fig. 2. Structure of the *pcap2lisp* emulator.

FQDN (Fully Qualified Domain Name). Insofar, several Mapping Distribution Systems have been proposed, but only the BGP-based ALT [6] is an official working item of the LISP Working Group. It is out of the scope of this paper to compare different mapping systems. Further information can be found in the work of Jakab et al. [14] and Mathy et al. [18].

The attentive reader should have noticed that there is a fundamental asymmetry in LISP when performing encapsulation and decapsulation operations. Indeed, while doing encapsulation the ITR uses the cache to locate what to put in the outer header, on the contrary, the ETR, while decapsulating the packet, does not use the cache. Rather, it just performs the checks described previously for the LISP Database and eventually decapsulates the packet.

This asymmetry leaves the LISP protocol vulnerable against specific attacks on the ETR, as pointed out by the work of Saucez et al. [24]. The authors describe attacks exploiting data packets as well as attacks that leverage on fields of the LISP header. Attacks are not limited to classic DoS and spoofing, but regard also cache poisoning, i.e., injecting wrong information into the cache, or cache overflow, i.e., saturating the cache and increasing the number of cache-misses. The analysis they performed, led the authors to conclude that to increase the level of security of the whole LISP architecture, the best solution is to use a symmetric model with a drop policy. Put differently, the authors suggest to only allow encapsulation *as well as* decapsulation if mappings are present in both the LISP Cache *and* the LISP Database. Otherwise, when the mapping is not in the cache a miss is generated and the packet is dropped. The rationale behind this suggestion is the assumption that the mapping distribution system is secured and trusted, hence, sanity checks can be performed at both ends, when encapsulating and when decapsulating. In particular, for the latter operation it means that if the packet contains information that is not coherent with the content of the cache, it is dropped. The implications of introducing the symmetric model to increase security are two-fold: on the one hand, performing additional checks when performing decapsulation may reduce the performances, however this is a common price for security mechanisms. On the other hand, this means that the size of the cache, its dynamics, and the control traffic overhead is increased.

In the second half of this paper, we will refer to the symmetric model as *symmetric LISP* and we will assess its impact as compared to vanilla LISP to evaluate what are the trade-offs to increase security in the LISP protocol.

4 LISP Emulation

To evaluate LISP we implemented *pcap2lisp*, which is meant to emulate the behavior of LISP's xTRs. The emulator is essentially designed to be fed with pcap-formatted traffic data and mimics as much as possible the LISP architecture; hence we based its implementation on two main modules (cf., Figure 2). The LISP Database, which is a manually configured list of internal network prefixes (EIDs). The LISP Cache, which stores EID-Prefixes and related statistic. Besides these two modules there is a central logic that creates the correct statistics, periodical reports that are written in logs, and that overviews the correct management of the cache timeouts. In addition, we use a local BGP prefixes database, fed with the list of BGP prefixes published by the iPlane Project [17]. The database is used to group EID-to-RLOCs mappings with the granularity of existing BGP prefixes, because, as for today, there is no sufficient information to predict what will be the granularity of mappings in a LISP-enabled Internet. This BGP granularity follows the methodology proposed in [11], thus making the results comparable. Furthermore, such an approach allows using the BGP database as a black box returning the mappings needed to feed the cache.

We fed *pcap2lisp*with two sets of anonymized traffic data collected within a large European ISP for 24 hours in April 2009 (APR09) and in August 2009 (AUG09) from a vantage point covering more than 20,000 DSL lines and already validated in other studies ([4, 15]).

5 Results

Similarly to the work in [11], we used three different cache timeout values, respectively 60 seconds, 180 seconds (i.e., three minutes), and 1,800 seconds (i.e., 30 minutes). The reason why we choose 60 seconds, instead of 300 minutes like in [11], is because that work already proved that the 300 minutes timeout value is inefficient considering the hit ratio vs. the cache size.

It is useful starting by identifying the working set, in this case represented by the number of observed BGP prefixes in our measurement environment. Figure 3 depicts the total number of contacted prefixes per minute, as well as the breakdown of incoming, outgoing, and bi-directional traffic. What we can identify in Figure 3 is that the majority of the observed prefixes (i.e., 70.1% in average in both traces) are bi-directional. One thing to keep in mind is that in the case of vanilla LISP, incoming packets do not play any role in the LISP cache, as explained in Section 3. On the contrary, when using symmetric LISP, incoming prefixes have an impact on the cache since the ETR does need a mapping. Hence, the 11.4% (average value in both traces) of incoming prefixes are a key differentiation between the two versions of LISP.

5.1 Vanilla LISP

A key benefit of LISP is the reduction of the routing table size in the DFZ [22], however, it exists a trade-off between the reduction of the routing table size and

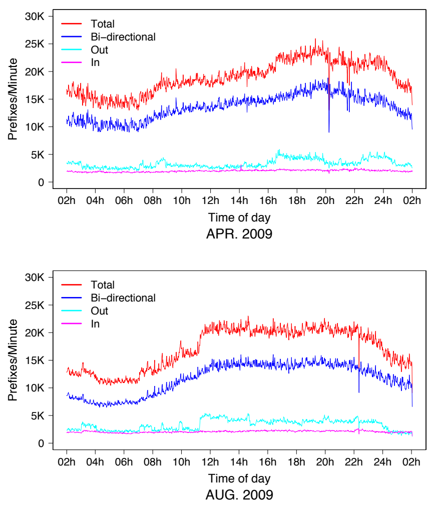

Fig. 3. Report of the number of correspondent prefixes per minute.

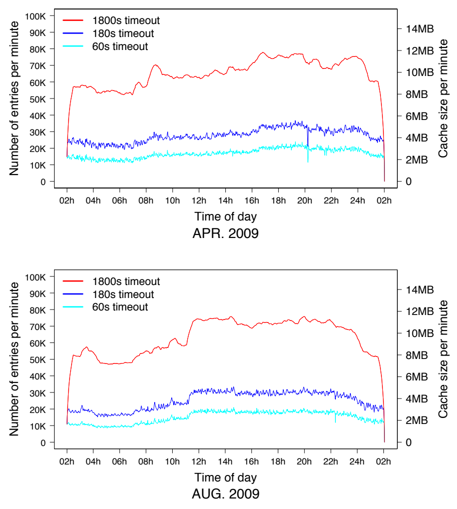

Fig. 4. Number of entries and cache size (assuming two RLOCs) per minute.

the LISP Cache size. The analysis of the caching *cost* is important for estimating the actual benefits of LISP. Figure 4 illustrates the number of entries and the size of the cache measured per one-minute time bin. Note that there are two measurement scales for the y-axis in the plot. While the left scale indicates the number of cache entries, the right scale indicates the size of the cache expressed in MBytes. The drop of the cache size at the end of the plot is due to the fact that we run the emulation until all cache entries are expired due to the timeout.

We can observe that the number of entries and the size of the cache are pretty low for the 60 and 180 seconds timeout cases (respectively slightly more than 10,000 and 20,000) and their curves show a spiky behavior. The 1,800 seconds timeout is much higher but with smoother changes, however, it takes about 30 minutes to reach the stable working set (almost 60,000 entries).

In the plot, and differently from [11], we calculate the size of the cache using the size of the data structure of a real LISP implementation, namely OpenLISP [3]. In OpenLISP each entry of the cache is a radix node containing all the information of the mapping and pointing to a chained list of RLOCs nodes. Thus, the size C_{size} of the cache is given by: $C_{size} = N_E \times (Radix_s + N_R \times RLOC_s)$. Where N_E and N_R represent respectively the number of entries in the cache and the number of RLOCs per entry. $Radix_s$ is the size of a radix node (56 bytes in OpenLISP), while $RLOC_s$ is the size of the RLOC node (48 bytes in OpenLISP). For the results in Figure 4 we assume two RLOCs per mapping.

Figure 5 depicts traffic volume and the traffic overhead of LISP expressed in MBytes/sec. This plot is based on 60 seconds timeout, because, as we will show next, the traffic overhead is in inverse proportion to the timeout value. Therefore, the estimation of the traffic overhead reported in this analysis shows

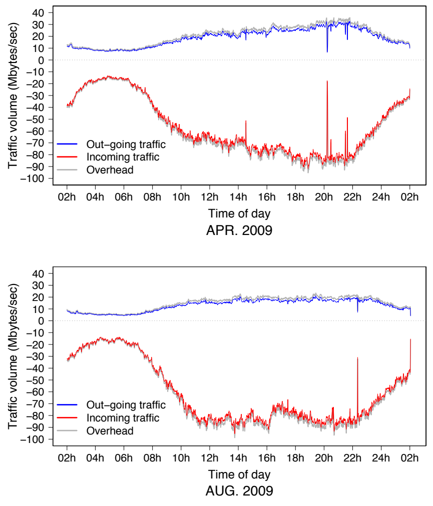

Fig. 5. Traffic volume and overhead due to LISP (60 seconds timeout value).

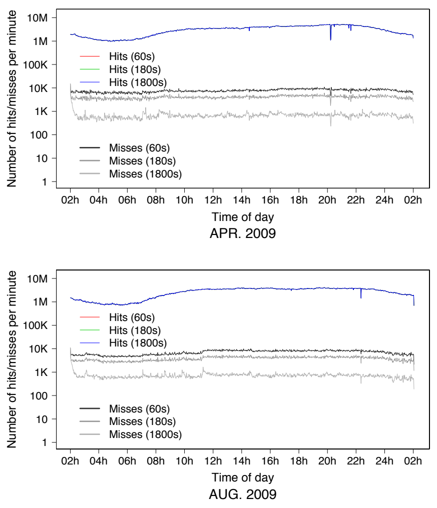

Fig. 6. Number of hits and misses per minute.

the maximum values. The line plotted in positive values indicates outgoing traffic, whereas the line plotted in negative values indicates incoming traffic. The shading over and under the traffic line indicates the traffic overhead. The overall overhead O can be evaluated by adding the overhead due to the LISP encapsulation, i.e., the number of packets N_P multiplied the size of the prepended header E_H and the volume of traffic generated to request and receive the missing mappings. Assuming the one single request/reply exchange is performed for each miss, the latter is given by the number of misses N_M multiplied by the size of the Map-Request message M_{REQ}, for outgoing traffic, or the size of the Map-Reply, for incoming traffic, which is given by a fixed size M_{REP} plus the size of each RLOCs record R times the number of RLOCs N_R. Note that the Map-Request and Map-Reply are the standard messages defined by LISP to request and receive a mapping from the mapping distribution system, independently from the specific instance of this latter. Putting everything together, for outgoing traffic we have: $O_{out} = N_{P_{out}} \times E_H + (N_M \times M_{REQ})$; while for the incoming traffic we obtain: $O_{in} = N_{Pin} \times E_H + (N_M \times (M_{REP} + N_R \times R))$. From the LISP specifications [7] the size of E_H is 36 bytes, the size of M_{REQ} is (without any trailing data) 24 bytes, the size of M_{REP} (without any trailing data) is 28 bytes, to which we need to add 12 bytes for each RLOC record R. With these numbers is possible to plot the overhead in Figure 5, which is between 3.6% and 5.2% for APR09 and 3.6% and 4.6% for AUG09.

As the previous analysis shows, the selection of an appropriate timeout value is of prime importance for the efficiency of the cache. Thus, we further analyze the implications of different timeout values. We first evaluate the efficiency of cache by investigating the ratio between cache misses and cache hits. Figure 6

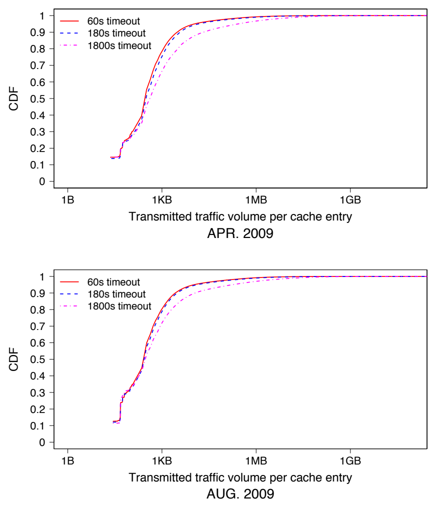

Fig. 7. Cumulative Distribution Function of the traffic volume per cache entry.

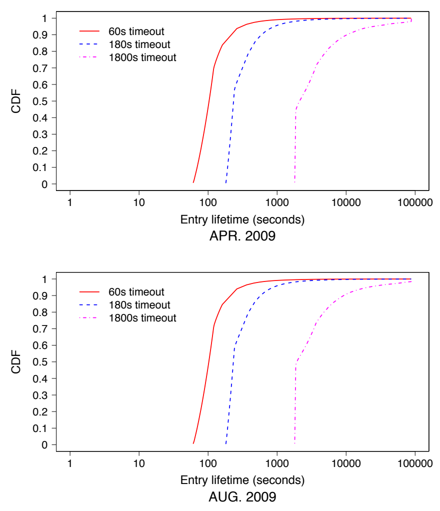

Fig. 8. Cumulative Distribution Function of entries' lifetime.

depicts the number of hits and misses in different timeout values. Even with the y-axis in logarithmic scale it is not possible to distinguish the lines of the cache hits, since they are overlapping each other. Hence, we need to look at the difference in the number of cache misses. Despite the identifiable difference, it is hard to avoid the claim that the benefit from the longer timeout value is insignificant compared to its cost in terms of size. Indeed, the hit ratio is always higher than 99% for every timeout value, while the size can be more than 5 times bigger when we compare the 60 seconds case with the 1,800 seconds case.

To confirm that a large cache is not useful, in Figure 7 we plot the CDF of the traffic volume forwarded by each entry, showing that the vast majority of cache entries carry less than 1 MBytes of traffic. Furthermore, Figure 8 shows how at least 50% of the entries have a lifetime only slightly longer than the timeout value. Both CDFs are the consequence of the well-known Internet phenomena that a small number of prefixes are responsible for the majority of Internet traffic. Coupled with the evidence mentioned above, we draw the inference that the minimum timeout value (60 seconds) is the most cost-beneficial.

5.2 Symmetric LISP

With respect to the security threats discussed in Section 3, we evaluate the extra cache size and the extra traffic overhead when the symmetric model is used. Since the extension of the mapping mechanism to incoming traffic is the main idea of the symmetric model, an increase in the number of entries and the size of the cache is expected and confirmed by the plot in Figure 9. For a clear comparison, the cache size shown in Figure 4 is presented again in gray color (lighter gray

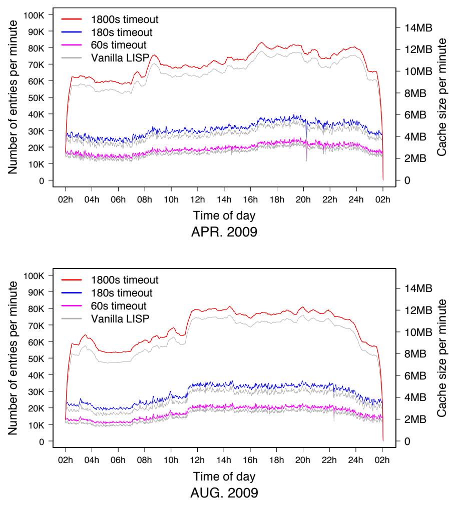

Fig. 9. Extra overhead caused by the security enhancement.

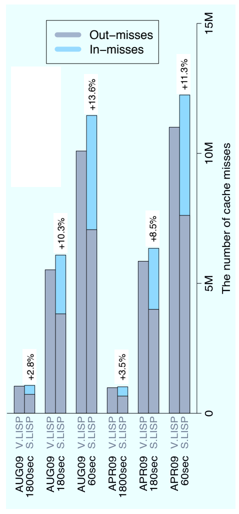

Fig. 10. Cache-miss for vanilla LISP and symmetric LISP.

Table 1. Cache size (in Mbytes) increase due to security enhancement (two RLOCs)

	APR09			AUG09		
Timeout	Vanilla	Symmetric	Difference	Vanilla	Symmetric	Difference
60 sec	3.47	3.76	+8.36%	3.04	3.34	+9.87%
180 sec	5.32	5.74	+7.89%	4.85	5.29	+9.07%
1800 sec	11.29	12.06	+6.82%	10.99	11.76	+7.01%

in black and white print). From Table 1, we see that there is a 6.8% to 9.9% increase in the cache size, when the symmetric model is used. Interestingly, the higher the timeout value, the lower the increase; we argue that it is due to the fact that longer timeouts increase entries' reuse probability.

Now, we expand the analysis to include the extra traffic overhead. Recall that each cache-miss, even for incoming traffic, triggers at least two messages (Map-Request and Map-Reply). Figure 10 compares the number of cache-miss under vanilla LISP and symmetric LISP. The most interesting result shown in the plot is the fact that the number of miss caused by incoming packets does not correspond to the increase in the number of cache-miss. Indeed, we observe that 2.8% to 13.6% of cache-miss are additionally produced under symmetric LISP, but the percentage of cache-miss due to incoming packets is much higher, while the number of cache-miss due to outgoing packets shrinks.

When translated into traffic overhead, the increase of miss generates a small increase in bandwidth consumption. Nevertheless, after applying the same computation like for vanilla LISP, the measured increase in bandwidth is always lower than 0.5%, hence not critical if not negligible.

6 Conclusion

The scalability issues of the current Internet have triggered an important amount of research on the Locator/ID Split paradigm. LISP, in particular, has gained momentum and seems to be a strong candidate to be widely adopted. The critical component in the LISP architecture, from a scalability and security point of view, is the LISP Cache. By using two 24-hour traces of a large European ISP, we thoroughly analyzed this component. In the present work we adopted the same methodology as in [11], in this way, the two works nicely complement and validate each other. An interesting comparison with the previous work is about the size of the cache vs. the number of users. In our case the size of the cache is almost doubled, however, in our traces there is more than three times the number of users than in the traces analyzed in [11]. This suggests that the average size of the LISP cache does not grow linearly with the number of end-systems. This is an important result, since it confirms that the LISP cache has good scalability properties. Future work will focus on further analyzing this aspect.

We went further in the analysis, mainly, but not only, in two points: the analysis of very short cache timeout and the analysis of the symmetric LISP model, which improves security. When the timeout value is as small as 60 seconds, the measurements prove that efficiency of the cache is still very high, with more than 99% hit-ratio, while the size reduces almost by half compared to the 180 seconds timeout case, and it is almost five times smaller compared to than the 1,800 seconds timeout case.

Another main contribution of the present work concerns the symmetric LISP model. The presented analysis shows that the increase in the size of the cache and the generated overhead is the order of 13% (for the 60 second timeout case). This looks like a very low cost compared to the security benefits, which have a very high value for ISPs and vendors.

Acknowledgments
This work is partially funded by the Trilogy Project (http://www.trilogy-project.org), a research initiative (ICT-216372) of the European Community under its Seventh Framework Program.

References

1. BGP Routing Table Analysis Report. *Online:* http://bgp.potaroo.net
2. International LISP Infrastructure. *Online:* http://www.lisp4.net
3. The OpenLISP Project. *Online:* http://www.openlisp.org
4. Ager, B., Schneider, F., Kim, J., Feldmann, A.: *Revisiting cacheability in times of user generated content.* 13th IEEE Global Internet Symposium (Mar 2010).
5. Atkinson, R.: *ILNP Concept of Operations.* IETF - Internet Engineering Task Force, draft-rja-ilnp-intro-03.txt (Feb 2010).
6. Farinacci, D., Fuller, V., Meyer, D., Lewis, D.: *LISP Alternative Topology (LISP+ALT).* IETF - Internet Engineering Task Force, draft-ietf-lisp-alt-04.txt (Apr 2010).

7. Farinacci, D., Fuller, V., Meyer, D., Lewis, D.: *Locator/ID Separation Protocol (LISP)*. IETF - Internet Engineering Task Force, draft-ietf-lisp-07.txt (Apr 2010).
8. Feldmann, A., Cittadini, L., Mühlbauer, W., Bush, R., Maennel, O.: *HAIR: Hierarchical Architecture for Internet Routing*. The Workshop on Re-Architecting the Internet (ReArch'09) (Dec 2009).
9. H, Z., Chen, M., Zhu, Y.: *Evaluating the Performance on ID/Loc Mapping*. The Global Communications Conference (Globecom'08) (Nov 2008).
10. Hiden, R.: *New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG*. IETF - Internet Engineering Task Force, RFC 1955 (Jun 1996).
11. Iannone, L., Bonaventure, O.: *On the Cost of Caching Locator/ID Mappings*. 3^{rd} International Conference on Emerging networking EXperiments and Technologies (CoNEXT'07) (Dec 2007).
12. Iannone, L., Levä, T.: *Modeling the Economics of Loc/ID Split for the Future Internet*. Future Internet Assembly (FIA) Book (Apr 2010).
13. Iannone, L., Saucez, D., Bonaventure, O.: *LISP Map Versioning*. IETF - Internet Engineering Task Force, draft-ietf-lisp-map-versioning-00.txt (Sep 2010).
14. Jakab, L., Cabellos-Aparicio, A., Coras, F., Saucez, D., Bonaventure, O.: *LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System*. IEEE Journal on Selected Areas in Communications (Sep 2010).
15. Kim, J., Schneider, F., Ager, B., Feldmann, A.: *Today's usenet usage: Characterizing NNTP traffic*. 13^{th} IEEE Global Internet Symposium (Mar 2010).
16. Li, T.: *Recommendation for a Routing Architecture*. IRTF - Internet Research Task Force, draft-irtf-rrg-recommendation-08.txt (May 2010).
17. Madhyastha, H.V., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., Venkataramani, A.: *iPlane: An Information Plane for Distributed Services*. 7th symposium on Operating Systems Design and Implementation (OSDI'06). USENIX Association (Nov 2006).
18. Mathy, L., Iannone, L.: *LISP-DHT: Towards a DHT to map Identifiers onto Locators*. The Workshop on Re-Architecting the Internet (ReArch'08) (Dec 2008).
19. Meyer, D., Zhang, L., Fall, K.: *Report from the IAB Workshop on Routing and Addressing*. IETF - Internet Engineering Task Force, RFC 4984 (Sep 2007)
20. Moskowitz, R., Nikander, P.: *Host Identity Protocol (HIP) Architecture*. IETF - Internet Engineering Task Force, RFC 4423 (May 2006)
21. Nordmark, E., Bagnulo, M.: *Level 3 Multihoming Shim Protocol for IPv6*. IETF - Internet Engineering Task Force, RFC 5533 (Jun 2009)
22. Quoitin, B., Iannone, L., de Launois, C., Bonaventure, O.: *Evaluating the Benefits of the Locator/Identifier Separation*. 2^{nd} ACM/IEEE Workshop on Mobility in the evolving internet Architecture (MobiArch'07) (Aug 2007).
23. Saltzer, J.: *On the Naming and Binding of Network Destinations*. IETF - Internet Engineering Task Force, RFC 1498 (Aug 1993).
24. Saucez, D., Iannone, L., Bonaventure, O.: *LISP Security Threats*. IETF - Internet Engineering Task Force, draft-saucez-lisp-security-01.txt (Jul 2010).
25. Saucez, D., Donnet, B., Iannone, L., Bonaventure, O.: *Interdomain Traffic Engineering in a Locator/Identifier Separation context*. The Internet Network Management Workshop (INM'08) (Oct 2008).
26. Vogt, C.: Six/One: *A Solution for Routing and Addressing in IPv6*. IETF - Internet Engineering Task Force, draft-vogt-rrg-six-one-01.txt (Nov 2007).
27. Kim, C., Caesar, M., Gerber, A., Rexford, J.: *Revisiting Route Caching: The World Should Be Flat*. Passive and Active Measurements Conference (PAM'09) (Apr 2009).