A Local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers

Damien Saucez, Juhoon Kim, Luigi Iannone, Olivier Bonaventure, Clarence Filsfils

▶ **To cite this version:**

Damien Saucez, Juhoon Kim, Luigi Iannone, Olivier Bonaventure, Clarence Filsfils. A Local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers. 11th International Networking Conference (NETWORKING), May 2012, Prague, Czech Republic. pp.397-408, 10.1007/978-3-642-30045-5_30 . hal-01531133

HAL Id: hal-01531133
https://inria.hal.science/hal-01531133

Submitted on 1 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

A Local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers

D. Saucez[1], J. Kim[2], L. Iannone[2], O. Bonaventure[3], and C. Filsfils[4]

[1] INRIA Sophia Antipolis, France
damien.saucez@inria.fr
[2] Telekom Innovation Laboratories – Technische Universität Berlin, Germany
{jkim,luigi}@net.t-labs.tu-berlin.de
[3] Université catholique de Louvain, Belgium
olivier.bonaventure@uclouvain.be
[4] Cisco Systems
cfilsfil@cisco.com

Abstract. LISP (Locator/ID Separation Protocol) has been proposed as a future Internet architecture in order to solve the scalability issues the current architecture is facing. LISP tunnels packets between border routers, which are the locators of the non-globally routable identifiers associated to end-hosts. In this context, the encapsulating routers, which are called *Ingress Tunnel Routers* (ITR) and learn dynamically identifier-to-locators mappings needed for the encapsulation, can cause severe and long lasting traffic disruption upon failure. In this paper, thanks to real traffic traces, we first explore the impact of ITR failures on ongoing traffic. Our measurements confirm that the failure of an ITR can have severe impact on traffic. We then propose and evaluate an ITR synchronization mechanism to locally protect ITRs, achieving disruptionless traffic redirection. We finally explore how to minimize the number of ITRs to synchronize in large networks.

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

1 Introduction

It is widely recognized that the current Internet architecture is a victim of its own success and is facing unforeseen scalability issues, in terms of increasing routing tables, multi-homing, and inter-domain traffic engineering ([1], [2], [3]). Both academic researchers and industrial companies concur that the locator/ID separation paradigm, *i.e.*, separating the semantic of end-systems' identifier and location (currently merged in the IP address), will provide the needed scalability improvement. Among the several recently proposed protocols leveraging on this principle, the *Locator/ID Separation Protocol* (LISP [4]) has encountered the wider success and is being developed in the IETF (Internet Engineering Task Force [5]). LISP assigns Endpoint IDentifiers (EIDs) to end-hosts and Routing

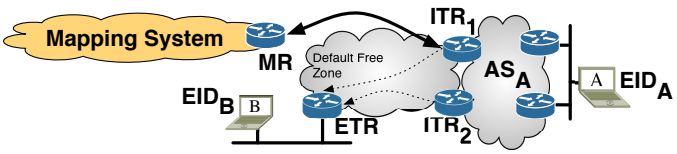


Fig. 1: LISP deployment example.

LOCators (RLOCs) to tunnel routers. In the core Internet only RLOCs are globally routable and used to tunnel packets between end-systems, which use EIDs that are globally unique but not globally routable (hence the use of tunnels).

Like the current Internet, LISP is affected by temporary link or node failures that may disrupt end-to-end reachability. More specifically, when an encapsulating router – the so called Ingress Tunnel Router (ITR) – fails, alternate/backup ITRs are not readily able to take over traffic encapsulation because they do not have the necessary identifier-to-locator mappings, resulting in prolonged packet drops and traffic disruption. Some aspects of the performance of LISP have been previously explored in other works, either focusing on the scalability of specific LISP elements, like the LISP cache ([6], [7], [8]) or on the mapping system used to distribute the mappings binding the locators with the identifiers ([9], [10], [11]). However, how LISP reacts to link and node failures has been largely neglected. The only document briefly discussing reachability issues is the draft by Meyer et al. [12], which focuses on the general implications of having an Internet architecture based on the locator/ID separation paradigm. To the best of our knowledge, the present paper is the first analyzing failures in LISP, and, from a more general perspective, the failure of encapsulating routers in the context of a locator/ID separated architecture.

In this paper, we first provide all the relevant information for a complete understanding of the problem, overviewing LISP in Sec. 2. In Sec. 3 we formalize the failure problem, followed by our proposal (Sec. 4) and its evaluation (Sec. 5). Then in Sec. 6 we tackle the recovery problem, showing how it can be solved as well with our approach. Sec. 7 and Sec. 8 explore the synchronization issue in large networks. Finally, Sec. 9 concludes the paper.

2 LISP Basics

LISP is typically implemented on customer-edge border routers, whose upstream IP address (*i.e.*, the one facing the Internet), which is a part of the Border Gateway Protocol (BGP) routing space, is the *Routing LOCator (RLOC)* used to locate end-systems of the local domain. The end-systems in the local domain use globally unique but not globally routable (hence not injected in the BGP routing infrastructure) IP addresses (or *End-point IDentifiers – EID)*. Then, as shown in Fig. 1, LISP tunnels packets in the core Internet from one RLOC of the source EID to one RLOC of the destination EID. The *Ingress Tunnel Router*

(ITR) prepends a LISP header to each packet using the ITR's RLOC as a source address and the *Egress Tunnel Router's (ETR's)* RLOC as a destination address, while the ETR strips this header and forwards the packet to its final destination.

The use of different addressing spaces, namely the EIDs and RLOCs, implies that there is the need of bindings between EIDs and RLOCs. These bindings specify which RLOC to use when encapsulating packets towards a given EID. These bindings are called *mappings*, hence the reason why LISP is called a *map-and-encap* approach. For scalability reasons mappings are obtained by ITRs on-demand and kept in a dedicated cache [8]. When, no suitable entry is present, a cache-miss is triggered, causing a query to a Map-Resolver (MR [13]), which is the entry point of the Mapping System. The query consists in a Map-Request message to which eventually the MR (or directly the destination ETR) will return a Map-Reply message with the requested mapping.[5] Mapping retrieval is a key point performance wise since a cache-miss causes signaling traffic, increases communication setup latency, and decreases the throughput. Such an impact is due to the fact that current implementations drop packets causing a cache-miss.[6]

In a normal IP network, such as the enterprise network shown in the right part of Fig. 1, the Interior Gateway Protocol (IGP) handles link and node failures. Consider for example that ITR_1 and ITR_2 advertise a default route in the IGP, a very common deployment. In this case, if (the link attached to) ITR_1 fails, then the other local routers will detect the failure and update their routing tables to forward the packets to ITR_2 so that they can reach the Internet through it. During the last years, various techniques have been developed to enable routers to quickly react upon such failures [14]. With LISP, the situation is different. If ITR_1 fails, the IGP will quickly redirect the packets to ITR_2. However, ITR_2 will be able to forward these packets only if it already knows the corresponding mappings. Otherwise, ITR_2 will have to drop packets while querying the mapping system, an operation that can take tens of milliseconds or more per mapping [11].

3 The ITR Failure Problem

In order to estimate how important the ITRs' failure problem is, the first step is to evaluate the level of redundancy present in enterprise and campus networks. Indeed, in today's Internet, these networks often contain several redundant border routers to preserve connectivity in case of failures.

To evaluate such an aspect, we analyze Internet topology information from the Archipelago project [15] and BGP tables of the Routeviews project [16]. We first extract the stub prefixes from the BGP table of Oregon-IX collected on August 12^{th} 2010 [16]. The BGP curve in Fig. 2 shows the number of neighbor ASes for each prefix. This is an approximation of the multi-homing degree. We

[5] The proposed solution does not rely on any mapping system, hence their description is out of scope. Jakab et al. [11] propose a comparison of different mapping systems.

[6] LISP does not specify any action for packets causing misses, which cannot be encapsulated due to the missing mappings. Current implementations, similarly to the case of missing MAC address in the ARP table, for scalability reasons drop the packet.

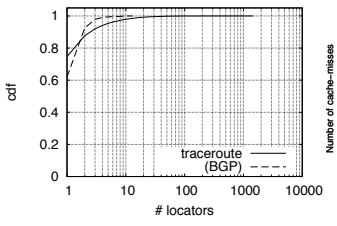
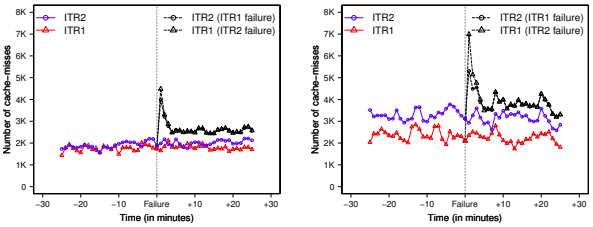


Fig. 2: Distribution of the number of border routers per BGP prefix.



(a) Campus (b) ISP

Fig. 3: Cache-misses per-minute due to ITRs failures.

obtain the traceroute curve in Fig. 2 by filtering an Archipelago trace captured July 24^{th} 2010 [15]. Archipelago traces are a collection of traceroutes performed from several vantage points to any possible `/24` prefix. We consider that the last router that does not belong to the stub BGP prefix is a border router for that prefix, providing an approximation of the number of active routers of the multi-homed stub prefixes.

The curves in Fig. 2 show that when a stub prefix is multi-homed, most of the time it has only two border routers. Nevertheless, the long tail of the distribution indicates that for some prefixes, the number of ITRs can be potentially high. For this reason, to assess to what extent ITRs' failures are a problem, we can analyze the impact on traffic for the simple scenario of 2 ITRs, by using trace-driven emulations and the topology presented in Fig. 1. Our emulations use the software and the methodology used in previous works ([6], [8]), assuming a mapping granularity equivalent to the current BGP table [8]. With this granularity, previous works ([6], [8]), have shown that cache size is pretty small, allowing to neglect cache overflow problems, which are unlikely to happen.

We use two different 24h-long traces collected in 2010. A first trace, collected in September 1^{st} with a NetFlow collector without sampling, is from a middle sized Campus network (∼9,000 active users), connected via a 1 Gbps link to its ISP and 122.35 Mbps average traffic. The second trace has been collected within a large European ISP (∼20,000 active DSL customers). We split the network of the captured traces into two subnetworks (served by ITR_1 and ITR_2) in order to implement the multi-homing scenario as in Fig. 1. Since ISP's topology was unknown, we attach half of the address space to ITR_1 and the other half to ITR_2. On the contrary, the Campus network has two border routers, which we assigned the role of ITR_1 and ITR_2, hence attaching the traffic to the ITRs according to the real topology. From the traces, we selected the busiest hour (*i.e.*, the worst case) and emulated the failure of one ITR in the middle of the selected hour.

Fig. 3 shows the impact of the failure on the number of cache-misses on the alternate ITR, for the ISP and the Campus networks, which have a similar behavior. The figure shows the normal behavior without any failure (solid lines), as well as the two cases where one of the ITRs fails (dashed lines). During the

first minute after the failure there are up to 5,000 additional cache-misses per minute (more than three times the normal rate) due to the traffic redirection on the alternate ITR. This failure can affect established TCP flows and cause packet drops for seconds or more. Fig. 3 shows as well that the transient state lasts around 5 minutes.

It has to be pointed out that we are underestimating the impact since we do not consider mapping delay, *i.e.*, we assume that the mapping is retrieved immediately after the cache-miss. In reality, failure-induced cache-misses have much more severe impact because they affect established high traffic volume flows. While a normal cache-miss causes the loss of a few packets at flow setup time [17], a failure-induced cache-miss could cause more losses because disrupting high throughput flows on high bandwidth links. Moreover, the peak of cache-miss causes a load peak on the mapping system to retrieve the missing mappings.

4 Local ITR Failure Protection

To solve the problem presented in the previous section, we propose to synchronize the caches of a group of ITRs of the same site. The set of ITRs belonging to the same group is called the *Synchronization Set*. Replicating the LISP cache on the ITRs ensures that in case of failure and traffic re-routing, the packets of an existing flow will never be dropped because of a cache-miss, whatever the alternate ITR of the set is used.

There are two ways to synchronize ITR caches: either the mappings are *pushed* to the ITRs of the Synchronization Set, or the latter are *notified* that a new mapping has to be retrieved. We discuss both approaches in further details in Sec. 8. Here we just assume that the mapping requested by one ITR of a Synchronization Set is immediately replicated on all ITRs of the set. To ensure that mapping caches remain synchronized, the ITRs should keep it for the same amount of time. This implies that synchronized ITRs cannot anymore use a simple inactivity timeout [6] to purge unused entries from their cache. Indeed, doing so would lead to a loss of synchronization, since the same mapping can expire on one ITR, because it has not been used, while it is kept on the other ITR that forwarded packets towards this EID prefix.

To avoid such loss of synchronization, we propose to use a different policy, namely to keep each mapping in the cache during the TTL (Time-To-Live) that LISP associates to this mapping. When the mapping TTL expires the ITR must check the entry usage during the last minute. If the entry has not been used, it is purged. Otherwise, the mapping entry is renewed by sending a Map-Request. This last action triggers the synchronization mechanism, again replicating the mapping to all ITRs of the Synchronization Set. Such approach guarantees that, if no ITR has used the mapping during the last minute before TTL expiration, all replicas on the different ITRs will be purged. Otherwise, if at least one ITR used the mapping, the mapping will be replicated on all ITRs, renewing it for another TTL time. In both cases there is a consistent state on all ITRs of the Synchronization Set.

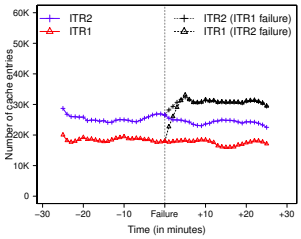


Fig. 4: Cache size without synchronization (ISP).

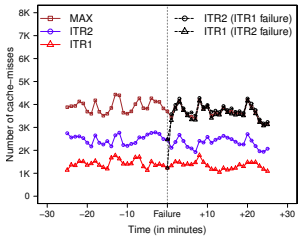


Fig. 5: Cache-misses with synchronization (ISP).

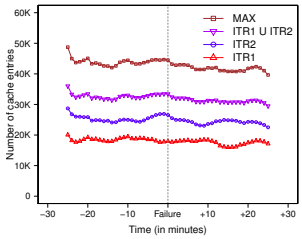


Fig. 6: Cache size with synchronization (ISP).

5 ITR Failure Protection Evaluation

To evaluate the cache synchronization technique we follow the same methodology and the same traces used in Sec. 3. On the one hand, we emulate the asynchronous cache strategy, *i.e.*, where each ITR manages its own cache independently. On the other hand, we emulate the synchronous cache strategy, *i.e.*, where ITR_1 and ITR_2 belong to the same Synchronization Set. In all of our simulations, we set the TTL to 5 minutes, which we consider as a reasonable worst case. Indeed, a lower value would generate too much overhead [6]. In practice, it is likely that the TTL will be longer than 5 minutes (the default in current LISP implementations being 24h), reducing the number of cache-misses, but increasing the number of entries that are stored in the cache [8]. However, it is worth noticing that the cost of slightly increasing the cache is very small compared to the cost of adding a redundant link. The figures in the remainder of this section show the cache behavior of ITRs in the normal case (*i.e.*, without failures) using solid lines, while dashed lines correspond to scenarios with failures.

In Fig. 3, the observed peak of cache-misses indicates that the traffic behind the two ITRs is not identical and not balanced; hence some mappings are in one ITR but not in the other. This is confirmed by Fig. 4, showing the evolution of the cache size for the ISP network (Campus network showing the same behavior), reinforcing the motivation for cache synchronization. Indeed, it can be observed that the ITRs have in general different cache sizes and when one of the ITRs fails the diverted traffic makes the size of the remaining ITR to grow.

Fig. 5 and Fig. 6 show the evolution of both cache size and cache-misses when our synchronization approach is used. The curves labeled `ITR1` and `ITR2` show the evolution on the ITRs when they are not synchronized and none fails. The curve labeled `MAX` shows the maximum obtained when ITRs are synchronized but the content of their cache is assumed to be completely disjoint (*i.e.*, the worst case). Finally, the curve labeled `ITR1 (ITR2 failure)` (resp. `ITR2 (ITR1 failure)`) corresponds to the actual values obtained with our synchronization approach if ITR_2 (resp. ITR_1) fails. Fig. 5 is interesting for two reasons. First, no peak is observed when the ITR fails, rather, the miss rate corresponds to the miss rate

that would have been observed if the network only had one ITR. As one could expect, the miss rate measured in steady state after the failure is identical to the miss rate observed in Fig. 3 once the steady state has been reached. Second, comparing Fig. 3 and Fig. 5, it turns out that the miss rate when no failure occurs is lower when the ITRs are synchronized than when they are not. This difference can be explained by the fact that some form of locality occurs between the traffic of the two ITRs.

In summary, our emulations on the 2 ITRs scenario clearly show that ITRs' cache synchronization brings two main advantages: (*i*) it avoids a miss storm (hence induced packet drops) upon ITR failure; (*ii*) reduces the number of misses (hence packet drops) in the normal case. These benefits come at a small cost of increased cache size. Indeed, Fig. 6 confirms that the ITRs have naturally some entries in common, which makes the burden of synchronization acceptable.

6 ITR Recovery: Problem, Protection, and Evaluation

The cache-miss storm in case of failure, as investigated so far, can also be observed when an ITR boots (or comes back online after failure). Indeed, in this case, its cache is empty, hence the traffic attracted for encapsulation will cause misses and will be dropped. Our synchronization mechanism can be used also in this scenario. In case of synchronization, the time needed for an ITR to be synchronized with the other ITRs of the Synchronization Set is at most equal to the TTL. In this situation, when the ITR starts, it could begin the synchronization process and wait TTL time before announcing itself as an ITR able to encapsulate packets. With this approach the miss rate is not different than if the ITR had always encapsulated traffic.

This naive approach is very simple but has a major drawback. The TTL can be set to a high value, refraining the ITR from being used for a long period of time. To overcome this issue, we suggest allowing the ITR to receive a copy of the cache from another ITRs in the Synchronization Set. The transfer of the cache's information must be done reliably, *e.g.*, using TCP. In this way the ITR can announce itself right after the cache transfer, shortening the start up time to a minimum, while preserving the benefits of the synchronization approach.

To better understand the impact on the traffic in the recovery case, we perform emulations in the two ITRs scenario, with ITR_2 recovering after ITR_1 runs alone for the last 30 minutes (using the ISP trace). Once back online, ITR_2 starts synchronizing with ITR_1, *i.e.*, it receives mapping information, but the traffic is still all routed toward ITR_1. After 5 minutes (*i.e.*, the TTL value we are using throughout all emulations), when the cache is synchronized with ITR_1, the original setup is restored, sending traffic again to ITR_2.

Fig. 7 shows the miss rate observed at the ITRs for both the synchronized and the non-synchronized cases. The curve `ITR1 (ITR2 failure)` gives the miss rate observed on ITR_1 before the recovery of ITR_2, while `ITR1 (sync)` (resp. `ITR2 (sync)`) shows the miss rate as observed on ITR_1 (resp. ITR_2) after ITR_2 has recovered when synchronization is used. For comparison, the curve `ITR1`

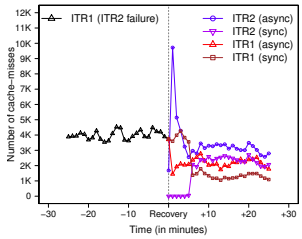


Fig. 7: Cache-miss on ITR_2 after boot.

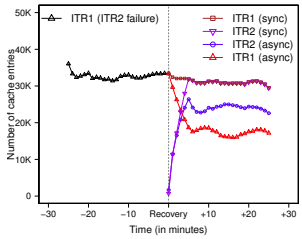


Fig. 8: Cache size on ITR_2 after boot.

	Failure	Increase	Increase*	Network
Entries (avg.)	ITR1	55.54%	46.59%	Campus
	ITR1	20.57%	20.30%	ISP
	ITR2	68.54%	56.98%	Campus
	ITR2	63.93%	56.54%	ISP
Misses (avg.)	ITR1	40.32%	55.25%	Campus
	ITR1	17.02%	28.22%	ISP
	ITR2	52.46%	72.95%	Campus
	ITR2	67.14%	96.34%	ISP

* Increase without counting the 5 min. transient period right after failure

Table 1: Increase of cache size and misses due to ITR failure.

`(async)` (resp. `ITR2 (async)`) shows the miss rate when no synchronization is used, with a peak of 10,000 cache-misses confirming that waiting TTL instants to lazily synchronize the cache avoids cache-miss storm. For completeness, Fig. 8 shows the evolution of the cache size before and after recovery. In all cases, the cache smoothly moves towards its new steady state TTL time after the recovery. This smooth convergence motivates the use a fast cache synchronization method (*i.e.*, explicit cache copy request) to speed-up the recovering of an ITR.

7 Synchronization Set in Large-Scale Networks

As shown in Sec. 3, in case of failure, when no synchronization is used, both the cache size and the cache-misses increase, as summarized in Tab. 1. While relatively similar increases can be observed comparing the results of the Campus and the ISP traces, the difference of the increases between ITR_1 and ITR_2 may be significant. Indeed, the average miss rate can be as low as 28% when ITR_1 fails, but as high as 96%, for the same trace, when ITR_2 fails. This result suggests that the Synchronization Set is a key point and should be carefully computed. The issue is exacerbated in large networks containing potentially many ITRs, which cannot all be synchronized.

For the above-mentioned reasons, it is important to have an idea of how large the Synchronization Set can be. To this end, we exhaustively calculated the Synchronization Set of 8 different real large-scale network topologies. Each of these topologies belong to one of the three following categories: (i) *TIER1* grouping

Name	Routers	Links
TIER1a	500^+	$2{,}000^+$
TIER1b	200^+	800^+
ISPa	50^+	200^+
ISPb	20^+	60^+
UCL	11	41
NREN	30^+	70^+
Internet2	11	30
Géant	22	72

Table 2: Topologies characteristics.

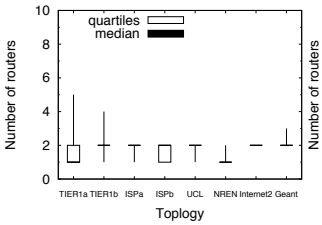


Fig. 9: Number of used ITRs per EID routers.

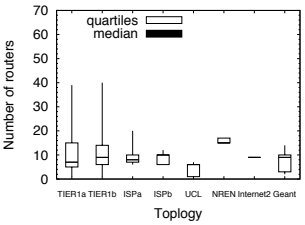


Fig. 10: Number of routers behind an ITR.

Tier 1 ISPs; (ii) *ISP* grouping national ISPs; (iii) *Research* grouping university campuses or research networks. A summary of these topologies is shown in Tab. 2. We model the networks as a directed acyclic graph where the nodes are the routers and the edges are the links. Then we differentiate the routers in three different types: the *ITRs*, the *EID routers*, and the *backbone routers*. For the topologies where we had enough information, we considered the routers connected to another ISP as ITRs; the routers connected to the customers or to LANs with end-hosts are considered the EID routers; all the other routers are classified as backbone routers. For the topologies where no enough information was available, we classified the routers based on their connectivity degree. All the routers with a connectivity degree higher or equal to the 90^{th} percentile of the connectivity degree are considered ITRs. The routers with a connectivity degree lower or equal to the 80^{th} percentile of the connectivity degree are EID routers. All the other routers are classified as backbone routers. Applying this heuristic provides results similar to the topologies where information was available. To construct the Synchronization Set, we exhaustively enumerate the topology changes for any possible single failure (*i.e.*, router or link).

Fig. 9 gives the quartiles (including min., max., and median) computed for the number of ITRs potentially used by each EID router. For this evaluation, we consider that only one EID router serves an EID prefix. This is a reasonable assumption since for the transit networks we evaluated (*i.e.*, TIER1a, TIER1b, NREN, Internet2, and Géant) the EID prefixes belong to the customers, which have only one router toward their ISP (but are multi-homed with several ISPs). The figure shows that the number of ITRs potentially used by the EID routers is relatively independent of the type of topology and is between 1 and 2. Meaning that only a small portion of the ITRs are serving the same EIDs, due to the fact that large networks are segmented in relatively independent points of presence. This result clearly shows that synchronizing all the ITRs of a large network would be inefficient, while in our case the burden will be reasonable as an ITR will synchronize only with few other ITRs.

Fig. 10 goes the other way around, showing the number of routers that are potentially behind a single ITR, using again quartiles. The figure shows that in

general an ITR has about half a dozen of routers behind it, going up to few tens for large networks. Therefore, the failure of an ITR can potentially impact on an important portion of traffic, which has to be shifted to the other ITRs. If the caches are not synchronized, the ITRs to which the traffic is diverted might experience a very important miss rate, with high packet drops. Synchronizing the ITRs' caches avoids such drops, as we demonstrate in Sec. 5.

In summary, protecting the ITRs is important as the failure of one of them can impact a large portion of the network. Fortunately, topologies of real networks seem to be segmented so to facilitate grouping the ITRs in small Synchronization Sets.

8 Synchronization Techniques

Even if, as discussed above, the Synchronization Set is in practice small also for large-scale networks, it is important to implement a mechanism to achieve such synchronization without introducing excessive overhead or management complexity. There are two possible ways to synchronize ITRs: either, the mappings are *pushed* to the ITRs, or the ITRs are *notified* of the presence of a new mapping and they retrieve the mapping on their own.

In both cases, the synchronization can be implemented either by leveraging on a routing protocol extension, by using a specific existing protocol (*e.g.*, [18], [19], [20], [21]), or by creating a brand new protocol. Extending a routing protocol to synchronize ITRs implies that they must be in the same routing protocol instance (*e.g.*, the same IGP). However, this assumption is too restrictive as it might exist cases where the ITRs are in different networks (*e.g.*, the ITRs are operated by the ISPs and the LISP site is multi-homed). Hence, we do not consider the extension of the routing protocol as an acceptable solution. Fortunately, LISP already proposes features that are handy to implement a synchronization mechanism. Indeed, LISP specifies the *Included Map-Reply* [4] feature to push mappings in ITR caches. An included Map-Reply is a special Map-Request, which piggybacks a mapping. LISP also specifies the *Solicit Map-Request* (SMR) bit in Map-Requests to force ITRs to refresh their cache [4]. When an ITR receives a Map-Request with the SMR bit set, it sends a Map-Request to retrieve a mapping for the EID indicated in the SMRed Map-Request.

Since both ITRs and MRs are involved in the mapping resolution, these are good candidates to trigger cache synchronization. However, ITRs are data-plane devices that need to forward packets at line rate. Therefore, imposing them to actively manage the synchronization protocol might cause excessive overhead with consequences on the data-plane performance. On the contrary, Map Resolvers are purely control-plane devices that are not intended to forward packet at line rate. Hence, MRs look like the best candidates to manage the synchronization protocol. To implement the cache synchronization based on notification messages, the MR only has to send an SMRed Map-Request to all the ITRs, when it receives a Map-Request. However, if the mappings are pushed to the caches, then the MR has to proxy the Map-Requests. The MR performs two

operation when it receives a Map-Reply. First, it forwards the Map-Reply to the ITR that requested the mapping. Second, it sends the mapping to the other ITRs by using an included Map-Reply.

As discussed above, the Synchronization Set is in practice small and can be statically configured in the MR. However, for large or very dynamic networks, the configuration burden might still become cumbersome and a dynamic ITR discovery protocol coupled with an automatic Synchronization Set computation algorithm should be considered. It is out of the scope of this paper to propose any specific solution for these two mechanisms. Furthermore, there is still the open question of what is the best trade-off between tight synchronization and signaling overhead. Depending on the importance accorded by the operator to the accuracy of the cache synchronization, the mapping distribution between to the ITRs can be performed with a reliable protocol (*i.e.*, TCP) or not. In addition, batching of synchronization messages can be used to reduce the number of exchanged synchronization messages. Moreover, when the network allows it, the mappings can be distributed by using IP multicast.

9 Conclusion and Further Work

In this paper, we propose a thorough study of failure protection and recovery in the context of the Locator/Identifier Separation Protocol (LISP) and propose a local failure protection mechanism for Ingress Tunnel Routers (ITRs). We first showed that ITR failures can indeed have large disruptive impact on ongoing traffic. Then, we explored how to minimize packet losses due to cache-misses caused by the redirected traffic on the alternate ITRs. Our proposal synchronizes the cache of set of ITRs in order to avoid such a phenomena. We thoroughly evaluate our proposal, showing that the load increase due to the synchronization is acceptable and suppresses the loss of packets upon ITRs failure/recovery.

Our ongoing work is aiming at extending the synchronization mechanism to ETRs and developing detailed specifications for implementation and experimentation in the *lisp4.net* testbed.

Acknowledgements. This work was partially supported by a Cisco URP grant. The research leading to these results has received funding from the French National Research Agency under grant agreement n° ANR-11-EITS-007-01. We would like to thank Simon van der Linden for his help.

References

1. BGP Routing Table Analysis Report. [Online]. Available: http://bgp.potaroo.net
2. B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure, "Evaluating the benefits of the locator/identifier separation," in *the 2nd ACM/IEEE Workshop on Mobility in the evolving internet architecture (MobiArch'07)*, August 2007.
3. D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing," RFC 4984, Internet Engineering Task Force, September 2007.

4. D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID separation protocol (LISP)," IETF, Internet Draft *draft-ietf-lisp-22.txt*, February 2012.
5. Locator/ID Separation Protocol (LISP) Working Goup. [Online]. Available: http://datatracker.ietf.org/wg/lisp/charter/
6. J. Kim, L. Iannone, and A. Feldmann, "A Deep Dive into the LISP Cache and What ISPs Should Know about It," in *the 10^{th} IFIP Internation Conference on Networking (Networking'11)*, May 2011.
7. Z. H, M. Chen, and Y. Zhu, "Evaluating the Performance on ID/Loc Mapping," in *the Global Communications Conference (Globecom'08)*, November 2008.
8. L. Iannone and O. Bonaventure, "On the cost of caching locator/id mappings," in *the 3^{rd} ACM Annual CoNEXT Conference (CoNEXT'07)*, December 2007.
9. K. Sriram, P. Gleichmann, Y.-T. Kim, and D. Montgomery, "Enhanced Efficiency of Mapping Distribution Protocols in Scalable Routing and Addressing Architectures," August 2010.
10. N. Choi, T. You, J. Park, T. Kwon, and Y. Choi, "ID/LOC Separation Network Architecture for Mobility Support in Future Internet," February 2009.
11. L. Jakab, A. Cabellos-Aparicio, F. Coras, D. Saucez, and O. Bonaventure, "LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System," *IEEE Journal on Selected Areas in Communications*, October 2010.
12. D. Meyer and D. Lewis, "Architectural Implications of Locator/ID Separation," IETF, Internet Draft draft-meyer-loc-id-implications-01, January 2009.
13. V. Fuller and D. Farinacci, "LISP Map Server," IETF, Internet Draft *draft-ietf-lisp-ms-15.txt*, January 2012.
14. J. Vasseur, P. Demeester, and M. Pickavet, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*, Elsevier, Ed., 2004.
15. K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: From art to science," in *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 205–211.
16. University of Oregon, "Route views, University of Oregon Route Views project," see http://www.routeviews.org/.
17. M. Ohmori, K. Okamura, and F. Tanizaki, "Analyses on first packet drops of lisp in end-to-end bidirectional communications," in *Internet conference*, Fukuoka, Japan, October 2011.
18. J. Luciani, G. Armitage, J. Halpern, and N. Doraswamy, "Server Cache Synchronization Protocol (SCSP)," RFC 2334 (Proposed Standard), Internet Engineering Task Force, Apr. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2334.txt
19. Y. Saito and M. Shapiro, "Optimistic replication," *ACM Comput. Surv.*, vol. 37, pp. 42–81, March 2005.
20. K. Obraczka and P. Dazing, "Evaluating the performance of flood-d: A tool for efficiently replicating internet information services," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 369–382, April 1998.
21. K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and A. Demers, "Flexible update propagation for weakly consistent replication," in *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP)*, October 1997, pp. 288–301.