# Locator/ID Separation: Study on the cost of Mappings Caching and Mappings Lookups

## Technical Report N. 2007-04

Luigi Iannone and Olivier Bonaventure
Université Catholique de Louvain
Louvain-la-Neuve, Belgium
{firstname.lastname}@uclouvain.be

**Abstract**

Very recent activities in the IETF and in the Routing Research Group (RRG) of the IRTG focus on defining a new Internet architecture, in order to solve scalability issues related to interdomain routing. The approach that is being explored is based on the separation of the end-systems' addressing space (the identifiers) and the routing locators' space. This separation is meant to alleviate the routing burden of the Default Free Zone, but it implies the need of distributing and storing mappings between identifiers and locators on caches placed on routers. In this technical report we evaluate the cost of maintaining these caches and requesting these mappings when the distribution mechanism is based on a pull model. Taking as a reference the LISP protocol, we base our evaluation on real Netflow traces collected on the border router of our campus network. We thoroughly analyze the impact of the locator/ID separation, and related cost, showing that there is a trade-off between the dynamism of the mapping distribution protocol, the demand in terms of bandwidth, and the size of the caches.

## 1 Introduction

The ever-increasing growth of the Internet is raising scalability issues mainly related to interdomain routing, creating an increasing concern on the scalability of today's Internet architecture [1, 2]. These issues are mostly due to the use of a single numbering space, namely the *IP addressing space*, for both host transport sessions identification and network routing [1, 3, 4]. In addition to the single numbering space, multihoming and Traffic Engineering (TE) are making BGP's routing tables in the Default Free Zone (DFZ) to grow restlessly to a level where manageability and performances start to be critical [1, 5, 6].

Recently, both the IETF and the Routing Research Group (RRG) of the IRTG have started to explore the possibility to design a new Internet architecture, in order to solve the above-mentioned issues [2]. In particular, there is a fairly amount of activity around the approach based on the separation of the end-systems' addressing space (the identifiers) and the routing locators' space. This separation is perceived as the basic component of the future Internet architecture [7, 8, 9, 10, 11]. The main benefits that are expected to be obtained are the reduction of the routing tables size in the DFZ and improved TE capabilities. Indeed, the use of a separate numbering space for routing locators will allow to assign Provider Independent (PI) addresses in a topologically driven manner, improving aggregation while reducing the number of globally announced prefixes. Furthermore, it will also allow to perform both inbound and outbound flexible TE, by setting tunnels between different locators based on several different metrics or policies [12].

Even if it is generally accepted that locator/identifier separation is the way to go, there is no clue insofar on its cost and on the impact of this approach on the actual Internet. Indeed, as a counterpart of the above-mentioned benefits, there is the need to distribute and store mappings between identifiers and locators on caches placed on border routers. In the present work, we try to fulfill this lack by exploring and evaluating the cost of maintaining these caches and the overhead, in both terms of lookup queries and tunneling, introduced in the current Internet by this locator/identifier separation.

Taking as a reference the Locator/ID Separation Protocol (LISP [13]), we base our evaluation on real Netflow traces collected on the border router of our campus network. We estimate the cost of maintaining the locator/ID mapping caches when the distribution mechanism is based on a PULL model. By PULL model we intend a model where each time a mapping is necessary and not present in the local cache, a query is sent to a particular mapping distribution service. Note that we do not refer to any particular mapping distribution service, rather we explore what is the load and the dynamism such a system should bear with. Our analysis shows that there is a trade-off between the dynamism of the mapping distribution protocol, the demand in terms of bandwidth, and the size of the caches.

This technical report is organized as follows. In section 2, we describe the principles of the locator/ID separation paradigm, describing at the same time the LISP proposal and its variants. We illustrate how we collected and analyzed Netflow traces in section 3. The emulation of the LISP cache is described in section 4, right before detailing the outcomes of our measurements in section 5. The main results are then summarized in section 6.

# 2 Locator/ID Separation: How does it work?

There are several works, which can be found in the literature, that tackle the issue of separating end-host identifiers from routing locators (*e.g.*, [14, 15, 16, 17]). Nevertheless, seldom the proposed approaches can be incrementally deployed, since they have a disrupting impact on the current Internet architecture, needing the introduction of shim layers and/or heavy changes in end-systems' protocol stack.

On the contrary, the Locator/ID Separation Protocol (LISP), proposed by Farinacci *et al.* [13], has the nice property of being suitable for incremental deployment, without any impact whatsoever on end-systems. In the next section, we give a general overview of LISP. On the one hand, this allows, through a simple example, to clarify how the locator/ID separation paradigm works. On the other hand, since we will use LISP as a reference, this allows to explain the

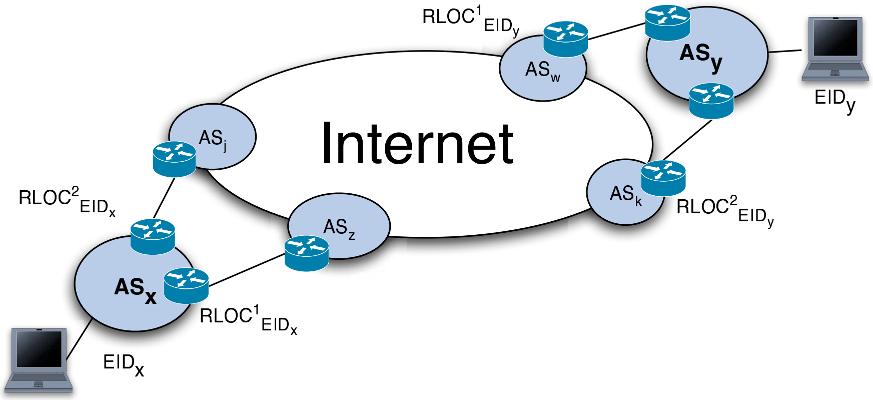


Figure 1: Position of EIDs and RLOCs in the global Internet.

basic mechanisms of the protocol. We will give further details about LISP and its variants in section 2.2.

## 2.1 LISP Overview

LISP is based on a simple IP-over-UDP tunneling approach, implemented typically on border routers, which act as *Routing LOCators* (RLOCs) for the end-systems of the local domain.[1] End-systems still send and receive packets using IP addresses, which in the LISP terminology are called Endpoint IDentifiers (EIDs). Remark that since in a local domain there may be several border routers, EIDs can be associated to several RLOCs.

The basic idea of LISP is to tunnel packets in the core Internet from the RLOC of the source EID to the RLOC of the destination EID. During end-to-end packet exchange between two Internet hosts, the Ingress Tunnel Router (ITR) prepends a new LISP header to each packet, while the Egress Tunnel Router (ETR) strips this header before delivering the packet to its final destination. In this way there is no need to announce local EIDs in the core Internet, but only RLOCs, which are necessary to correctly tunnel packets. As we demonstrated in our previous work [12], this last point allows to achieve the main objective of the locator/ID separation paradigm: the reduction of the size of BGP's routing tables.

In order to understand the main behavior of LISP, let us consider the topology depicted in Figure 1. For the sake of simplicity, we use the same acronyms to indicate both the name of the system and its IP address, *i.e.*, $\mathsf{EID_x}$ as well as $\mathsf{RLOC^2_{EID_y}}$ indicates both a name and an IP address. In this topology, the end-host $\mathsf{EID_x}$ is reachable through two border routers, meaning that it can

[1] Actually, LISP was defined as an IP-over-IP tunnel in the first draft [18]. The IP-over-UDP approach has been introduced only in the second draft [13] published the 29th of June 2007. In this last version an additional custom header is put right after the UDP header and before the original IP header. The purpose of this additional header is to add a basic level of security against spoofing by the exchange of a random value. Security considerations are out of the scope of the present technical report, thus, we will not detail the related issues. Interested readers can refer to the work of M. Bagnulo [19].

be associated to two RLOCs: $\mathsf{RLOC}^1_{\mathsf{EID_x}}$ and $\mathsf{RLOC}^2_{\mathsf{EID_x}}$. Similarly, $\mathsf{EID_y}$ has two locators: $\mathsf{RLOC}^1_{\mathsf{EID_y}}$ and $\mathsf{RLOC}^2_{\mathsf{EID_y}}$. Assuming that $\mathsf{EID_x}$ wants to open a connection to $\mathsf{EID_y}$, the following steps are performed:

1. $\mathsf{EID_x}$ issues a first IP packet, using its ID ($\mathsf{EID_x}$) as Source Address (SA) and using $\mathsf{EID_y}$ as Destination Address (DA). This packet is routed inside $\mathsf{AS_x}$ in the usual way in order to be delivered to one of $\mathsf{EID_x}$'s locators.

2. The ITR (*e.g.*, $\mathsf{RLOC}^1_{\mathsf{EID_x}}$) receives the packet. Remark that this is done in practice by intradomain routing or TE policies, coherently to the local EID-to-RLOC mapping. These policies can vary from one AS to another. Nonetheless, the EID is reachable from the outside through all of its RLOCs. Each RLOC is aware of local EID-to-RLOC mappings, which are stored in the local database.

3. $\mathsf{RLOC}^1_{\mathsf{EID_x}}$ performs the EID-to-RLOC lookup in its local cache, using Algorithm 1, to determine the routing path to the locator of $\mathsf{EID_y}$. Remark that all the entries of the cache can time out due inactivity. Note also that the creation of a new entry may imply several operations, including the transmission of a lookup query to a mapping distribution service. We will explain in the next section, and in further details, all the possibilities. By now, let us assume that the local cache lookup operation returns $\mathsf{RLOC}^2_{\mathsf{EID_y}}$.

4. A new LISP header is prepended to the original IP packet, having $\mathsf{RLOC}^1_{\mathsf{EID_x}}$ as SA and $\mathsf{RLOC}^2_{\mathsf{EID_y}}$ as DA. The packet is then routed at IP level in the Internet. Remark that core Internet routers do not need to have routes towards $EID_y$, in order to correctly forward the packet, only routes for the RLOCs are requested.

5. When the packet reaches $\mathsf{RLOC}^2_{\mathsf{EID_y}}$ the outer LISP header is stripped off. At the same time, a check is done on the local cache following Algorithm 2. This operation can result in the issue of some packets, as we will detail in the next section. The packet is then normally forwarded inside $\mathsf{AS_y}$ to be delivered to $\mathsf{EID_y}$.

Note that, after the first packet has gone through the LISP tunnel, the caches on both endpoints have the appropriate information to correctly forward all the subsequent packets, *i.e.*, all subsequent packets will always give a cache hit.

## 2.2 LISP Variants

LISP is defined in four different variants, depending on the "routability" of EIDs in the core Internet and on the type of mapping distribution protocol it is supposed to work with. These variants are: LISP 1, LISP 1.5, LISP 2, and LISP 3 [13]. They all work basically in the same way as we described in the previous section, except for the cache that can have a slightly different behavior, in particular in the case of a cache hit, depending on the variant. In the following sections, we describe the peculiarities of each variant.

**Algorithm 1** LISP Cache outgoing packets processing

```
 1: if ( ∃ EID-to-RLOC map in Cache for destination EID ) then
 2:     /* Cache Hit */
 3:     Update Timeout;
 4:     Select corresponding RLOC;
 5:     Return Selected RLOC;
 6: else
 7:     /* Cache Miss */
 8:     Create New Entry;
 9:     Set Timeout;
10: end if
```

**Algorithm 2** LISP Cache incoming packets processing

```
1: if ( ∃ EID-to-RLOC map in Cache for source EID ) then
2:     /* Cache Hit */
3:     Update Timeout;
4: else
5:     /* Cache Miss */
6:     Create New Entry;
7:     Set Timeout;
8: end if
```

### 2.2.1 Lisp 1 and 1.5

LISP 1 and 1.5 both assume that EIDs are routable IP addresses. The only difference is that LISP 1.5 assumes that routing based on EIDs is done via a "separate topology", however, in the draft, no details are given on this topology.

The fact that EIDs are routable addresses means that the mapping distribution protocol can be embedded into LISP itself, in the following way. When a packet arrives on the ITR, *i.e.,* the RLOC of the source address, and there is no corresponding cache entry, LISP will still prepend an LISP header to the original packet. Nevertheless, the DA of this LISP header is set to the destination EID (since it is a routable address), while the SA of the LISP header is set to the RLOC that is performing the encapsulation. The packet is then injected into the Internet. The information that a communication has been initiated toward an EID for which no mapping is known is kept in a particular queue for pending mappings. Thus, in Algorithm 1, in the case of a cache miss, step 8 is split in the followings steps.

```
Set DA in the LISP Header equal to destination EID;
Set SA in the LISP Header equal to me;
Put EID in Pending_Mappings_Queue;
```

At the other end of the tunnel, hence on the ETR, on the reception of such a packet, two main actions are performed. First, in the case that the received packet causes a cache miss, the new entry is created right away. Indeed, the SA of the LISP header is the RLOC of the SA of the inner header, which is an EID, hence there is a complete mapping information. Second, the ETR recognizes that the packet has been routed in the Internet using the EID for which it is one of its RLOCs. As a consequence, in order to announce that it is the RLOC for the destination EID (*i.e.*, communicating the EID-to-RLOC mapping), it sends a `Map-Reply` message (as a UDP packet) back to the ITR, which is known since it is the SA in the LISP header.

When the original ITR receives a `Map-Reply` packet for a pending mapping request, it creates the new entry in the cache. At this point, both routing locators have the complete mapping information. The steps described above mean that in Algorithm 2, before step 1, the following check is performed.

```
if ( DA of the incoming packet is an EID ) then
   Retrieve complete mapping from Database;
   Send Map-Reply packet to remote RLOC;
end if
```

The content of this `Map-Reply` packet is taken from the local LISP database of the ETR. The LISP *database* should not be confused with the LISP *cache.* While the former contains local mappings for local EIDs (*i.e.*, mappings concerning EIDs in the local domain), the latter temporarily contains mappings concerning remote EIDs with which a communication is ongoing.

There is already a proposal, called NERD [20], to distribute mappings in order to fill local databases. NERD assumes that there are one or more *external authorities* (*e.g.*, RIRs - Regional Internet Registries) that store and distribute mappings all over the Internet. While it is presumable that the LISP database will have a less dynamic behavior, compared to the LISP cache, such an approach is really static. The fact that external authorities distribute local mappings does not allow any type of intradomain TE (*e.g.*, to perform tasks like fast re-route or similar).

The NERD proposal, however, can be used as an EID-to-RLOC distribution protocol in the context of a PUSH model. By PUSH model we intend a mapping distribution protocol that "pushes" all mappings to each routing locator, without the need of explicit queries. This means that there would not be any LISP cache, but only a large database containing all Internet-wide mappings. We will no further explore this approach, since, as already stated, here we focus on a PULL model and because the static nature of NERD imposes several limitations (*e.g.*, impossibility of dynamic TE).

### 2.2.2 Lisp 2 and 3

Both LISP 2 and LISP 3 assume that the mapping distribution protocol is totally separated from the tunneling protocol (*i.e.*, LISP itself) and that EIDs are addresses not routable anymore in the core Internet. This means that, if not present in the cache, mappings need to be retrieved through an explicit query. Both variants assume that queries consist in a LISP `Map-Request` message at which the mapping distribution service must reply with a LISP `Map-Reply` message (see Farinacci *et al.* [13]). The difference between these two variants lays on the mapping distribution protocol, and related model, they assume will exist.

LISP 2 assumes a DNS (Domain Name System) based mapping distribution system, *i.e.*, a PULL model. This means that in both Algorithms 1 and 2, a cache miss and the consequent entry creation implies a lookup query to a DNS server. The advantage of such an approach is that the DNS can even be used in order to select interdomain paths (by choosing among the available RLOCs of the EID) that have good performances [21].

LISP 3, instead, assumes a Distributed Hash Table (DHT) based mapping distribution system. The DHT approach can either consist of a PUSH model, where all tunnel routers will have full knowledge of the mapping, or consist of a hybrid PULL/PUSH model, where tunnel routers will perform queries (PULL part) to a separate DHT system (PUSH part). In the hybrid approach, this means that, like for LISP 2, each time there is a cache miss and a new entry needs to be created, a lookup query is issued. Queries are sent to the DHT infrastructure, which is different from the DNS infrastructure. A first proposal of hybrid approach can be found in the work of Brim *et al.* [22]. A full PUSH model, which can be even based on a NERD-like approach, is impossible to evaluate without detailed specification on the mapping function and the mapping distribution protocol.

Remark that it is out of the scope of this paper to propose any mapping function, distribution model, or protocol. Here we limit our study to a PULL model in the context of a LISP-like approach. In particular looking at the dynamics of the cache temporarily storing remote mappings and exploring under what kind of trade-offs a PULL model can scale.

# 3 Netflow Traces Collection

During the end of May and the beginning of June 2007, we started to collect traces of the traffic from and to our campus network, which counts almost ten thousand active users/day. The network uses a class B /16 prefix block and is connected to the Internet through a border router that has a 1 Gigabit link toward the Belgian National Research Network (Belnet).

To collect the traffic, we rely on the Netflow [23] measurement facility supported by our border router. Netflow provides a record for each flow, containing information like the timestamp of the connection establishment, the duration, the number of packets, and the amount of bytes transmitted. The advantage of Netflow is that the traffic is collected and stored in a very compact way, allowing to easily collect daylong traces of several physical links. The drawback of Netflow is that traces have the granularity of flows, thus hiding characteristics like the burstiness of the traffic, however, this does not represent an issue in the present work. Thus, Netflow is less precise than traces collected with tools like, for instance, TCPDUMP [24]. These types of tools allow to collect each packet transiting through an interface. The drawback of this greater precision is that it is much more difficult to capture the traffic on several physical links and the post-processing task becomes much more complex. We use version 7 of the Netflow Collector (*i.e.*, the demon collecting the traffic), thus producing traces containing the version 7 of the flow records.

While collecting Netflow traces, we used the default configuration of our border router: a Cisco Catalyst 6509. In order to identify a flow, Netflow groups together all packets having the same source/destination IP addresses, source/destination ports, protocol interface, and class of service. Furthermore, there are three main configuration parameters that are used to shape flows (*e.g.*, to decide when a flow ends), but also to keep the Netflow table to an acceptable size. These parameters are:

| Parameter | timeout (sec) | packet threshold |
|---|---|---|
| normal aging | 300 | N/A |
| fast aging | 60 | 100 |
| long aging | 1920 | N/A |

Table 1: Netflow Configuration

**normal aging** - If no packets are received on a flow within the duration of the timeout set by this parameter the flow entry is deleted from the table, *i.e.*, it is considered as ended.

**fast aging** - The fast aging parameter uses a timeout value to check if at least a threshold value of packets have been switched for each flow. If a flow has not switched the threshold number of packets during the time interval, then the entry is aged out. Typical flows that are aged out by this parameter are short-lived DNS flows.

**long aging** - Long aging is used to prevent counter wraparound, which can cause inaccurate statistics. If a flow has a duration longer than the timeout set by this parameter, then it is aged out even if still in use.

Table 1 shows the default configuration values that we used for these parameters while collecting our traces. We will explain in section 5 the impact of these parameters on the cache emulation.

In order to analyze the traces we collected we use a two-step post-processing method. In a first step we analyze traces using the flow-tools [25], then, in a second step, we use our own filtering software to refine the results.

## 4 LISP Cache Emulation

As explained in section 2, under a PULL model all variants of LISP store mapping information concerning remote EIDs in a cache. We coded a small software able to emulate the behavior of such a cache, which can be fed with the Netflow traces we collected. This allows us to evaluate various parameters (*e.g.*, size, hits, misses, timeouts, etc) of the cache itself, but also to make some estimations of the lookup traffic.

In our analysis, we assume that the granularity of the EID-to-RLOC mapping is the prefix blocks assigned by RIRs. We call it /BGP granularity. In particular, we used the list of prefixes made available by the iPlane Project [26], containing around 240,000 entries. Using /BGP granularity means that each EID is first mapped on a /BGP prefix. The cache will thus contain /BGP to RLOC mappings.[2] This is a natural choice, since routing locators are supposed to be border routers.

Note that the granularity of the mapping has a deep impact on BGP's routing table size. Indeed, the more aggregation is performed on EIDs, (*i.e.*, the coarser is the granularity), the more the size of BGP routing table can be reduced [12]. It is important to recall that the reduction of the size of BGP's tables is the main target of the locator/ID separation paradigm [2].

[2] Note that both EIDs and RLOCs still remain full /32 IP addresses.

The /BGP granularity does not change the BGP's routing table size, hence, we can emulate the behavior of the LISP cache, as if LISP was deployed on the border router of our campus network. The /BGP granularity we used implies that the LISP database will contain a single entry, mapping all addresses in our class B /16 prefix block to our border router, which becomes the routing locator (RLOC) of all EIDs of our network.

The LISP cache emulator we implemented basically performs the two algorithms described in section 2.1 (*i.e.*, Algorithms 1 and 2). For each flow record, in order to apply the correct algorithm, the emulator looks at its direction (*i.e.*, if the flow is incoming or outgoing) and selects the correct prefix to look at. Then it performs the correct algorithm.

Remark that Netflow considers all flows as unidirectional, and even bidirectional sessions, like TCP, are stored as two distinct flows in opposite direction. Nevertheless, in the context of our emulation, there is no reason to rebuild sessions. Indeed, the unidirectional flow that starts the session will create the entry in the cache (if it is not already existing), while the flow that will close the session, thus ending later, will set up the correct value of the timeout. Even more, the direction of the flow that initiate the session is not important for the cache itself, since in both cases LISP assumes that a mapping for the remote EID needs to be inserted in the cache. The only difference is that if an outgoing flow creates an entry in the cache, then a lookup operation may be necessary, depending of the variant of LISP (see section 2.2).

We actually emulated all variants of LISP under a PULL model. We were thus able to provide statistics about the cache itself, the lookups (when necessary), and the generated `Map-Request`/`Map-Reply` traffic.

# 5 Measurements Results

In the following sections we present the main results we obtained from our analysis, by showing various measurements we performed on the traffic itself and on the outcome of our LISP cache emulator.

## 5.1 Correspondent Prefixes

Since we use /BGP as a granularity for EID-to-RLOC mappings, we first explore the characteristic of incoming and outgoing traffic using the number of correspondent prefixes as a metric. Figure 2 shows a one weeklong plot of the number of correspondent prefixes/hour, for both incoming and outgoing flows. Basically the plot shows the number of different correspondent prefixes contacted during each hour. As can be observed from the figure, the number of correspondent prefixes is more or less the same for both incoming and outgoing traffic. Nonetheless, this should not be interpreted as the fact that almost all flows are bidirectional. Indeed, in the same figure there is the plot of the total number of correspondent prefixes, obtained as:

$$Tot_{Pfxs} = \left| Pfxs_{In} \bigcup Pfxs_{Out} \right| . \qquad (1)$$

Where $Tot_{Pfxs}$ represents the total number of correspondent prefixes, while $Pfxs_{In}$ and $Pfxs_{Out}$ represent the sets of correspondent prefixes, respectively,

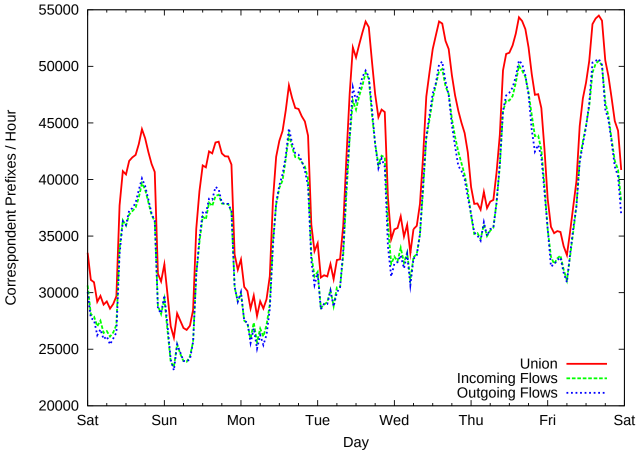


Figure 2: One week report of the number of correspondent prefixes per hour.

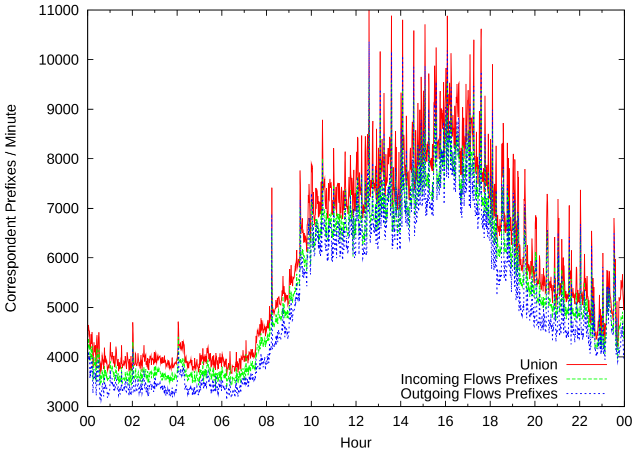


Figure 3: One day report of the number of correspondent prefixes per minute.

for incoming and outgoing traffic. The union operation avoids counting twice prefixes toward/from which there are bidirectional flows. As it can be observed, there is a difference of roughly 20%. Thus, there is a non-negligible amount of prefixes toward or from which there are unidirectional flows. It is useful to mention that the number of correspondent prefixes follows a night/day regular cycle, as the amount of traffic, with day peaks that can reach around 55,000 distinct prefixes in one hour, while during the night it can drop down to 25,000 prefixes/hour. Night/day cycles are an essential characteristic of Internet interdomain traffic [27].

A more detailed plot of the correspondent prefixes can be found in Figure 3, where we show a one daylong plot of the same measurements with a per minute granularity, *i.e.*, the figure shows the number of different prefixes contacted each minute. As it can be observed, the number of correspondent prefixes drops when we use such a granularity, which is not surprising. Nevertheless, the

20% difference is roughly preserved. This second picture allows us to highlight that, during daytime, the number of prefixes does not have such a smooth distribution as Figure 2 suggests. On the contrary, there are lots of spikes, due to the inherently “spikily” distribution of contacted prefixes, when observed at this granularity, caused by the important topological variability of the large majority of the interdomain traffic ([28, 27, 29]).

As already mentioned in section 4, LISP creates an entry in its cache no matter the direction of the flow and whether it is uni- or bi-directional. This means that when the timeout of the entries is set to a value larger than one minute, the total number of correspondent prefixes, shown in Figure 3, represents the lower bound of the size of LISP’s cache. Hence, representing the minimum number of entries that are present. This observation is confirmed by the results presented in the next section.

## 5.2 LISP Cache Emulation

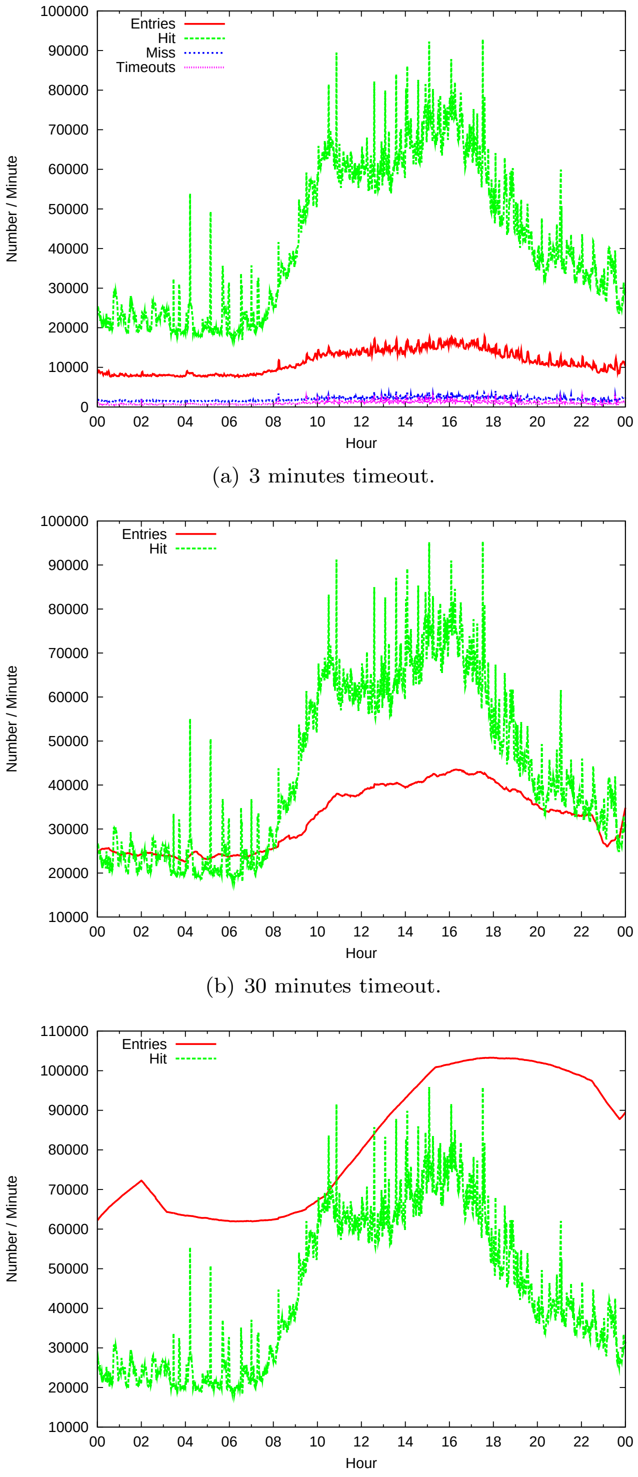
In section 4 we have described the LISP cache emulator. What we have not mentioned there is the value of the timeout associated to each entry. Indeed, if an entry is not used for at least the time set in the timeout, the entry is considered expired and it is purged. We performed LISP cache emulations using three different values for the timeout, namely three, thirty, and three hundred minutes. The general behavior of the LISP cache for the different values of the timeout is presented in Figure 4. In particular, the figure show the emulation, over one day, of the main parameters characterizing the cache, *i.e.*, size, number of hits, number of misses, and number of expired timeouts.

The size of the cache is expressed in number of entries and follows the day/night cycle of the traffic. The range of this cycle for the cache size depends on the timeout value. When using a three minutes timeout, the number of entries ranges from around 7,500 during the night, up to around 18,000 during the day. This means that the size of the cache has an increase of 140% between night and day. In the case of a thirty minutes timeout, the number of entries ranges from roughly 22,500 during the night, up to around 43,500 during the day. It can be observed that, as expected since entries live longer, the average size of the cache is larger, while the size during the day is almost the double (93% actually) compared to the night period. This is not the case when using a three hundred minutes timeout. Indeed, the number of entries ranges from 62,000 up to 103,000, meaning an increase of around 60%. Thus, the longer the timeout value, the larger the average cache size and the smaller is the percentage of variation between night and day.

As explained in section 2, due to multihoming, an EID (or a /BGP prefix as in our case) can be associated to more than one RLOC. We can assume that each set of EIDs (/BGP prefix) can be represented by 5 bytes, *i.e.* IP prefix and prefix length. Concerning RLOCs, we can consider that they have a size of 6 bytes. This because LISP associates to each RLOC (4 bytes IP address) two values: its *Priority* (one byte) and its *Weight* (one byte) [13]. These parameters are supposed to be used for traffic engineering purposes. With these values we can estimate the size of the cache as follows:

$$S = E \times (5 + N \times 6 + C)\ . \tag{2}$$

Where $S$ is the size of the cache expressed in bytes, $E$ is the number of entries,



(a) 3 minutes timeout.

(b) 30 minutes timeout.

(c) 300 minutes timeout.

Figure 4: One day report of the behavior of LISP cache.

| Timeout | Period | 1 RLOC | 2 RLOCs | 3 RLOCs |
|---|---|---|---|---|
| 3 Min. | Night | 139 | 183 | 227 |
| | Day | 334 | 440 | 545 |
| 30 Min. | Night | 417 | 550 | 681 |
| | Day | 807 | 1062 | 1317 |
| 300 Min. | Night | 1132 | 1490 | 1847 |
| | Day | 1917 | 2522 | 3127 |

Table 2: LISP cache size estimation (in KBytes).

$N$ the number of RLOCs per EID, and $C$ represents the overhead in terms of bytes necessary to build the cache data structure. Assuming the cache is organized as a tree, $C$ can be set to 8 bytes, just the size of a pair of pointers. In Table 2, we provide an estimation, for both day and night periods and for all the three timeout values, of the size of the cache, expressed in KBytes, when for each /BGP prefix there are up to three RLOCs. Depending on the timeout value, the size of the cache can range from a bit more than a hundred KBytes, up to few MBytes.

Always from Figure 4, it can be observed that the large majority of the flows already find a mapping in the cache when they start. Indeed, the number of hits ranges from around 20,000 hits/minute during the night, up to more than 70,000 hits/minute during the day, with spikes that reach 90,000 hits/minute. The value of the timeout does not have a large impact on the number of hits. Indeed, the corresponding curve remains almost unchanged in all the three cases. It is important to remark that in our analysis, the number of hits is slightly overestimated. As we explained in section 3, long lasting flows are divided in smaller flows that last at most *long aging* time. This split has no impact on the number of entries in the cache, since the first chunk will create the entry (if necessary) and all subsequent chunks will not. Nonetheless, the number of hits results biased. Indeed, all subsequent chunks will generate a hit, which is not conform to reality. However, the number of flows that last more than 32 minutes (this is the value the long aging parameter was set) are very few, thus the resulting overestimation is negligible.

Differently from the number of hits, the number of misses and expired timeouts has a deep change when changing timeout value. We plotted these two curve only in Figure 4(a), where are still visible. We did not plot them in Figure 4(b) and 4(c) since they become practically invisible due to the range of the vertical axis. We summarize in Table 3 and 4 the ranges of values for each timeout value. With the number of misses/minute being slightly higher than the number of expired timeouts/minute. In Figure 4(a), however, it can be observed that also the number of expired timeouts and misses per minute have a night/day cycle, with minimum during the night and maximum during the day.

In order to better understand the dynamics of the LISP cache, we plot in Figure 5 the Cumulative Distribution Function (CDF) of the entries' lifetime. Obviously the distribution is lower bounded by the timeout value, as it can be seen in the figure. What is interesting to remark is that the large majority of the entries have a lifetime slightly higher than the timeout value. This means that the large majority of the flows have a very small duration and are directed

| Timeout | Min (Night) | Max (Day) |
|---:|---:|---:|
| 3 Min. | 1250 | 4300 |
| 30 Min. | 250 | 1100 |
| 300 Min. | 30 | 320 |

Table 3: Expired timeouts per minute.

| Timeout | Min (Night) | Max (Day) |
|---:|---:|---:|
| 3 Min. | 1300 | 4050 |
| 30 Min. | 260 | 1200 |
| 300 Min. | 20 | 330 |

Table 4: Number of cache misses per minute.

or come from prefixes that are not contacted so often. On the other hand, the distribution shows also that a small number of entries can have a lifetime as long as the whole period of observation, *i.e.*, 24 hours. This does not mean that there are flows of such a length, but that there are a small number of prefixes that are contacted as often as the size of the timeout. This observation is also corroborated by the fact that, as it can be seen in the figure, the larger the timeout, the lower is the "knee" of the CDF. Meaning that more and more flows help to keep alive a larger number of cache entries.

We also evaluated the CDF of the amount of bytes that are forwarded using each particular cache entry. Recall that each entry is used to set up the correct content in the LISP header that is prepended to the original packet. The result is shown in Figure 6. Obviously, the longer an entry lasts in the cache, the higher the probability to be used. This explains why the higher the timeout value, the higher the ratio of entries that are used to forward a large volume of traffic. For instance, when using a three minutes timeout, 99.5% of the entries are used to forward less than 1 MBytes of traffic, while when using a three hundred minutes timeout, this percentage drops to 95.5%. On the other hand, for the three values of timeout, there are entries that are used to forward several GBytes of traffic. These results need to be interpreted carefully. It is known that there exist several different classes of flows [30]. It is also known that flows are very volatile in terms of volume, changing their behavior on the flight [31]. Due to the granularity of Netflow traces, we are not able to explore these issues, however, as far as the locator/ID mapping is concerned, the presented analysis provide sufficient insight on the subject.

## 5.3 Mapping Lookups

The measurements described in the previous section concern parameters related to the LISP cache itself. Nevertheless, they can be used to estimate other parameters related more generally to the locator/ID separation and the different variants of LISP.

Assuming a PULL model for mapping information distribution, like for example in LISP 2, which assumes a DNS-based solution, we are able to estimate the number of lookups/minute our border router would issue. Remark that in the context of LISP 2, a mapping lookup means to send a `Map-Request` message

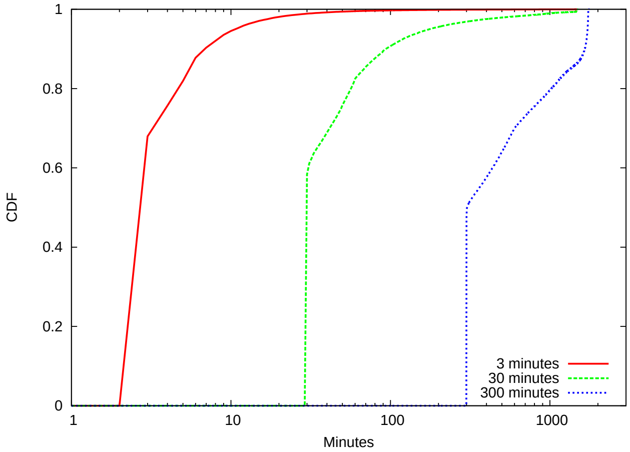


Figure 5: Cumulative Distribution Function of the cache entries lifetime.

| Timeout | Period | 1 RLOC | 2 RLOCs | 3 RLOCs |
|---|---|---|---|---|
| 3 Min. | Night | 4 | 4.9 | 5.7 |
| | Day | 24.4 | 29.2 | 34 |
| 30 Min. | Night | 0.814 | 0.974 | 1.14 |
| | Day | 8.2 | 9.7 | 11.3 |
| 300 Min. | Night | 0.041 | 0.049 | 0.057 |
| | Day | 2.36 | 2.82 | 3.29 |

Table 5: Incoming volume of traffic concerning `Map-Reply` messages (in Kbit/sec).

and to receive a `Map-Reply` message from the DNS. A lookup query to the mapping distribution system is issued whenever there is an outgoing flow for which there is no corresponding entry in the cache, *i.e.*, there is no EID-to-RLOC map present for the destination prefix. Thus, counting the number of cache miss for outgoing flows gives us the estimation we look for. Figure 7 shows the result of this counting for a daylong period for all the three values of timeout we used for the emulation. We find again a night/day cycle, with minimum values that can drop to 30 queries/minute when using a three hundred minutes timeout, growing to a maximum of 3,000 queries/minute when using a three minutes timeout. Table 5 shows the corresponding volume of incoming traffic generated by the lookup replies (*i.e.*, LISP `Map-Reply` messages), when, for each EID, up to three RLOC are returned. As it can be remarked, the volume never grows over few tens of Kbit/sec.

Note that, insofar, we did not take into account incoming flows that generate a cache miss, since, in the context of LISP, there is no lookup query generated. Indeed, the mapping can be retrieved by looking at the source address of the outer LISP header and the source address of the inner IP header. This, however, brings to light a limitation of the LISP proposal. Indeed, we are emulating a cache that has a /BGP granularity, while from incoming packet it can be retrieved only a single /32 EID-to-RLOC map. In order to populate the cache with the correct entries there are two possible solutions.

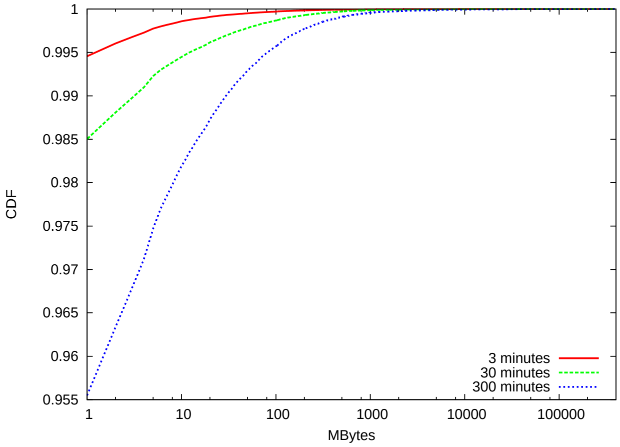
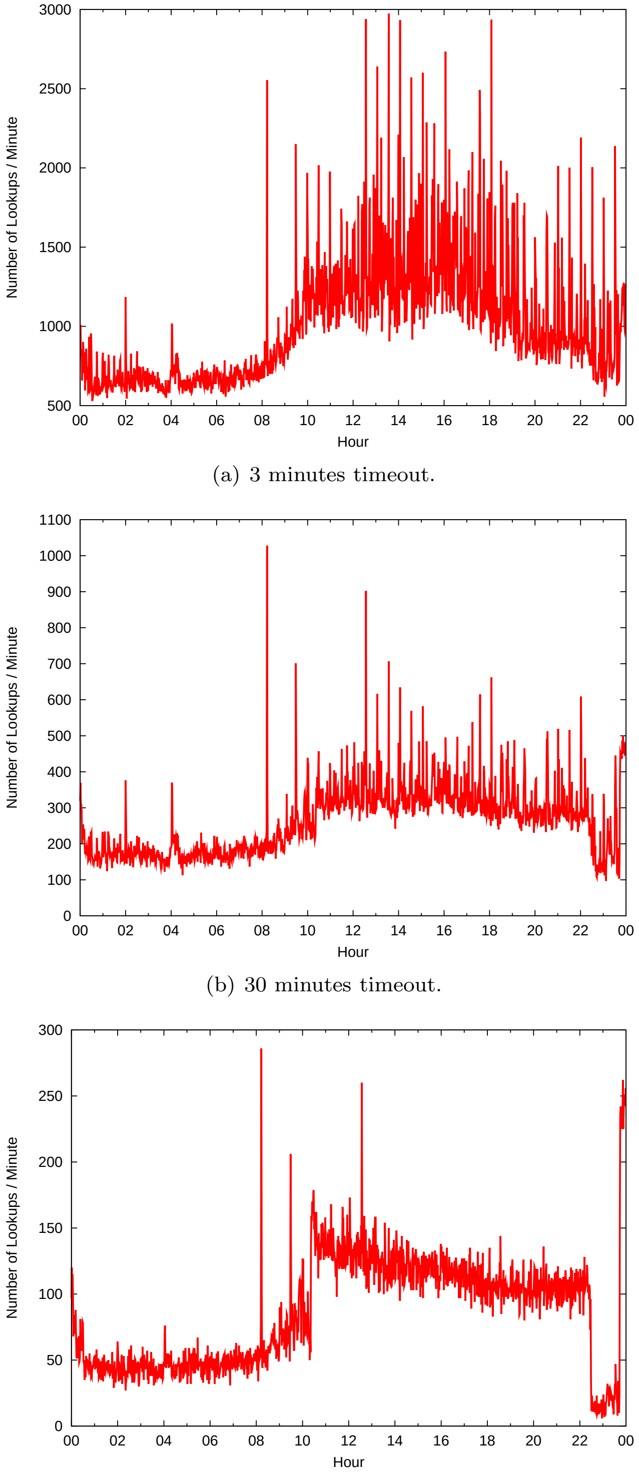


Figure 6: Cumulative Distribution Function of the volume of traffic per cache entry.

The first solution is to make LISP have a local copy of all announced prefixes. In this way, when the first packet of a new incoming flow arrives, the source address of the inner header is mapped on the corresponding announced prefix and an entry is then created in the cache. This solution, while advantageous in terms of latency and bandwidth, since an entry can be created after a simple local lookup, has the drawback of needing to store the whole list of /BGP prefixes. This in turns raises issues related to how to keep the list up to date and what amount of space this list will consume. In today's Internet, we have more 240,000 prefixes [1], that need some few Mbytes of storing space, which is far larger of the size of the cache when using a three minute timeout. Remark that this approach is somehow similar to the NERD solution. However, differently from NERD, here there is only a database containing all possible EIDs' sets, while the association to RLOCs is done dynamically, thus enabling dynamic TE.

The second solution is to issue a lookup also for incoming flows, thus enforcing a full PULL model for mapping distribution. This solution is not storing space consuming, and has no problem related to the freshness of information. Nevertheless, this roughly means to have an increase of 150% in the number of queries during night period and to have spikes 30% higher during the day, in the case of a three minutes timeout, as shown in Figure 8. The increase is less important when using the other values of the timeout, as can also be seen in Table 6, which summarizes the volume of incoming reply traffic in this case.

As already explained in section 2.2.1, in the most simple LISP variants, namely LISP 1 and LISP 1.5, for each new incoming flow that has not a corresponding entry in the cache, a `Map-Reply` packet is sent back to the RLOC of the source EID in order to communicate the mapping to use. Similarly, for each outgoing flow that creates a new entry in the cache, a `Map-Reply` packet, containing the mapping of the destination EID, will be received. Figure 9 shows the number of incoming and outgoing `Map-Reply`/minute, for the three different timeout values. As it can be observed, the larger the timeout value, the lower the amount of packet that are sent/received. The number of incoming `Map-Reply`/minute

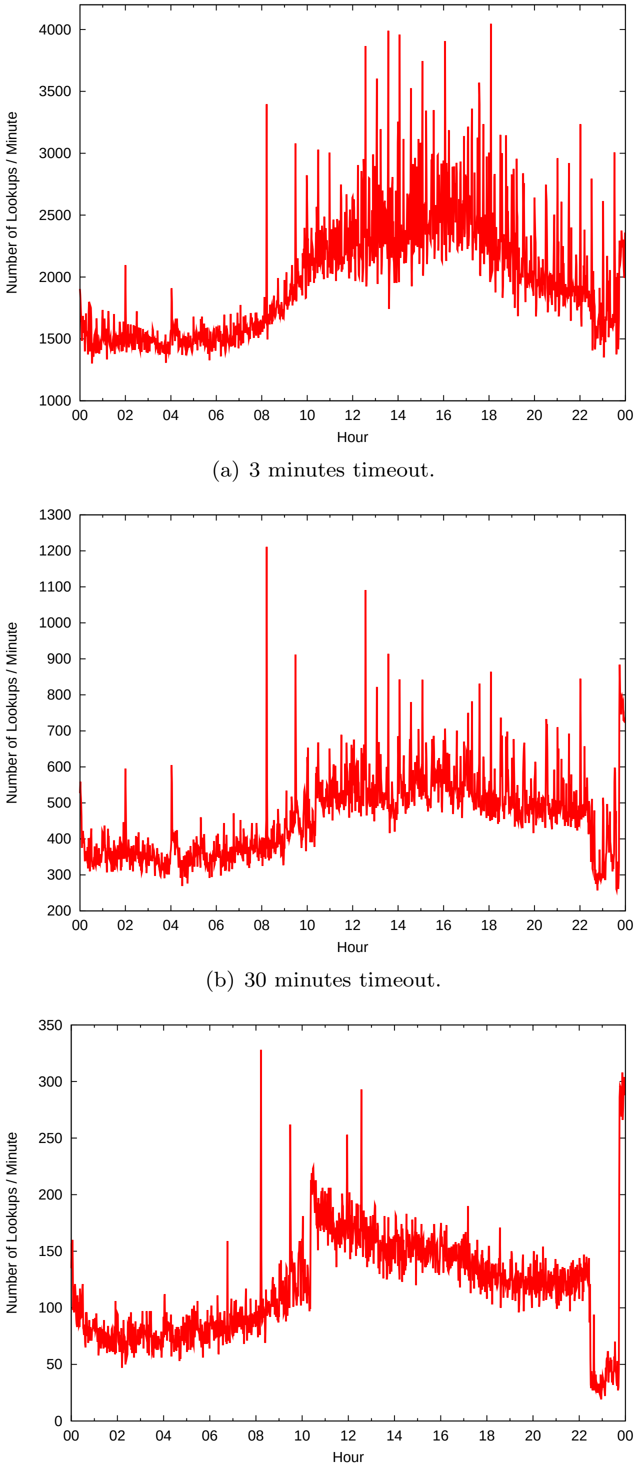


(a) 3 minutes timeout.

(b) 30 minutes timeout.

(c) 300 minutes timeout.

Figure 7: Number of lookup queries per minute in the context of a PULL model.



(a) 3 minutes timeout.

(b) 30 minutes timeout.

(c) 300 minutes timeout.

Figure 8: Number of lookup queries per minute in the context of a full PULL model.

| Timeout | Period | 1 RLOC | 2 RLOCs | 3 RLOCs |
|---|---|---|---|---|
| 3 Min. | Night | 10.17 | 12.17 | 14.17 |
| | Day | 34.97 | 41.85 | 48.74 |
| 30 Min. | Night | 2.04 | 2.44 | 2.84 |
| | Day | 8.95 | 10.71 | 12.47 |
| 300 Min. | Night | 0.163 | 0.195 | 0.227 |
| | Day | 2.68 | 3.21 | 3.74 |

Table 6: Incoming volume of traffic concerning `Map-Reply` messages in the context of a full PULL model (in Kbit/sec).

is always higher and more spiky than the corresponding number of outgoing `Map-Reply`/minute, no matter the timeout value used. This suggests that the entries in the LISP cache are more often created by local outgoing traffic.

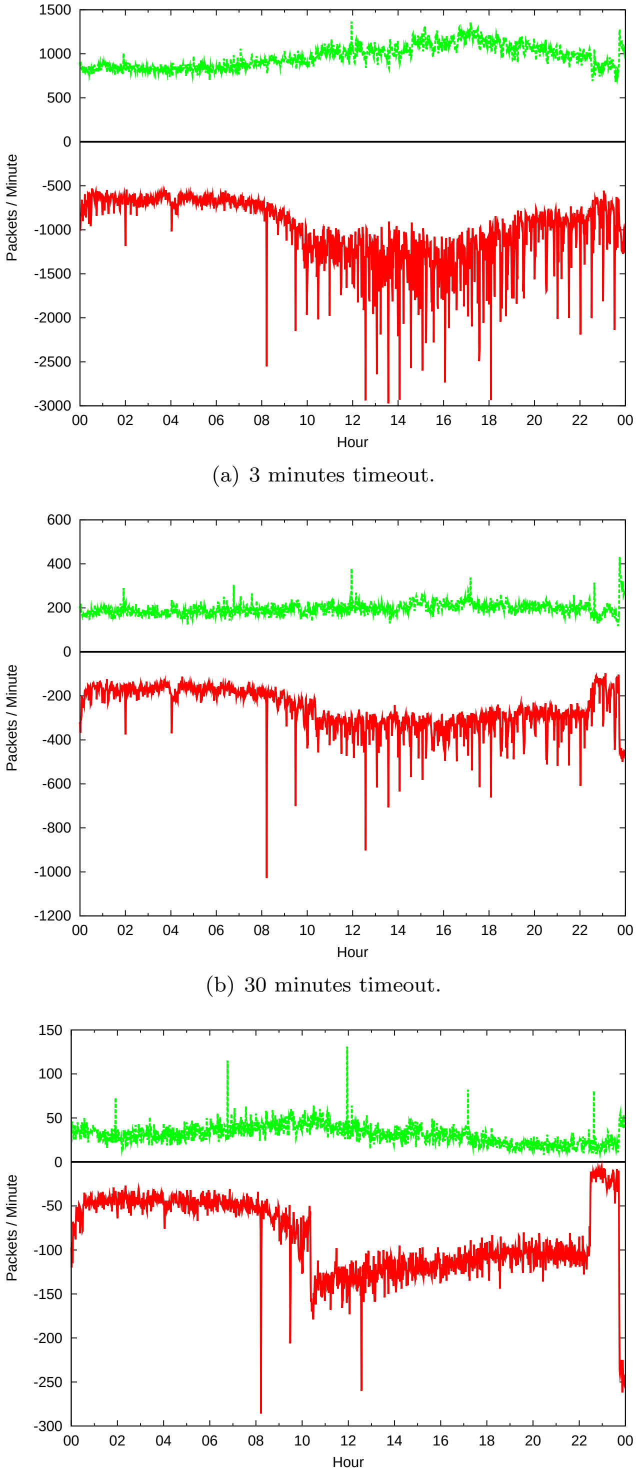
Insofar, we thoroughly analyzed the traffic load generated by mapping lookup in both a PULL model, as suggested by LISP, and a (more general) full PULL model. In the PULL model, the mapping for incoming flows is retrieved from the flow itself. In the full PULL model mappings for both incoming and outgoing flows are retrieved by sending a query to a mapping distribution service. Nevertheless, it is important to be able to estimate the real value of the measurements we presented. For this purpose, and in order to have a reference toward we can compare to, we measured the DNS traffic outgoing from our campus network, which is depicted in Figure 10. The figure shows a daylong report of the number of DNS packets (actually UDP packets destined to port 53) sent each minute. As it can be observed, the number of packets/minute ranges from levels as low as 1,800 packets/minute, up to 15,000 packets/minute. This range is higher than the range of values for lookups/minute sent when using the short three minutes timeout (*cf.* to Figure 8). This is not sufficient to let us state that an extension in the DNS service, in order to distribute mappings, could be implemented, since, even if smaller than the existing DNS load, it is however not negligible. Nevertheless, it proves that a mapping distribution system based on the existing DNS protocol (not the DNS service), or a similar system [32], can easily perform the task, at least in the static case.[3]

## 5.4 Traffic Volume overhead

The locator/ID separation paradigm is based on tunnels set up between RLOCs, which introduce an overhead in terms of traffic volume. As a final evaluation, we measured this overhead, for both incoming and outgoing traffic, when LISP is used. Remark that the size of the prepended LISP header is the same for all the variants.

Figure 11 shows a one daylong report of the volume of traffic expressed in Mbit/sec. Positive values are for outgoing traffic, while negative values for incoming traffic. As the figure shows, the overhead introduced by the tunneling approach consists in few Mbit/sec. For outgoing traffic this means an overhead that ranges from 15% during the night down to 4% during the day. For incoming traffic this means an overhead that ranges from 10% during the night down to

[3] This may not hold anymore if locator/ID separation will be also used to manage mobility, as suggested in some discussions in the IRTG.



(a) 3 minutes timeout.

(b) 30 minutes timeout.

(c) 300 minutes timeout.

Figure 9: Number of `Map-Reply` messages per minute (negative values express incoming packets).

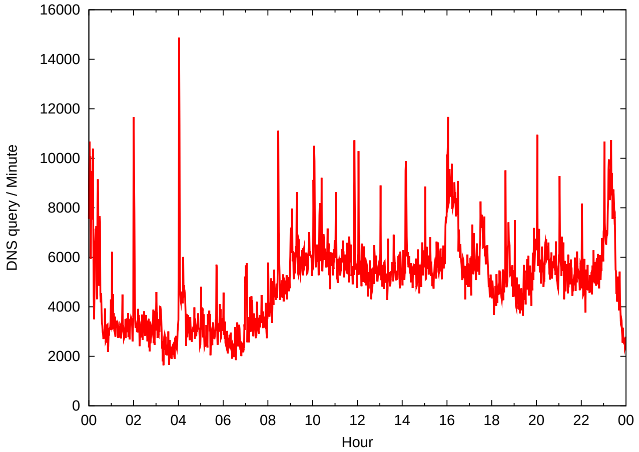


Figure 10: Measured number of DNS queries per minute.

2% during the day. Remark that this overhead does not depend on the mapping function or the mapping distribution protocol.

# 6 Conclusions

Recent research activities focus on separating the existing IP addressing space, in order to mark the distinction between end-systems' identifiers and routing locators. This approach is meant to overcome the scaling issues that the actual Internet architecture is facing. Nevertheless, there is still no clear indication of the cost that this approach will have in terms of bandwidth and size of data structures. Furthermore, there is no hint on the level of dynamism that will have the protocol distributing the mappings of the end-systems' identifier space into the routing locators' space.

The present technical report provides a complete evaluation of these open questions. We based our analysis on real Netflow traces collected from our campus network, which we are currently anonymizing to make them available to the research community. We fed the traces to a software, coded by ourselves, emulating the behavior of the LISP cache. The analysis is done in the context of a PULL model, as suggested by LISP, but we extended it to a more general full PULL model. We showed that the size of the cache maintaining the mappings can be limited in size by using a relatively small timeout for the entries. Nevertheless, this increases the traffic generated for mapping lookups, which is never negligible. On the other hand, this traffic is smaller than existing DNS traffic, demonstrating that a similar architecture is suitable for mapping distribution. Maintaining large caches would reduce the amount of lookup traffic, however, as we showed in the paper, a huge amount of flows have a short lifetime, thus a large number of cache entries are used to forward a small amount of traffic. Finally, we also showed that the overhead introduced by the tunneling, on which the locator/ID separation paradigm is based, does not pose any problem since it is quite small.

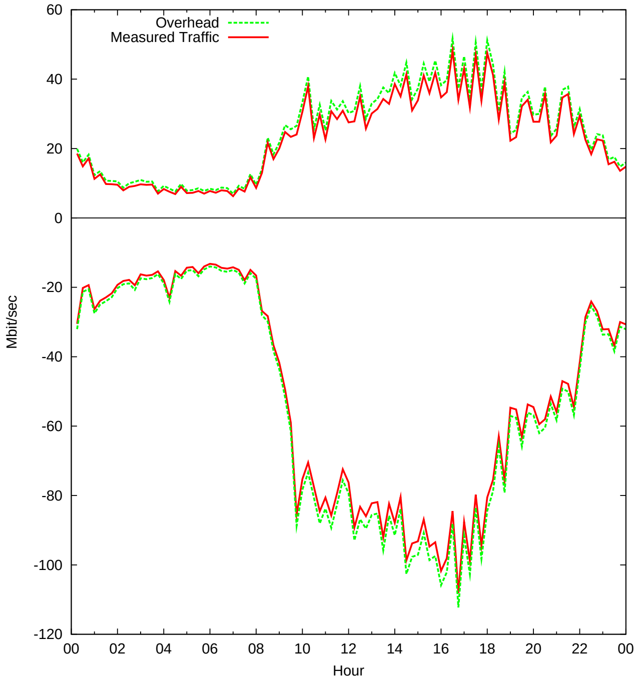


Figure 11: Traffic and correspondig tunneling overhead in a one daylong report (negative values express incoming traffic).

## Acknowledgments

This work has been partially supported by the European Commission within the IST-027609 AGAVE project (`www.ist-agave.org`). We would like also to thank the iPlane Project, for the dataset they made available.

# References

[1] G. Huston, "The growth of the bgp table - 1994 to present." [Online]. Available: `http://bgp.potaroo.net`

[2] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing," Internet Draft draft-iab-raws-report-01.txt, February 2007.

[3] J. Chiappa, "Endpoints and Enpoint names: A Proposed Enhancement to the Internet Architecture," Tech. Rep., 1999. [Online]. Available: `http://www.chiappa.net/∼jnc/tech/endpoints.txt`

[4] J. Saltzer, "On the Naming and Binding of Network Destinations," RFC 1498, August 1993.

[5] S. Agarwal, C.-N.Chuah, and R. H. Katz, "OPCA: Robust Interdomain Policy Routing and Traffic Control," *Proc. 6th International Conference*

*on Open Architectures and Network Programming (OPENARCH)*, April 2003.

[6] L. Zhang, "An Overview of Multihoming and Open issues in GSE," *IETF Journal*, vol. 2, September 2006.

[7] O. Bonaventure, "Reconsidering the Internet Routing Architecture," Internet Draft draft-bonaventure-irtf-rrg-rira-00.txt, March 2007.

[8] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture," RFC 4423, May 2006.

[9] E. Nordmark and M. Bagnulo, "Level 3 Multihoming SHIM Protocol," Internet Draft draft-ietf-shim6-proto-07.txt, May 2006.

[10] M. O'Dell, "GSE - An Alternate Addressing Architecture for IPv6," Internet Draft draft-ietf-ipngwg-gseaddr-00.txt, 1997.

[11] X. Zhang, P. Francis, J. Wang, and K. Yoshida, "Scaling Global IP Routing with the Core Router-Integrated Overlay," *Proc. 14th IEEE International Conference on Network Protocols (ICNP)*, November 2006.

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

[13] D. Farinacci, V. Fuller, D. Oran, and D. Meyer, "Locator/ID Separation Protocol (LISP)," Internet Draft draft-farinacci-lisp-01.txt, June 2007.

[14] R. Hiden, "New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG," RFC 1955, June 1996.

[15] D. Massey, L. Wang, B. Zhang, and L. Zhang, "A Proposal for Scalable Internet Routing & Addressing," Internet Draft draft-wang-ietf-efit-00.txt, February 2007.

[16] B. Quoitin, *BGP-based Interdomain Traffic Engineering.* Université Catholique de Louvain, PhD Thesis, August 2006.

[17] X. Yang, "NIRA: a new internet routing architecture," *Pro. ACM SIGCOMM workshop on Future directions in network architecture (FDNA)*, August 2003.

[18] D. Farinacci, V. Fuller, and D. Oran, "Locator/ID separation protocol (LISP)," Internet Draft draft-farinacci-lisp-00.txt, January 2007.

[19] M. Bagnulo, "Preliminary lisp threat analysis," Internet Draft draft-bagnulo-lisp-threat-00.txt, 2007.

[20] E. Lear, "NERD: A Not-so-novel EID to RLOC Database," Internet Draft draft-lear-lisp-nerd-01.txt, June 2007.

[21] O. Bonaventure, C. de Launois, B. Quoitin, and M. Yannuzzi, "Improving the quality of interdomain paths by using IP tunnels and the DNS," Technical Report, January 2005.

[22] S. Brim, D. Farinacci, V. Fuller, D. Lewis, and D. Meyer, “LISP-CONS: A Content distribution Overlay Network Service for LISP,” Internet Draft draft-meyer-lisp-cons-00.txt, July 2007.

[23] Cisco, “Introduction to cisco ios netflow - a technical overview,” Available Online at: `http://www.cisco.com/en/US/products/ps6601/products_white_paper0900aecd80406232.shtml`.

[24] “Tcpdump/Libcap.” [Online]. Available: `http://www.tcpdump.org/`

[25] M. Fullmer, “flow-tools – tool set for working with netflow data.” [Online]. Available: `http://www.splintered. net/sw/flow-tools/docs/flow-tools.html`

[26] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, “iPlane: An Information Plane for Distributed Services,” *Proc. 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, November 2006. [Online]. Available: `http://iplane.cs.washington.edu/`

[27] S. Uhlig and O. Bonaventure, “On the cost of using MPLS for interdomain traffic,” *Proc. 1st COST263 workshop on Quality of future internet services*, September 2000.

[28] E. Kohler, J. li, and V. P. an S. Shenker, “Observed Structure of Addresses in IP Traffic,” *Proc. 2nd Internet Measurement Workshop (IMW)*, November 2002.

[29] S. Uhlig, V. Magnin, O. Bonaventure, C. Rapier, and L. Deri, “Implications of the Topological Properties of Internet Traffic on Traffic Engineering,” *Proc. 19th ACM Symposium on Applied Computing (SAC)*, March 2004.

[30] A. Soule, K. Salamatian, N. Taft, R. Emilion, and K. Papagiannaki, “Flow Classification by Histograms,” *ACM SIGMETRICS*, June 2004.

[31] K. Papagiannaki, N. Taft, and C. Diot, “Impact of Flow Dynamics on Traffic Engineering Design Principles,” *Proc. IEEE INFOCOM*, March 2004.

[32] C. de Launois and O. Bonaventure, “The NAROS Approach for IPv6 Multihoming with Traffic Engineering,” *Proc. 4th COST 263 International Workshop on Quality of Future Internet Services (QoFIS)*, October 2003.