Topology Discovery at the Router Level: A New Hybrid Tool Targeting ISP Networks

Pietro Marchetta, Pascal Mérindol, Benoit Donnet, Antonio Pescapé, Jean-Jacques Pansiot

Abstract—For a long time, traceroute measurements combined with alias resolution methods have been the sole way to collect Internet router level maps. Recently, a new approach has been introduced with the use of a multicast management tool, `mrinfo`, and a recursive probing scheme. In this paper, after analyzing advantages and drawbacks of probing approaches based on traceroute and `mrinfo`, we propose a hybrid discovery tool, MERLIN (MEasure the Router Level of the INternet), mixing `mrinfo` and traceroute probes. Using a central server controlling a set of distributed vantage points in order to increase the exploration coverage while limiting the probing redundancy, the purpose of MERLIN is to provide an accurate router level map inside a targeted Autonomous System (AS). MERLIN also takes advantage of alias resolution methods to reconnect scattered multicast components. To evaluate the performance of MERLIN, we report experimental results describing its efficiency in topology exploration and reconstruction of several ASes.

Index Terms—Network topology, Router level, IGMP probing, alias resolution, traceroute, MERLIN.

I. INTRODUCTION

INTERNET topology discovery [1], [2] has been an intensive research topic during the past decade, leading to a large set of tools for collecting data. CAIDA's *Archipelago* [3] uses 41 monitors tracing towards roughly 8.25 millions destinations. *iPlane* [4] constructs an annotated map of the Internet and evaluates end-to-end performance metrics (latency, bandwidth, capacity, etc). *DIMES* [5] is publicly released as a daemon running on end hosts tracing a set of targets obtained from a central server. *Rocketfuel* [6] uses roughly 800 vantage points to map targeted Internet Service Providers (ISP). Finally, the recently introduced *Scamper* is able to conduct parallel IPv4 and IPv6 measurements at a specified packets-per-second rate [7]. All of these efforts are based on one of the most common probing tool: *traceroute* [8].

The Internet topology may be seen at three different levels: the AS level, the IP interface level, and the router level. While the two first levels do not require to find which IP interfaces belong to a given router, router level topology discovery usually relies on *alias resolution* techniques [6], [9], [10], [11]. This process is known as quite intrusive and/or resource consuming whereas it suffers from several bias. Inferring the router level topology of IP networks is an important concern in particular to study routing characteristics. More specifically, inferring the design of an Autonomous System (AS) is crucial for analyzing intra-domain routing protocol performance. Network protocols designers should evaluate the performance of their proposals on realistic topologies in order to highlight their advantages and limitations. For example, performance of fast-rerouting schemes or multipath transport protocols may strongly depend on the underlying topology. Inferring AS at the router level may help them to develop efficient solutions able to perform well on various topology designs and common patterns.

Recently, there has been an increasing interest in multicast-based approaches to discover topologies at router level: if multicast routing is enabled, a single probe can collect much more information than traceroute without suffering from alias problems [12], [13], [14].

In this paper, we propose a novel client/server platform we called MERLIN (MEasure the Router Level of the INternet) efficiently mixing three active probing tools: IGMP probing using an improved version of `mrinfo` [13], ICMP probing with Paris Traceroute [15], and alias resolution with *Ally* [6]. MERLIN makes use of IGMP probes to natively discover ISPs at the router level, while Paris Traceroute and Ally are used to overcome `mrinfo` limitations. The methodology we propose here does not depend on those two last specific tools, our platform can be deployed with any other probing mechanisms.

To the best of our knowledge, our work extends the topology discovery literature on the following aspects:

- it provides a detailed comparison of IGMP-based and traceroute-like probing (Sec. III). Using the `mrinfo` (Sec. II) and the *Skitter* [16] data, we demonstrate that both tools are not antagonist but, rather, complement each other. Starting from this statement we propose the MERLIN architecture (Sec. IV);
- MERLIN relies on an improved version of `mrinfo-rec` (Sec. II) for large scale measurement campaigns;
- MERLIN uses in background Paris traceroute and Ally in order to improve its Internet cartography effort: this is the first attempt of combining these three tools (Sec. IV);
- MERLIN provides accurate router level maps with a limited probing overhead that does not rely on intensive probing campaigns: it is easy to setup and does not require a large set of vantage points (Sec. V);
- the set of collected ISP networks is publicly available through a high level XML description providing various features: an IP multigraph perspective, layer-2 inference, DNS resolution, and router geolocation (Sec. V).

II. `MRINFO`

`mrinfo` is a multicast tool that uses IGMP (*Internet Group Management Protocol*, [17]) messages to silently (i.e., with a single probe) discover all IPv4 multicast interfaces of a router. We first describe (Sec. II-A) the basics of IGMP probing, leading to the standard `mrinfo` tool. Then (Sec. II-B), we explain how we recursively exploit `mrinfo`, leading to `mrinfo-rec` probing tool.

A. IGMP Probing

In the late 1980s, the developers of IP multicast designed the MBone, an overlay network composed of tunnels that interconnected workstations running an implementation of DVMRP (Distance Vector Multicast Routing Protocol) [18]. Several tools have been developed to monitor and debug the MBone [19]. Most of these tools have been deprecated with the replacement of DVMRP by the PIM (Protocol Independent Multicast) family of multicast routing protocols with one notable exception: `mrinfo` [20]. `mrinfo` uses IGMP messages. IGMP was designed to allow hosts to report their active multicast groups to a multicast router on their LAN. Most IGMP messages are sent with a Time-to-Live of 1. However, DVMRP defined two special types of IGMP messages that can be used to monitor routers [18]. Although current IPv4 multicast routers no longer use DVMRP, most of them still support those special IGMP messages. Upon reception of an IGMP `ASK_NEIGHBORS` message, an IPv4 multicast router replies with an IGMP `NEIGHBORS_REPLY` message that lists all its multicast adjacencies with several information about their states. Due to space constraints we cannot report more details on `mrinfo` and its main features allowing to discover IP network topologies. Interested readers can find further details on `mrinfo` in [12], [13], [14].

B. Recursive IGMP Probing

Up to now, `mrinfo` measurements were conducted recursively with the `mrinfo-rec` tool [13], [14]. It starts by probing a given target using `mrinfo` and then recursively invokes `mrinfo` on all neighbor IP addresses discovered in responses. This approach is designed to discover and study the largest multicast component reachable from a single starting target address, the *seed*, and from a single vantage point.

In the past, `mrinfo-rec` campaigns were conducted daily in order to understand the Internet dynamics. To maximize discovered topology, we used the set of responding routers of a given day as seeds for the next day's recursive campaign[1]. This seeding procedure allowed us to take advantage of any changes in the routing system to discover new areas of the multicast-enabled Internet. Between May 1st, 2004 and December 31st, 2008, `mrinfo-rec` was able to discover 10,000 routers on average from a single vantage point in Strasbourg, France. We observed two notable and sudden changes in data collection over this period. First, during the second half of 2005, a forwarding change or the removal of filtering allowed `mrinfo-rec` to discover a larger portion of the

[1]It is worth noting that, in the vast majority of cases, a single day was enough to collect the entire resulting topological information.

TABLE I
ASes TARGETED AND GLOBAL STATISTICS

Level	Name	ASN	IP addresses collected			
			Skitter (T)		`mrinfo-rec` (M)	
Tier-1	Global Crossing	3549	2,464	±149	903	±223
	Level3	3356	5,482	±64	1,898	±84
	Sprint	1239	9,308	±99	3,120	±156
	NTTC	2914	3,622	±61	1,052	±73
Transit	TDC	3292	5,192	±98	2,932	±113
	IUNet	1267	1,429	±99	2,630	±80
	DFN	680	4,398	±87	1,600	±66
Stub	GARR	137	1,929	±111	808	±111
	Cisco System EU	109	83	±3	129	±2
	Uninett	224	972	±20	1,531	±21

multicast map. Second, at the beginning of 2007, the addition of filters significantly reduced the number of reachable routers. Such sudden and significant changes are not due to network dynamics: they are an artifact of `mrinfo-rec` launched from a single vantage point, making data collection susceptible to IGMP filtering (more collaborating vantage points help to overcome such situations, see Sec. III-D and [21]).

III. COMPARISON BETWEEN TRACEROUTE AND `MRINFO`

To underline both the differences and the complementarities between traceroute and `mrinfo`, we provide a quantitative comparison between their network coverage. More specifically, we analyze the probing coverage of `mrinfo-rec` and Skitter [16]. It is worth to notice that the objective of this section is not to determine whether `mrinfo-rec` is better than Skitter (or the contrary). Rather, we want to point out the limitations of both tools and understand whether they could be combined to increase our vision of the topology. Moreover, our comparison only focuses at the IP level, our goal is not to draw precise AS boundaries.

Sec. III-A presents our methodology while in Sec. III-B and in Sec. III-C we discuss our main results. Finally, Sec. III-D briefly discusses the limitation of `mrinfo`.

A. Methodology: an IP-based Comparison

As already mentioned in Sec. II-B, the best period of `mrinfo-rec` in terms of amount of collected routers and IP interfaces was during 2006. In this section, we thus consider data collected by `mrinfo-rec` during 2006. We preprocess this dataset using the *router-to-AS mapping* [13]. This algorithm assigns a unique AS number to each router and identifies AS border routers (ASBRs). The algorithm is based on a set of rules using common IP address allocation guidelines and probabilistic methods. To reinforce the mapping, it verifies, at each step, the consistency of the assignment returned by each rule. Furthermore, to perform the IP level comparison, we also provide the IP-to-AS mapping of each IP interface.

In 2006, CAIDA's traceroute probing campaigns were based on *Skitter* [16], a measurement tool using 19 monitors spread all over the world, each targeting on the order of 16 million destinations. Note that, at that time, Skitter did not yet implement Paris Traceroute [15] and, thus, may suffer from load balancing anomalies coming with the standard traceroute tool.

In order to compare the two 2006 datasets, (*i*) we select one `mrinfo-rec` topology snapshot per month considering
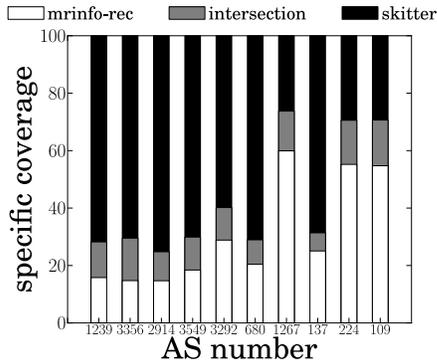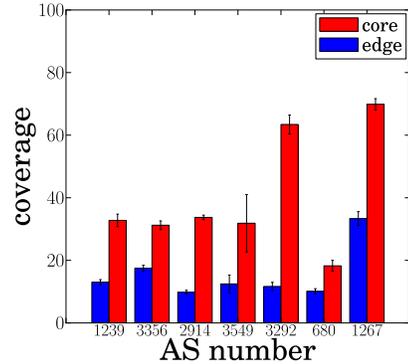
Fig. 1.   Specific coverage

Fig. 2.   `mrinfo-rec` coverage

the day where it produced the largest coverage in terms of collected routers; (ii) then, we compare it to the union of datasets collected by the 18 Skitter monitors (we removed the "arin" monitor from the dataset due to its too slow probing process) for the probing cycle that started the same day[2]; (iii) from both raw datasets, we extract only all globally routable IP addresses. Non-publicly routable addresses (i.e., 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 0.0.0.0/8, 198.18.0.0/15, and 127.0.0.0/8 prefix blocks) were filtered from our datasets (such a filter removes, on average, 1.5% of collected IP addresses); (iv) remaining globally *routable* IP addresses are then mapped to the AS they belong to thanks to an IP-to-AS process based on the BGP prefixes announces received by the RouteViews project. In case of Multiple Origin Autonomous System (i.e., when an IP address is mapped to more than one AS), we decide to map the address to the first *valid* AS.

For the comparison between traceroute and `mrinfo-rec`, we considered a set of 12 ASes. Table I shows the subset of ASes of interest used in the remainder of this section. Due to space constraints we cannot present the results of the whole set of ASes we analyze. Interested readers can refer to [22] for further details: in practice, we select multicast enabled ASes that `mrinfo-rec` is able to capture. This selection has been made so that it remains fair and representative across different levels in the AS graph hierarchy. Note that results provided in Table I are averaged over the twelve months in 2006. In addition to the mean value, we determine 95% confidence intervals (in small font, after the $\pm$) for the mean based on the Student t distribution.

B. Global Analysis

This section shows the relative performance of both tools on a per AS basis. We use the term *specific coverage* to refer to the proportion of IP addresses that were discovered *only* by one tool. Thus, if T and M respectively denote the set of IP addresses collected with traceroute and `mrinfo-rec`, then the specific coverage of `mrinfo-rec` is $\frac{|M \setminus T|}{|M \bigcup T|}$ and the one of Skitter is $\frac{|T \setminus M|}{|M \bigcup T|}$. These relative proportions are subject to the IP-to-AS mapping applied on a given AS. The average

cardinality (with the confidence interval over the mean) of sets T and M are provided in Table I.

Fig. 1 reports the results of our comparison for the selected ASes shown in Table I. Considering this representative set of ASes, the specific coverage of `mrinfo-rec` is, on average, around 37%: between 14% for Level3 and 60% for IUNet. Surprisingly, the intersection set is quite low: only 11% on average. Finally, Skitter provides a better specific coverage than `mrinfo-rec`: 52% on average.

This first set of results can be interpreted as follows: while traceroute campaigns are inherently designed to discover most used links in term of forwarding representativity (such as Tier-1 backbone links), `mrinfo-rec` has the potential to discover an entire Stub AS if this one enables multicast. Indeed, we can notice that the coverage of Tier-1 AS is in favor of Skitter while the coverage of `mrinfo-rec` progressively increases for ASes lower in the AS graph hierarchy (Stub AS or Transit AS close to the leaf level). Furthermore, the systematic low intersection between both tools (whatever the AS level) highlights the complementarity of those two probing methodologies: while `mrinfo-rec` is limited to a multicast scope and may suffer from IGMP filtering[3], traceroute, being forwarding dependent, is not able to discover backup links and some network parts may be hidden to traceroute due to MPLS or other tunneling techniques.[4]

Typically, if a network enables multicast capabilities, at least the backbone area of the routing domain should be almost fully multicast compliant. Finally, it is worth to notice that Skitter obtains better results thanks to a much more huge cost in terms of injected traffic (*number of sent probing packets*), discovery time (*period of probing*) and hardware resources (*number of vantage points*) than `mrinfo-rec`.

C. Edge vs. Core Analysis

In order to investigate which part of the network both techniques are able to discover, we refine our comparison methodology by classifying addresses collected by Skitter into two categories: *edge* and *core*. Usually, a traceroute sequence

---

[2]Note that a typical Skitter's cycle lasts roughly three days so that we compare the amount of IPs collected during one day with `mrinfo-rec` to the data collected in three days using Skitter.

[3]As it may be also the case for ICMP probes for different parts of the network.

[4]It is worth to notice that Skitter is not designed to reveal an entire AS topology but rather to provide a subset of the Internet IP level graph (generally used to draw the AS level graph). This is particularly true since Skitter targets a single IP address per routable /24.
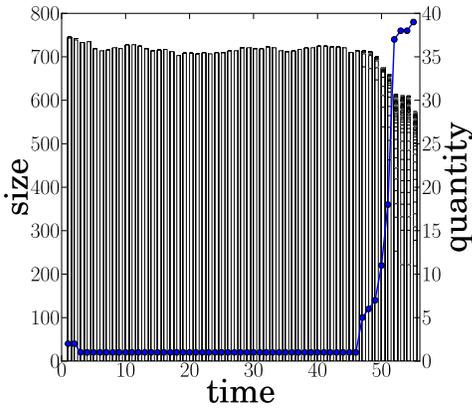
Fig. 3. Connected components evolution over time (AS1239)

contains several consecutive IP addresses mapped to the same AS. IP addresses tagged as edge are those at the extremity of a sub-trace traversing the AS of interest. Core IP addresses do not necessarily refer to the routing backbone area of the ISP: they simply depict non AS border links. Furthermore, an IP address is classified in the core category only if it never appears as an edge IP address in another trace: if the two sets intersect, an address belonging to the intersection is considered as an edge IP address. Using this simple classification, we can highlight the difference of coverage of `mrinfo-rec` related to both categories of Skitter IP addresses. In this structural analysis, the notion of coverage is related to the intersection between `mrinfo-rec` and the Skitter IP addresses set T labeled as edge, T_e, or core, T_c, such that it verifies either $\frac{|M \cap T_e|}{|T_e|}$ or $\frac{|M \cap T_c|}{|T_c|}$.

Fig. 2 shows the results of this analysis (i.e., the `mrinfo-rec` coverage values are reported for both edge and core classes). The `mrinfo-rec` coverage significantly increases if we consider the core of the AS: this interesting result greatly varies among our set of selected ASes (between 50% and 400% of relative improvement) but it always reveals the ability of `mrinfo-rec` to better depict the core of a given AS. On average, the coverage proportion grows from 17% to 39% from the edge to the core class: `mrinfo-rec` is able to compete with Skitter for addresses belonging to the core routing area while inter-domain peering connections that do not implement multicast adjacencies are not retrievable through `mrinfo-rec`. This drawback is due to the fact that even if `mrinfo-rec` may discover a border router (if it is multicast), it will not return non multicast adjacencies[5].

D. `mrinfo-rec` limitations

The main drawback of `mrinfo-rec`, also explaining the superiority of Skitter coverage, is the fact that it is only able to discover the multicast and `mrinfo` compliant parts of a network. Furthermore, IGMP messages – as ICMP ones for traceroute – can be filtered or rate limited by border routers. The recursive probing approach of `mrinfo-rec` is affected by two limitations: the data collection may be stopped by

[5]Note that a Skitter IP labeled as an *edge* IP is likely to be outside the AS when the AS is a high-level ISP such as a Tier-1. Interest reader might refer to the provider-to-customer IP allocation rule described in [13].

non-responding routers and by filtering routers. While the second problem can be at least partially circumvented by using several vantage points, the first one limits the efficiency of the recursive scheme. The consequence of this limitation is the partitioning of the multicast enabled topology of a given AS. Fig. 3 illustrates this problem on the Sprint Tier-1 AS. While this AS is seen as a whole connected graph during the first three years of probing, the situation starts to deteriorate at the beginning of 2007. The explanation of this degradation is the progressive introduction of non `mrinfo` compliant multicast routers in the network. Indeed, the number of connected components increases with the number of non-responding routers. Keeping in mind that `mrinfo-rec` uses the set of responding routers of the previous day as a list of seeds for the next probing period, it is able to collect non connected components while a pure recursive scheme starting from scratch cannot.

Using DNS names of IP interfaces that were connected to routers "disappearing" from the dataset, we notice that this problem corresponds to the deployment of new Cisco CRS routers that are non `mrinfo` compliant in the AS. This *unresponsive router effect* affects several ASes in the `mrinfo-rec` dataset and implies the need for alias resolution techniques and new way of seeding for launching new `mrinfo` campaigns. Indeed, the interdomain multicast graph cannot be seen anymore as a single connected component via `mrinfo-rec`. It is necessary to reconnect disconnected components in order to provide a consistent topology and develop new seeding techniques to overcome the limitations of the recursion. MERLIN (see Sec. IV) comes with several improvements to compensate this degradation.

A first conclusion of this section is the fact that `mrinfo` and traceroute are complementary tools to probe IP networks. In this paper we focus on ways to use traceroute to feed the `mrinfo` probing process. Compared to `mrinfo-rec`, the two main challenges of MERLIN are related to the:

(i) use of traceroute to discover active addresses in the targeted AS in order to circumvent the recursion limitations;
(ii) use of alias resolution techniques and traceroute in order to reconnect disjoint multicast regions to provide connected ISP maps.

IV. MERLIN

The objective of MERLIN (MEasure the Router Level of the INternet) is to collect an accurate router-level topology snapshot of a targeted AS by combining IGMP probing, traceroute, and an alias resolution technique. More precisely, the first challenge of MERLIN is to use traceroute seeds in order to improve the probing coverage, by overcoming the limitations of a pure IGMP recursion scheme. The action of the server aims to limit the traffic injection through a strong centralized coordination of the monitors. Then, the use of an alias resolution technique and traceroute allows us to provide consistent ISP maps by reconstructing connected topologies. In Sec. IV-A, we provide a global overview of the MERLIN architecture, while Sec. IV-B and Sec. IV-C respectively focus on the client (i.e., monitor) and the server sides. A complete description of the architecture and a detailed analysis of the collected data are reported in [21] and [22].
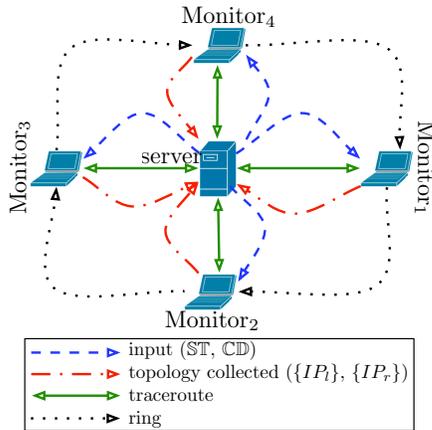
Fig. 4. MERLIN global overview

A. General Overview

MERLIN is based on a centralized client-server architecture and runs over Unix platforms. A central server, written in Python and C++, controls the vantage points, called *monitors*, written in C. The MERLIN monitors are potentially spread all over the world and logically organized as a ring, as illustrated in Fig. 4. Only one monitor actively probes the targeted domain at a given time. This probing period by a single monitor is called *run*. There are as many runs as monitors in the system per probing *cycle*. A *cycle* is completed when all monitors have probed the targeted domain during a given round around the monitor ring. Note that, for a given targeted AS, several cycles can be required per IGMP probing *stage*, i.e., the number of cycles required to take advantage of all current seeds including those recursively found during previous runs. An IGMP probing stage might be made of an incomplete number of cycles: a stage stops when no more topological data can be collected using the current seed set. In order to increase the coverage of an AS probing *campaign*, between each stage, MERLIN launches internal Paris traceroute campaigns to collect new fresh seeds and better understand the topology lacks. Thus, for a given AS, a complete MERLIN campaign consists in several stages of IGMP probing (made of several cycles) seeded by external measurements and internal traceroute campaigns.

The input sent by the server to each monitor is composed of a list of IP addresses called *CurrentSeed* ($\mathbb{CD}$ in Fig. 4 - see Sec. IV-C for its exact description). This set may contain seeds coming from several kinds of input. At the first stage, the seed list is initially built by the server from various external dataset while, for subsequent stages, the list is initially and dynamically built by the server based on intermediate internal traceroute measurements. During each stage the current seed set is recursively updated using neighbor IP collected by each monitor (the set $\{IP_r\}$ in Fig. 4). The initial seeds list used during the first stage is based on three inputs:

- Previously collected `mrinfo-rec` data is used as part of the seeds.

- "External"[6] traceroute data collected by Archipelago [3].
- "Initial"[6] traceroute campaign. As already stated by Spring et al. [6], it is possible to extract on an AS basis a set of *reachable prefixes*. Those prefixes are used as destinations for the initial internal traceroute campaign. This campaign is launched from all vantage points supporting the Paris traceroute tool.

Each input list is subject to a standard IP-to-AS mapping filtering process so that only IP addresses mapped to the targeted AS are used. Each monitor probes the network recursively (see Sec. IV-B for details). For "controlling" this recursion and then avoiding *inter-monitor* redundancy, the server computes a list of IP addresses, called the *Stop Set* ($\mathbb{ST}$ in Fig. 4 - see Sec. IV-C). This list contains IP addresses that have been previously discovered by MERLIN monitors and, thus, do not have to be probed again. The purpose is to avoid redundancy at two levels: *intra-monitor* (i.e., a given monitor probing several times the same router - see Sec. IV-B for details) and *inter-monitor* (i.e., a monitor re-probes a router already discovered by another monitor). Initially (i.e., for the first monitor in the first stage), the Stop Set is empty. Then, it is cumulatively populated based on the topological data brought back to the server: each local IP address of a discovered router ($\{IP_l\}$ in Fig. 4) is added to the Stop Set so that the next monitor in the ring will not probe it again.

B. MERLIN *Monitor*

Fig. 5 depicts the architecture of the MERLIN monitor. The monitor is composed of two parallel processes: *send*, in charge of sending probes to the network, and *receive*, in charge of processing the replies returned back by the routers. To minimize redundancy, the sending process never probes an IP address previously discovered: the "history" box in Fig. 5 is initialized with the Stop Set and is maintained up-to-date by the monitor, avoiding thus intra- and inter-monitor redundancy.

The *send* process is fed by both a static IP address list and a dynamic IP address list collected from replies. This dynamic list is used for recursion. When the monitor starts, the *send* process operates in a static mode using so IP addresses from the static list. Once replies are collected from the *receive* process, the dynamic list is built based on publicly routable neighbor IP addresses: the recursion is engaged. The *send* process enters in the dynamic mode and gives priority to targets of the dynamic list. Each time the dynamic list becomes empty (i.e., the current recursion is finished), the *send* process is again fed with remaining IP addresses from the static list. This recursion first scheme was a design choice that has been made in order to minimize the probability of reprobing a given router (see [21] for details). Moreover, this scheme allows for collecting a connected part of the probed network in a short timescale, increasing the reliability in case of topological changes. The fast dynamics of the Internet may lead to false connectivity inferences when routers are probed over a timescale greater than the one of potential changes [23].

During the dynamic mode, the probe inter-departure time is fixed to a given value α in the order of a second. In contrast,

---

[6]Here the terms external and internal respectively describe the use of data previously collected by others or through our own probing campaigns.
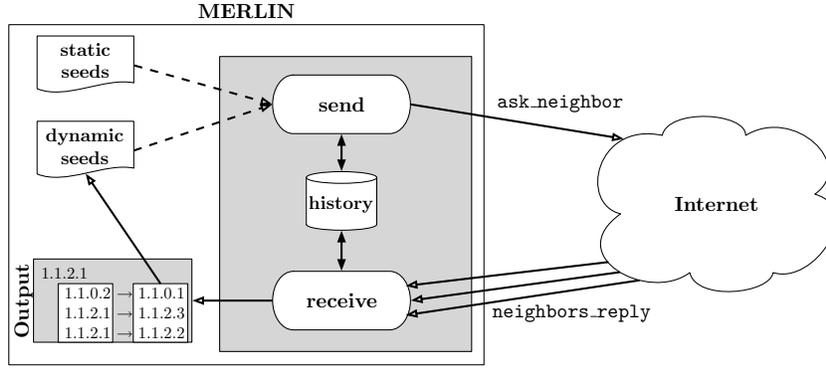
Fig. 5.   MERLIN monitor architecture

during the static mode, the inter-departure parameter is fixed to a lower value β with $\beta << \alpha$ in order to fasten the probing. The success rate of the static mode being much more lower than the one of the dynamic mode, the reprobing risk is then really lower. To summarize, the *send* process prioritizes its tasks as follows: (1) if a new router has been discovered, it marks all its local IP interfaces as already seen, (2) if there are neighbor IP addresses to probe, it elapses the probing with the timer α, (3) otherwise it uses the static list and elapses probes with the timer β. Finally, note that the recursion stops at routers that have no interfaces mapped to the AS of interest thanks to the use of a *patricia tree* populated by the server. Interested readers can find more details in [21].

C. MERLIN *Server*

The main server task is to keep track of the entire probing process by collecting the reassembled replies coming from each monitor and avoiding the injection of useless probes: while probing the same destination from different vantage points using traceroute is likely to give new information, once a router has responded to an IGMP query from a vantage point, it is useless to query it from another one (it is also necessary to stop the monitor recursion when discovering an already known router, see Sec. IV-D). Between two monitor probing runs, the server computes a new list of $CurrentSeed$ and its associated Stop Set based on the union of the already collected data. These two lists are then sent to the next monitor in order to increase the probing coverage while remaining as network friendly as possible: MERLIN minimizes the inter-monitor probing redundancy. Indeed, a MERLIN client stops its recursive probing exploration at an IP address belonging to the Stop Set. As described in [21], the IP interface list of a given router returned by `mrinfo` may be incomplete (see Sec. IV-D) such that, despite efforts to avoid redundancy, MERLIN may probe several times the same router. Fortunately, this risk does not imply the reprobing of an entire connected component: the monitor stops its recursion at the first duplicated probe thanks to the $StopSet$ containing all the IP interfaces belonging to already collected routers.

In order to improve the global coverage and offer a better view of the probed network (see Sec. IV-D), MERLIN also launches dynamic traceroute campaigns when the list of seeds given as input to a monitor becomes empty. In practice, the

end of an IGMP probing stage occurs when all current seeds respond or have been probed by all monitors. Those inter-stage traceroute campaigns are performed using a specific hitlist of IP addresses. For this purpose, the server constructs a set called $BorderIP$ consisting of the IP addresses located at the "border" of connected components. Those addresses correspond to the ones that have been collected as neighbor IPs but that do not reply directly to MERLIN. This list is then used by all monitors to select destinations for inter-stage traceroute campaigns. The objective is twofold: produce new seeds and improve the data returned by MERLIN replies. It allows for obtaining missing unicast information [21, Sec. IV.A.2.] as described in Sec. IV-D.

A current view of the $BorderIP$ set is maintained between each run to take benefit of the recursion across monitors. As long as new potential seeds are discovered (recursively – *intra stage* – and based on internal traceroute measurements – *inter stage*), the server continues to "feed" monitors. The server sequentially sends to monitors an updated list of IP addresses corresponding to new seeds to probe: $CurrentSeed$. This set consists of IP addresses not responding to other monitors, new neighbor IP addresses discovered during previous runs, or IP addresses collected through Paris traceroute campaign between two IGMP probing stages. However, the $CurrentSeed$ set excludes interfaces belonging to already collected routers and also the ones that have been already probed without success by the next monitor in the ring.

To control each probing period, MERLIN uses three main sets: $\mathbb{ST} = StopSeT$, $\mathbb{BP} = BorderIP$, and $\mathbb{CD} = CurrentSeeD$. The notation S^n refers to a current set view $S \in \{\mathbb{ST}, \mathbb{BP}, \mathbb{CD}, \{IP_l\}, \{IP_r\}, \{IP_v\}\}$ computed after the nth run performed on any monitor, n being the global identifier of the run number for all monitors. The three main sets used by MERLIN to control the global monitoring can be formally described as follows:

$$\mathbb{ST}^n = \bigcup_{\forall i \leq n} \{IP_l^i\}. \tag{1}$$

$$\mathbb{BP}^n = \bigcup_{\forall i \leq n} \{IP_r^i\} \setminus \mathbb{ST}^n. \tag{2}$$
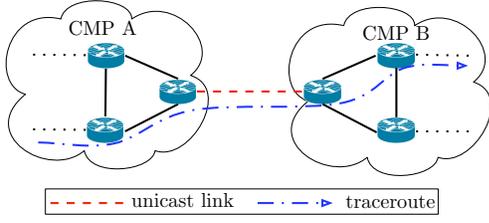
Fig. 6. Missing unicast link

$$\mathbb{CD}^n = \begin{cases} \{\{IP_s\} \bigcup \mathbb{BP}^n\} \setminus \mathbb{ST}^n\} \setminus \{IP_v^n\} & \text{at stage 0,} \\ \{\{T(X^p) \bigcup \mathbb{BP}^n\} \setminus \mathbb{ST}^n\} \setminus \{IP_v^n\} & \text{at stage p.} \end{cases}$$
(3)

where $\{IP_l\}$ is the set of all local IP interfaces (addresses on the left side of an `mrinfo` reply - see [24] for details on `mrinfo` output format), $\{IP_r\}$ is the set of all neighbor IP addresses (addresses on the right side of an `mrinfo` reply), and $\{IP_s\}$ is the initial set of static seeds. The sets $\mathbb{CD}^n$ and $\mathbb{ST}^n$ are used as input for the $(n+1)^{th}$ run performed by the next monitor in the ring. Note that the set $\mathbb{CD}^n$ excludes all IP addresses already probed by the current monitor v:

$$\{IP_v^n\} = \bigcup_{\forall i \in r(v) | i < n} CD^i.$$
(4)

with $r(v)$ being the set of runs performed at monitor v.

$T(X^p)$ denotes the set of IP addresses collected during the pth inter-stage traceroute campaign using the set X^p as hitlist. The set X^p is equal to the set $\mathbb{BP}^n \setminus \mathbb{BP}^t$ where t is the index of the first run of the previous $(p-1)^{th}$ IGMP probing stage. An IGMP probing stage ends after the nth run if the current set $\mathbb{CD}^n$ becomes empty. Either $\mathbb{BP}^n$ did not grow during the last m consecutive runs (m being the total number of monitors), or the nth monitor run reported a set of local IP addresses $\{IP_l^n\}$ such that $\mathbb{CD}^{n-1} \subseteq \{IP_l^n\}$ without increasing $\mathbb{BP}$, $\mathbb{BP}^n = \mathbb{BP}^{n-1}$.

As soon as the $\mathbb{CD}^n$ set of the pth stage is empty, the server computes the set $X^{(p+1)}$, performs an inter-stage traceroute campaign and then launches the new $(p+1)^{th}$ IGMP probing stage. This design choice corresponds to an IGMP probing first approach: inter-stage traceroute campaigns are used only when the IGMP recursion does not produce any new data all around the ring. It is worth to notice that when a new stage begins, it re-starts at the first monitor of the ring and MERLIN does not rely on any specific monitor ordering within the ring.

Initially, all IP sets are empty except $\{IP_s\}$. As soon as sets $\mathbb{CD}^n$ and $X^{(p+1)}$ are both empty, the MERLIN server enters in the "reconnect mode" described in Sec. IV-D. The discovery probing phase is finished, the server now focuses on the router level graph reconstruction. More details on the server and algorithmic sketches introduced in this section are explained in [22].

D. Reconnecting the Disconnected Components

When the discovery probing phase of MERLIN stops, MERLIN enters in a post-processing phase dedicated to the router level graph refinement. After having removed potential
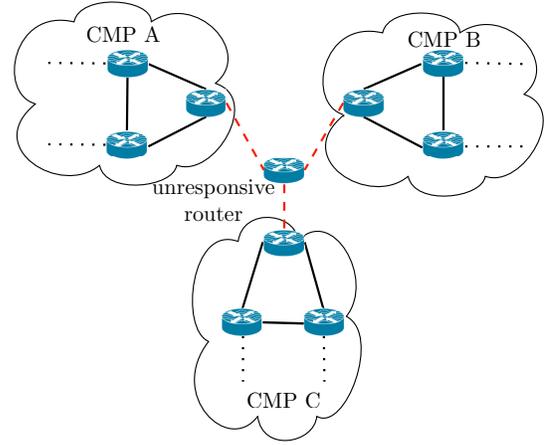

Fig. 7. The unresponsive router issue

duplicates due to pure unicast IP addresses[7] and applied a router-to-AS election [13] focusing on routers belonging to the targeted AS, MERLIN performs two reconnecting processes: (i) resolving the 1-unicast-hop distance problem, (ii) fixing the unresponsive router issue. These two mechanisms potentially reconnect disconnected components of a partitioned topology.

Using internal traceroute paths collected during the campaign, (i), MERLIN is able to reconnect routers that are connected through a unicast link. Indeed, if a traceroute path consecutively crosses two responding routers that are not adjacent according to the multicast connectivity (the traceroute path reports two consecutive IP interfaces belonging to this pair of routers: the second IP address is necessarily either a pure unicast interface or a multicast interface with no known neighbor), it implies that, actually, a unicast link connects them. Thus, if those two routers do not belong to the same connected component, MERLIN is able to virtually merge them into a single larger component. This situation is illustrated in Fig. 6: a traceroute path consecutively crosses components A and B. Furthermore, as suggested in previous work [21], if a unicast IP address of a multicast responding router is discovered through a traceroute path, then, at the next stage, if this IP responds to one of the monitor, the server aliases this unicast IP address with the previously collected multicast router (the server merges the two responses into a single one).

Using an alias resolution technique such as Ally[8] [6] on the final $\mathbb{BP}$ set, (ii), MERLIN tries to reconnect disconnected components that may be 2-multicast-hop distant. For example considering the unresponsive multicast router problem mentioned earlier (see Sec. III-D), a MERLIN IGMP probe may fail on a multicast neighbor IP belonging to a multicast enabled router that is not `mrinfo` compliant (local filtering). Such an IP thus belongs to the $\mathbb{BP}$ set: although they are discovered through MERLIN they do not appear as local interface ($\{IP_l\}$) of any collected router.

---

[7]This lack is due to multicast routers responding with a unicast IP address not present in the list of multicast reported interfaces [21]. Such an address is then added to the router as a local interface.

[8]Note that MERLIN can work whatever the alias resolution technique considered, Ally is just used here as a practical example.
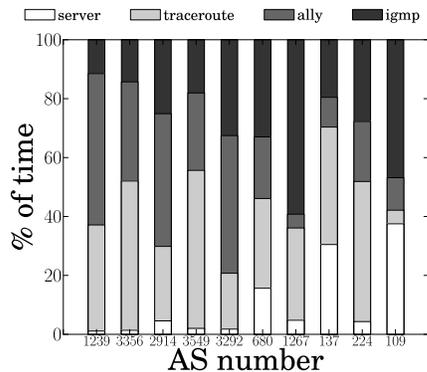
Fig. 8.   Discovery time per AS

Fig. 9.   Probing evolution for AS1239

Indeed, routers adjacent to an unresponsive multicast router are potentially able to capture a subset of its interfaces through their multicast adjacencies (dashed red lines in Fig. 7). Then, applying an alias resolution campaign with a tool like Ally on those IP addresses, MERLIN can rebuild these unresponsive routers and, consequently, reconnect together routers that are 2-multicast-hop distant. Fig. 7 provides an illustration of such a reassembling technique when involved routers are located in different connected components: IGMP components A, B and C appear as disjoint components due to a single unresponsive router located in the center of the figure. Aliasing the border IP addresses of such disjoint components (part of the $\mathbb{BP}$ set) allows MERLIN to rebuild the missing and "connecting" router leading so to the reconstruction of the topology. Another possible way to reconnect such a disconnected topology would be to generalize the use of an alias resolution technique on all traceroute measurements (this is out of the scope of this paper). This might allow one to extend the network view by focusing on intermediary IP addresses between MERLIN responding routers. To limit the space of IP addresses pairs that we should explore some rules and heuristics must be defined (for example using DNS information to define potential cluster of IP belonging to the same router). Such approaches are left for further work.

V. DEPLOYMENT

A. Methodology

We conducted a set of experiments using six monitors distributed across the Internet: Strasbourg (France), Louvain-la-Neuve (Belgium), Napoli (Italy), San Diego (USA), Redwood City (USA), and Hamilton (New Zealand). The main advantage of using several vantage points is the ability to circumvent IGMP filtering applied on some border routers that limit the scope of MERLIN probes. Interested readers can refer to [21, Sec. 4.1.] for a study on the marginal utility of considering several vantage points. The MERLIN server is located in the Osiris Stub network in Strasbourg. ASes targeted during our experiments are those listed in Table I. Experiments have been conducted between November 9th 2010 and November 17th 2010, leading to six probing campaigns. Monitors are calibrated using those parameters: $\alpha = 500ms$ and $\beta = 50ms$ such that they are much more scalable and efficient than using the original `mrinfo` command. Interested
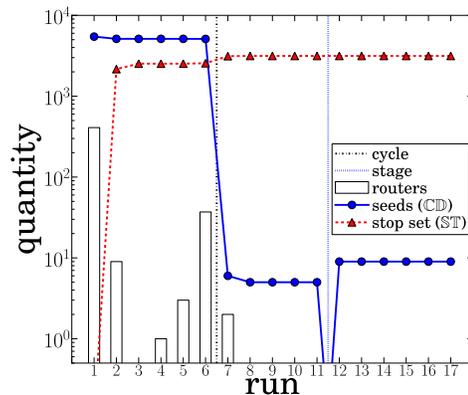
readers may find explanations about those values in [21]. The collected topologies are publicly available to the research community[9].

B. Global Statistics

Table II provides some details on probing efficiency such as the total discovery time and the number of IP addresses collected. But, more specifically, it focuses on the number of nodes (routers and layer-2 devices, such as switches), links, and connected components collected during our MERLIN probing campaigns. The results have been averaged over the six probing campaigns performed (the standard deviation over the mean is also provided in order to show the low variation between campaigns). The values indicated in Table II correspond to the final topologies, i.e., topologies obtained after the post-processing: duplicates removal (by merging of unicast IP addresses into multicast routers), IP-to-AS mapping and router-to-AS election, reconnection of disconnected components, and layer-2 aware graph conversion.

On the one hand, we can notice the difficulty to provide completely connected topologies despite our efforts. However, on the other hand, we observe an increase in the IGMP probing coverage thanks to the use of traceroute seeds. Comparing values of the last two columns of tables I and II suggest that the IP level coverage of IGMP probing has been extended thanks to our new methodology[10]. The use of such seeds allows us to improve the probing scope of MERLIN but at the cost of collecting disconnected multicast regions inside the same AS. Depending on the AS, this effect is more or less important: while Sprint and IUNet are captured as almost connected topologies, TDC and Level3 graphs are clearly more partitioned. Generally speaking, even in the presence of disconnected topologies, it is possible to extract some typical graph patterns such as PoP structures.

C. Probing Evolution

Fig. 8 shows the relative duration of each phase of the MERLIN probing (on November 9th 2010). The white bar

---

[9] http://svnet.u-strasbg.fr/merlin

[10] Note that values given for Archipelago [3] correspond to the number of traceroute external seeds injected in the set $\{IP_s\}$.

TABLE II
GLOBAL STATISTICS ON TOPOLOGIES COLLECTED WITH MERLIN

AS		Topology			Connected Components				Discovery Time		IP addresses collected			
		Nodes		Links										
Level	Number	Routers		L2 devices	L3 links	Number		Largest		Minutes		Archipelago		MERLIN	
Tier-1	AS3549	306	±10	1 ±1	667 ±11	74	±8	219	±3	252	±39	4,341	±94	2,509	±53
	AS3356	399	±3	85 ±1	530 ±14	130	±11	47	±13	384	±14	8,391	±10	3,197	±57
	AS1239	428	±4	6 ±1	816 ±19	23	±2	392	±12	370	±35	5,419	±25	2,405	±37
	AS2914	200	±2	18 ±1	395 ±5	10	±1	191	±3	165	±13	5,707	±53	1,487	±30
Transit	AS3292	223	±4	67 ±1	478 ±36	72	±2	11	±1	232	±14	7,657	±42	1,909	±23
	AS1267	386	±4	215 ±2	810 ±1	23	±3	361	±0	37	±7	2,817	±8	2,771	±15
	AS680	270	±20	16 ±3	314 ±16	102	±9	51	±0	62	±7	2,648	±16	2,974	±249
Stub	AS137	126	±8	13 ±0	141 ±8	36	±4	75	±1	41	±18	1,385	±32	731	±45
	AS109	99	±1	10 ±0	147 ±0	19	±1	16	±1	29	±9	254	±9	466	±3
	AS224	281	±12	10 ±0	369 ±17	18	±2	257	±17	36	±6	786	±8	2,473	±153

corresponds to the server running time. The black bar refers to cumulative IGMP probing time (i.e., the time required by all monitors to do IGMP probing). The bar labeled "traceroute" refers to the time allocated to traceroute probing, i.e., the time required to build the static seed list and the time required to perform traceroute for discovering unicast links. Finally, the bar labeled "ally" refers to the post processing phase in which alias resolution is used for fixing the unresponsive router issue.

In most of the cases, the server time is negligible. The relative time allocated to the server mainly depends on the quantity of managed seeds and collected answers. For large multicast networks, this period is really short compared to the pure probing period (i.e., traceroute and IGMP probing) and the reconnection phase (Ally). However, for small Stub networks, the server time may seem significant because the pure probing period is really short and there is a constant time required by the server setup and control. Depending on the targeted AS size, relative time of each period differs. Indeed, the pure probing period depends on the number of prefixes (and their sizes) belonging to the ISP targeted by MERLIN. In contrast, the Ally period only depends on the size of the border IP set, $\mathbb{BP}$. If this set contains n IP addresses at the end of the probing period, and since we perform an almost brute force approach, the required time complexity is in $\approx \frac{n \times (n-1)}{2}$. Note that generally, for large ASes, this period may represent about 40% of the total time required by MERLIN but may be strongly reduced using recent improvements in the alias resolution field [9].

Fig. 9 provides a more precise view of the behavior of MERLIN when probing the Sprint network (on November 9th 2010). In addition to the number of routers collected during each run, we plot (note that the vertical axis is in log-scale) the evolution of the two main sets maintained by the server: the Stop Set ($\mathbb{ST}$) and the Current Seeds ($\mathbb{CD}$). We can notice that the exploration is almost finished at the end of the first cycle. While the Stop Set remains almost constant after the 6th run, the Current Seeds set, $\mathbb{CD}$, starts to become empty due to the lack of new discovered seeds. It means that after taking advantage of each vantage point location specificity on the static list IP_s (their own ability to circumvent IGMP filtering), the second cycle is almost useless: the Border IP set $\mathbb{BP}$ seems to produce too few IP addresses to really take advantage of potential forwarding changes between monitors on dynamic seeds. In the same way, the second stage does not bring any new data, the internal traceroute paths only bring
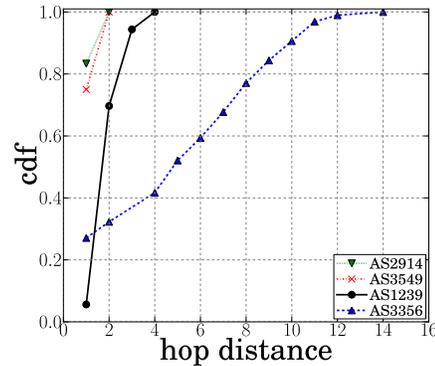
Fig. 10. Distance between disconnected components (November 9th 2010)
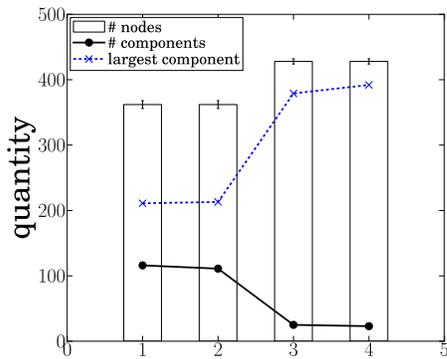
ten new seeds that do not respond to IGMP probes. However, those seeds are also used as a basis for the reconnection phase: traceroute measurements allow us to find new unicast links.
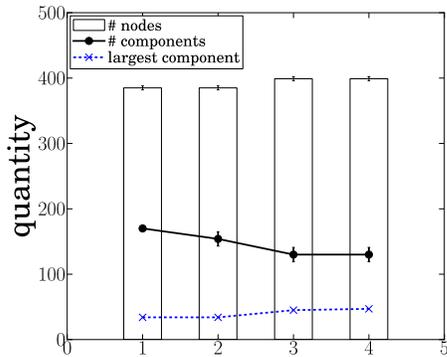
D. Disconnected Components

In this section, we first investigate a specific graph property: the connectivity. Then, we discuss the efficiency of our reconnection technique on two specific cases.

Fig. 10 shows the distance distribution (as a cumulative mass) of connected components for Tier-1 ASes as collected by MERLIN on November 9th 2010. This distance distribution takes into account the "minimal hole distances" between disconnected components considering a graph where connected components are nodes and distances refer to the number of IP hops in the underlying traceroute level graph, e.g., two multicast zones are distant of x hops if and only if the minimal traceroute path between them contains x hops and there does not exist any other traceroute path connecting these two components that goes through other components that are less distant between themselves. Furthermore, this computation depends on the traceroute performed on the targeted AS, and then, by definition, not necessarily traces all possible distances and shortest paths as well.

For most Tier-1, the situation seems quite normal: traceroute reveals that, in general, disconnected components are separated by at most three hops. This situation may be due to a core sub-network made of PoPs that are multicast enabled but that contains routers not supporting IGMP probing. Such clusters of PoP seem to have a diameter of three or four hops in general. However, on Level3 (AS3356), as the maximal measured
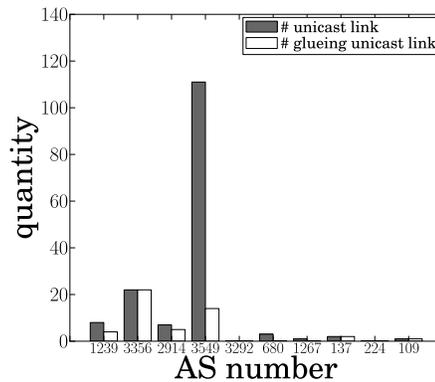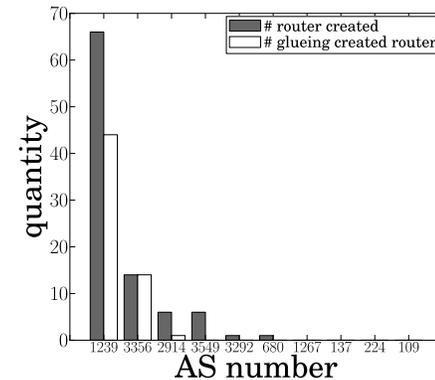
(a) AS1239

(b) AS3356

Fig. 11. Efficiency and evolution of the reconnection phase

(a) missing unicast link (traceroute)

(b) unresponsive router (Ally)

Fig. 12. Graph reconstruction - efficiency

distance is higher, such structures seem larger and impacting border routers of other intra-domain routing areas as well. Thus, except for AS3356 (the connected components seem to be far from each other), we may expect that our reconnection technique will be able to merge 60% of the components (from one hop to two hops). However, note that the number of useful traceroutes (the ones crossing disconnected multicast zones) is far from covering all possible distances between connected components. Thus, our set of active traceroutes $\bigcup_{\forall p} T(X^p)$ only gives some insights about the nature of the problem: are the multicast zones far from each other?

Fig. 11 shows the evolution and the efficiency of the reconnection phase MERLIN for AS1239 and AS3356. Results are averaged over the different probing campaigns and the standard deviation around the mean is provided. Each column depicts the connectivity situation before and after a reconnection treatment (see Sec. IV-D). Transitions $1 \rightarrow 2$ and $3 \rightarrow 4$ correspond to the use of traceroute for discovering missing unicast links while the transition $2 \rightarrow 3$ refers to the use of Ally. As expected, our reconnection technique works better on Sprint than on Level3 when we look at the evolution of the number of connected components (full line). For Sprint, the main and largest connected component "captures" smaller multicast zones (dashed line) while a low proportion of isolated routers remains disconnected (see Table II). For both networks, this is the use of Ally that provides the best efficiency but at the cost of a huge computation time. Note that both techniques (i.e., traceroute and Ally) allow us to improve

the graph connectivity inside connected components as well.
Fig. 12 allows us to better understand the situation:

(a) Fig. 12(a) gives the absolute number of unicast links retrieved thanks to traceroute according to two classes: the total number of added links (grey bar) and the number of links able to reconnect disconnected multicast zones (white bar).
(b) Fig. 12(b) provides the absolute number of "virtual routers" built after the use of Ally (grey bar), as well as the number of "virtual routers" allowing to reconnect scattered multicast zone (white bar).

Note that, obviously, the smaller the AS, the lower the absolute values. On the one hand, we can notice that both techniques work very well on Sprint, in particular for Ally: the unresponsive router issue is almost fixed on this topology. Furthermore, note that results provided in Fig. 12 only take into account the effective number of virtual links and routers (white bars) that reconnect the topology such that if several links or routers connect a given pair of disconnected areas, we count only one among them. On the other hand, although the traceroute technique works quite well on Level3 and Global Crossing, Ally has a limited positive impact on those topologies. This result intends to show that there is a need to push further our reconnection mechanisms by generalizing them on the unicast part of the network as well as on unresponsive multicast parts. Indeed, performing alias resolution campaigns on border IP addresses coming both from IGMP probing and traceroute traces, one may be able to reconnect routers that

are much more distant than 2 hops. However, such generalized campaigns require extensive probing so that one must rely on several heuristics and efficient algorithms to remain scalable.

VI. RELATED WORK

MERLIN adopts a hybrid approach mainly based on traceroute and IGMP probing and uses alias resolution to improve reconstructed topologies. It has already been demonstrated how hybrid approaches can be profitable in the context of topology discovery at the router level [25].

The main advantage of MERLIN is that, in a single IGMP reply, a router lists all its multicast IP interfaces and the IP addresses of its neighbor routers. Thus, MERLIN suffers less from the alias resolution problems affecting pure traceroute campaigns. Second, all links of a responding router are captured, even if the IGP weight of a link is high and no data packets are forwarded over it. Furthermore, the IGMP monitoring load is smaller than the one induced by traceroute campaigns. Indeed, a single IGMP probe per router is sufficient to collect the topology of a multicast enabled network. Compared to traceroute [8] and its variants [3], [4], [5], [6], [15], [26], MERLIN has both drawbacks and advantages. The main drawback of MERLIN is that it can only be used on routers having IPv4 multicast activated. IPv4 multicast is not always enabled in IP networks, but thanks to the deployment of video or television services that rely on IP multicast, more and more ISP networks have enabled multicast. Standard traceroute is only able to discover a single path from the source to the destination. To discover more topology information, it is required to increase both the number of destinations and vantage points [5], [27], [28]. Paris Traceroute [15] is able to discover load balancing routers, as well as the set of paths joining those load balancing routers. However, this works mostly for intra-domain routers, BGP load balancing being much more difficult to detect. MERLIN is able to discover all links between routers from a single source if domains authorize multicast. In particular, MERLIN is able to report backup links inside and between domains. *iPlane* [4] is a service providing Internet path performance predictions by building an annotated map of the Internet. Measurement points are deployed on the PlanetLab testbed and the network is daily probed (traceroute probing). The obtained data is then postprocessed in order to provide finer grained information, such as router level topology (the alias resolution techniques used are DNS based [6] and address-based [29], [30]), IP-to-AS mapping, IP-to-PoP mapping, bandwidth estimation, etc. Discarte [31] is an example of a new scalable probing technique that relies on existing techniques. Indeed, Discarte extends traceroute-like probing by using the *Record Route* option defined in the IP header. This option is a way to record the route of an IP packet. If a router detects the *Record Route* option in the IP packet received and this router enables this option, the router must record in the IP header the address it uses for forwarding the packets. Using the *Record Route* option within traceroute comes with several advantages, one of them being the ability to gather multiple IP interfaces of a router (and, thus, perform alias resolution) without additional probing. However, it has the drawbacks that this option is length limited (only nine IP addresses can be inserted in

this IP option) and is not broadly supported by routers. In addition, Discarte uses a logical inference and constraint solving technique to merge Record Route and traceroute data. Hynetd [25] also uses *Record Route* to improve the discovering process. *Rocketfuel* [6] is probably the tool that is the closest to MERLIN. Indeed, Rocketfuel tries to collect the best possible picture of ISPs, typically ones that are in the center, not on the edges of the Internet. Rocketfuel is based on traceroute and makes used of, roughly, 800 vantage points and nearly 300 traceroute web servers. Rocketfuel uses *Ally* to convert interface level maps into router level ones. Compared to Rocketfuel, MERLIN does not require a generalized alias resolution phase as it provides, in a single IGMP packet, all multicast interfaces of the targeted router. It only relies on Ally to improve its cartography on the non-responding borders of multicast areas. Further, by definition of traceroute, Rocketfuel will likely provide a tree-like map, while MERLIN is able to map a whole connected component.

VII. CONCLUSION

Accurate and efficient topology discovery at router level is a difficult and challenging task because of the heterogeneity, scale, and complexity of the current Internet. In addition, the continuous temporal evolution of network configurations seriously compromises the results of approaches designed and developed in the past. In this paper, we proposed a platform using an hybrid approach – based on IGMP probing, traceroute, and alias resolution – we called MERLIN. We presented the MERLIN architecture, its features, and how it benefits from each of the three probing techniques it relies on while remaining network friendly. Extensive experimental campaigns have shown how MERLIN is able to efficiently collect and reconstruct the network topology of several real ASes. Those campaigns demonstrated how MERLIN is able to overcome the multicast scope limitation. Finally, MERLIN is not necessarily dedicated to map ISPs: it can also be deployed on the whole Internet. Future works should reveal how to control various MERLIN monitors for efficiently probing the whole Internet.

REFERENCES

[1] B. Donnet and T. Friedman, "Internet topology discovery: a survey," *IEEE Commun. Surveys Tutorials*, vol. 9, no. 4, pp. 2–15, December 2007.

[2] H. Haddadi, G. Iannaccone, A. Moore, R. Mortier, and M. Rio, "Network topologies: Inference, modeling and generation," *IEEE Commun. SurveysTutorials*, vol. 10, no. 2, pp. 48–69, April 2008.

[3] k. claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: from art to science," in *Proc. IEEE Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH)*, March 2009.

        IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 29, NO. 9, OCTOBER 2011

[4] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, November 2006.

[5] Y. Shavitt and E. Shir, "DIMES: Let the internet measure itself," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, October 2005.

[6] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM SIGCOMM*, August 2002.

[7] M. Luckie, "Scamper: a scalable and extensible packet prober for active measurement of the Internet," in *Proc. USENIX/ACM Internet Measurement Conference (IMC)*, November 2010.

[8] V. Jacobson et al., "traceroute," UNIX," man page, 1989, see source code: ftp://ftp.ee.lbl.gov/traceroute.tar.gz.

[9] A. Bender, R. Sherwood, and N. Spring, "Fixing Ally's growing pains with velocity modeling," in *Proc. ACM/USENIX Internet Measurement Conference (IMC)*, October 2008.

[10] M. H. Gunes and K. Sarac, "Resolving IP aliases in building traceroute-based internet maps," *IEEE/ACM Trans. Netw. (ToN)*, vol. 17, no. 6, pp. 1738–1751, December 2009.

[11] J. Sherry, E. Katz-Bassett, M. Pimenova, H. V. Madhyastha, A. Krishnamurthy, and T. Anderson, "Resolving IP aliases with prespecified timestamps," in *Proc. ACM/USENIX Internet Measurement Conference (IMC)*, November 2010.

[12] P. Mérindol, V. Van den Schriek, B. Donnet, O. Bonaventure, and J.-J. Pansiot, "Quantifying ASes multiconnectivity using multicast information," in *Proc. ACM USENIX Internet Measurement Conference (IMC)*, November 2009.

[13] J.-J. Pansiot, P. Mérindol, B. Donnet, and O. Bonaventure, "Extracting intra-domain topology from mrinfo probing," in *Proc. Passive and Active Measurement Conference (PAM)*, April 2010.

[14] P. Mérindol, B. Donnet, O. Bonaventure, and J.-J. Pansiot, "On the impact of layer-2 on node degree distribution," in *Proc. ACM/USENIX Internet Measurement Conference (IMC)*, November 2010.

[15] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with Paris traceroute," in *Proc. ACM/USENIX Internet Measurement Conference (IMC)*, October 2006.

[16] B. Huffaker, D. Plummer, D. Moore, and k. claffy, "Topology discovery by active probing," in *Proc. Symposium on Applications and the Internet Workshop (SAINTS)*, January 2002.

[17] S. Deering, "Host extensions for IP multicasting," Internet Engineering Task Force, RFC 1112, August 1989.

[18] T. Pusateri, "Distance vector multicast routing protocol version 3 (DVMRP)," Internet Engineering Task Force, Internet Draft (Work in Progress) draft-ietf-idmr-dvmrp-v3-11, October 2003.

[19] P. Sharma, E. Perry, and R. Malpani, "IP multicast operational network management: Design, challenges, and experiences," *IEEE Network*, vol. 17, no. 2, pp. 49–55, March 2003.

[20] V. Jacobson, "mrinfo," 1995, http://cvsweb.netbsd.org/bsdweb.cgi/src/usr.sbin/mrinfo/?only_with_tag=MAIN.

[21] P. Mérindol, B. Donnet, J.-J. Pansiot, M. Luckie, and Y. Hyun, "MERLIN: MEasure the Router Level of the INternet," in *Proc. 7th Euro-NF Conference on Next Generation Internet (NGI)*, June 2011.

[22] P. Marchetta, "An Internet topology discovery approach based on multicast," October 2010, M.S. Thesis, University of Napoli Federico II, http://wpage.unina.it/pescape/doc/MSThesisPMarchetta.pdf.

[23] C. Magnien, F. Ouédraogo, G. Valadon, and M. Latapy, "Fast dynamics in internet topology: preliminary observations and explanations," in *Proc. Fourth International Conference on Internet Monitoring and Protection (ICIMP09)*, 2009.

[24] B. Huffaker, k. claffy, and E. Nemeth, "Tools to visualize the Internet multicast backbone," in *Proc. International Networking Conference (INET)*, June 1999.

[25] A. Botta, W. de Donato, A. Pescapé, and G. Ventre, "Discovering topologies at router level: Part II," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, November 2007.

[26] N. Spring, D. Wetherall, and T. Anderson, "Scriptroute: A public Internet measurement facility," in *Proc. USENIX Symposium on Internet Technologies and Systems (USITS)*, March 2002.

[27] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Deployment of an algorithm for large-scale topology discovery," *IEEE J. Sel. Areas Commun. (JSAC), Sampling the Internet: Techniques and Applications*, vol. 24, no. 12, pp. 2210–2220, Dec. 2006.

[28] K. Chen, D. R. Choffnes, R. Potharaju, Y. Chen, F. E. Bustamante, D. Pei, and Y. Zhao, "Where the sidewalk ends: Extending the Internet AS graph using traceroutes from P2P users," in *Proc. ACM CoNEXT*, December 2009.

[29] J. J. Pansiot and D. Grad, "On routes and multicast trees in the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 1, pp. 41–50, January 1998.

[30] K. Keys, "iffinder," a tool for mapping interfaces to routers. See http://www.caida.org/tools/measurement/iffinder/.

[31] R. Sherwood, A. Bender, and N. Spring, "Discarte: a disjunctive Internet cartographer," in *Proc. ACM SIGCOMM*, August 2008.

Pietro Marchetta received the M.S. Laurea Degree in Computer Engineering in 2010 at the University of Napoli Federico II (Italy). Currently, he is a Ph.D. student at the Department of Computer Engineering and Systems of the University of Napoli Federico II. His research interests are network measurements, traffic analysis, and topology discovery.

Pascal Mérindol received his Ph.D. degree from the University of Strasbourg (France) in 2008. Then, he spent two years in Belgium at the Université catholique de Louvain as a post-doc researcher. He is now Assistant Professor at the Networks and Protocols team of the LSIIT laboratory (https://lsiit.u-strasbg.fr/rp-en/) in Strasbourg. His main research topics are routing and Internet topology discovery.

Benoit Donnet received his MS degree in computer science from the Facultés Universitaires Notre Dame De La Paix (Namur - Belgium) in 2003. Mr. Donnet received his Ph.D. degree in computer science from the Université Pierre et Marie Curie in 2006. He is currently FNRS fellow at the Université catholique de Louvain in the IP Networking Lab (http://inl.info.ucl.ac.be). His research interests are in Internet measurements, Bloom filters, and traffic engineering techniques.

Antonio Pescapé is an Assistant Professor at the Department of Computer Engineering and Systems of the University of Napoli Federico II. He received his M.S. Laurea degree in computer engineering and his Ph.D. in computer engineering and systems, both from the University of Napoli Federico II. His research interests are in the networking field with focus on models and algorithms for Internet traffic, network measurements and management of heterogeneous IP networks, and network security. He has coauthored over 80 journal and conference publications and he is co-author of several patents pending.

Jean-Jacques Pansiot received a M.Sc. in Computer Science from Nancy University (France, 1972), a Ph.D. in Computer Science from Cornell University (USA, 1976) and a "Doctorat d'Etat" from the University of Strasbourg (France, 1983). He joined the Department of Computer Science from the same University where he was successively appointed as assistant professor, associate professor and full professor (1984). He leads the Network and Protocol team of the LSIIT Laboratory. His research interests include routing, multicasting, traffic engineering, and Internet cartography.