MERLIN: MEasure the Router Level of the INternet

Pascal Mérindol
LSIIT
Université de Strasbourg

Benoit Donnet
ICTEAM
Université catholique de Louvain

Jean-Jacques Pansiot
LSIIT
Université de Strasbourg

Matthew Luckie
WAND Network Research Group
Waikato University

Young Hyun
CAIDA
University of California, San Diego

Abstract

The Internet topology discovery has been an extensive research subject those last years. While the raw data is collected using large traceroute campaigns, additional probing and/or extensive computation are required to gather subsets of IP addresses into single identifiers corresponding to routers. This process, known as alias resolution, leads to a router level map of the Internet.

In this paper, we push further the Internet router level mapping with a new probing tool called MERLIN. MERLIN is based on `mrinfo`, a multicast management tool. `mrinfo` is able to silently collect all IPv4 multicast enabled interfaces of a router and all its multicast links towards its neighbors: it does not need or rely on any alias resolution mechanism. In addition, MERLIN comes with the advantage of being much more scalable than standard data gathering techniques. In this paper, we deploy and evaluate the performance of MERLIN. We demonstrate that the use of several vantage points is crucial to circumvent IGMP filtering in order to collect large amounts of routers. We also investigate the completeness of MERLIN by providing a lower bound on the proportion of information that it may miss. Finally, our dataset and the MERLIN implementation are freely available.

1 Introduction

Internet topology discovery has been an intense research subject during the past decade [1, 2]. Most of the deployed tools are based on *traceroute* [3, 4, 5, 6, 7, 8]. Traceroute discovers the Internet topology at the interface level, i.e., the IP interfaces of routers and end-hosts. All routers and some hosts have multiple interfaces, and each interface may appear as a separate entity in this topology. The resulting graph consists of the link-layer connections between those pseudo-nodes. These may not be point-to-point beneath IP: there may be tunnelling across other layer protocols, such as MPLS, and there might be traversal of layer-2 devices [9].

If one wants to build a topology map at the router level, it is necessary to gather all discovered interfaces of a given router into a single identifier. This summary technique is called *alias resolution* [10, 11, 12, 13, 14]. The accuracy of alias resolution has an important effect on the observed graph characteristics such as the node degree distribution [15]. However, alias resolution comes with several drawbacks. First, it is based on a preliminary traceroute campaign. Traceroute is known to be intrusive and redundant although improvements have been proposed to reduce its impact on the network [16]. It is also likely that traceroute will not discover all interfaces of a given router (in particular the ones used for backup paths). Second, alias resolution is either intrusive (it requires additional probing), or computation expensive (it requires a lot of post-processing). Finally, alias resolution is not entirely accurate as it might generate false positives, i.e., two IP addresses are tagged as aliases while they are not.

Recently, `mrinfo`, a multicast management tool has been used for topology discovery [17]. `mrinfo` comes with the strong advantage of listing all multicast interfaces of a router and its multicast links towards others using a single probe. `mrinfo` offers, by design, a router-level view of the topology: it does not suffer from the same lacks generated by combined traceroute and alias resolution techniques. However, its view is limited to multicast components of the Internet and, in the same way that ICMP messages may be rate limited or filtered for traceroute, IGMP messages can be filtered by some ISPs [18].

In this paper, we start by pointing out several technical limitations of `mrinfo`. In particular, `mrinfo` suffers from a fragmentation issue that leads to an important loss in the data collected. Another issue is the lack of multiplexing support. In order to fix these limitations, we implement, deploy, and evaluate a new tool: *MEasure the Router Level of the INternet* (MERLIN). MERLIN allows one to infer the multicast map of the Internet at the router level. MERLIN fixes the `mrinfo` technical issues and even goes further by increasing the amount of information collected. MERLIN is designed for large scale topology discovery campaigns.

We deploy MERLIN on several geographically-distributed vantage points and demonstrate that each vantage point is able to discover a significant portion of unique routers (i.e., routers that cannot be seen by other vantage points).[1] Because `mrinfo` -like probing is only applicable to multicast-enabled routers, we investigate the notion of *completeness* and provide a lower bound on the quantity of topological data that MERLIN may miss compared to standard probing techniques. We also provide a detailed description of the behavior of our tool and insights to calibrate and efficiently use it.

The remainder of this paper is organized as follows: Sec. 2 discusses `mrinfo` and its limitations; Sec. 3 presents our new tool MERLIN and discusses calibration procedures and limitations; Sec. 4 evaluates the performance of MERLIN; Sec. 5 positions our work compared to the state of the art; finally, Sec. 6 concludes this paper by summarizing its main achievements and discussing future research directions.

2 MRINFO

This section focuses on the original `mrinfo` client.[2] We first quickly describe the basics of this tool (Sec. 2.1). We next explain our data collection methodology (Sec. 2.2) and, finally, discuss and quantify the limitations of the initial `mrinfo` client (Sec. 2.3).

2.1 Tool Description

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 [19]. Several tools have been developed to monitor and debug the MBone [20]. Most of these tools have been deprecated with the replacement of DVMRP by the Protocol Independent Multicast (PIM) family of multicast routing protocols with one notable exception: `mrinfo` [21].

`mrinfo` uses the *Internet Group Management Protocol* (IGMP) [22]. DVMRP has defined two special types of IGMP messages that can be used to monitor routers [19]. Although current IPv4 multicast routers no longer use DVMRP anymore, they still support those special IGMP messages. Upon reception of an IGMP `ASK_NEIGHBORS` message, an IPv4 multicast router will reply with an IGMP `NEIGHBORS_REPLY` message that lists all its multicast adjacencies with some information about their state. Interested readers can find further details on `mrinfo` in [17, 23, 9].

2.2 Data Collection

Previously, `mrinfo` measurements were conducted recursively with `mrinfo-rec` [23, 9], which would probe a target with `mrinfo` and then recursively invoke `mrinfo` on all 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, we conducted daily `mrinfo-rec` runs in order to understand the dynamics of the Internet graph. To maximize discovered topology, we used the set of responding routers of a given day as the seed for the next day's recursive run.[3] 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 multicast map. Second, at the beginning of 2007, the opposite circumstance significantly reduced the number of reachable routers. Such sudden and significant changes cannot be due to network dynamics: they are an artifact of `mrinfo-rec` launched from only a single vantage point, making data collection susceptible to filtering.

Moreover, `mrinfo-rec` is not scalable to large experiments and the initial implementation of the `mrinfo` client suffers from several drawbacks as explained in Sec. 2.3. Our objective in this paper is to propose a new `mrinfo` implementation capable to probe millions of IPs in a reasonable timescale.

Furthermore, we conduct a study using six vantage points 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 these six vantage points is the ability to circumvent IGMP filtering applied on some border routers limiting so the scope of

[1]Our dataset is freely available online at `http://inl.info.ucl.ac.be/content/mrinfo`.

[2]`mrinfo` belongs to the `mrouted` multicast toolkit containing the multicast daemon `mrouted` and several management utilities such as `mtrace`. There exist several versions of this package whose behavior depends on the platform. For instance, the version running on Cisco routers does not provide the same results as the one running on end-hosts.

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

`mrinfo` probes. We discuss their utility in a dedicated section (Sec. 4.1). Finally, and again in contrast to the previous approach, we use a large list of IP addresses as seeds. This list is made of 1,643,005 IP addresses selected as follows:

- 1.2 million addresses from CAIDA's Archipelago traceroute measurements [5],
- 3,580 addresses from known topologies provided by research and educational networks,
- 24,429 addresses from a Tier-1 ISP,
- 155,674 addresses from traceroute, record route, and IP timestamps measurements issued from the Reverse Traceroute system [24] and
- 224,762 addresses that initially responded to `mrinfo` probes using the four previous datasets.

The data discussed in this paper was collected in July 2010 by running MERLIN on all six vantage points (each vantage point using the same list of IP addresses as seeds). Together the six vantage points collected 480,000 IP addresses aggregated into almost 50,000 routers scattered in more than 3,000 ASes. Note that we apply several pre-processing filters to discard redundant and useless information. The data considered in this paper corresponds to the union of all relevant information collected through all vantage points.

The raw data collected is available online at `http://inl.info.ucl.ac.be/content/mrinfo`. Details about the architecture of our new tool and its performance are given in Sec. 3 and Sec. 4.

2.3 Implementation Issues

We recently discovered that the initial `mrinfo` implementation [21] suffers from several issues and limitations. In this section, we investigate two critical problems: the lack of support for IGMP-fragmented `NEIGHBORS_REPLY` messages (Sec. 2.3.1) and the inability to multiplex IGMP-based measurements (Sec. 2.3.2).

While the first problem is simply a shortcoming of the initial `mrinfo` client, the second problem is more challenging as it raises the question of a compromise between the efficiency and the correctness of a large-scale `mrinfo` campaign.

2.3.1 Fragmentation

Fragmentation is an important feature of the IP protocol. When a packet is too large to send in its entirety (that is, the packet size exceeds the MTU), the packet is forwarded on as smaller fragments, with the fragmented

Probable brand	**Version**[4]	**Proportion**
Cisco IOS	11.*, 12.*, 15.*	78.25%
Juniper	3.255	7.61%
Not classified	[0-9].*, 21.3, 21.95, 37.90, 60.1, 76.0	13.12%

Table 1: Router version as captured by `mrinfo`

state indicated in the IP header of each fragment. The receiver of the fragments is in charge of reconstructing the whole packet.

IGMP packets may face fragmentation, since they are encapsulated within IP headers. The total size, in terms of bytes, of a `NEIGHBORS_REPLY` message can be computed as follows:

$$|\text{headers}| + \sum_{i=1}^{n} (8 + 4 \times m_i). \qquad (1)$$

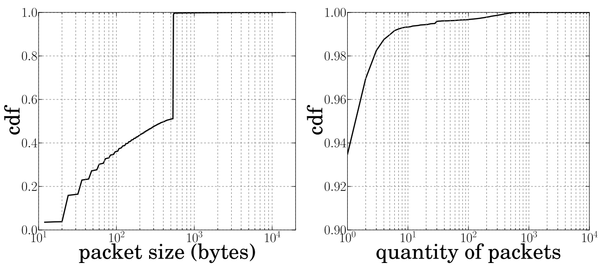
where "header" refers to the sum of the IP and IGMP message headers (20 + 8 bytes), n is the number of local addresses belonging to the router and m_i refers to the number of distant addresses seen through the i^{th} local address. The description of a point-to-point link (i.e., a direct connection between two routers) takes up 12 bytes and consists of the two endpoint IP addresses and several attributes of the local address (the multicast metric, threshold, flags, and the number of distant addresses, which is $m_i = 1$ in this case). In contrast, a point-to-multipoint link (i.e., a broadcast oriented connection involving several routers connected through a layer-2 device) takes up $12 + (m_i - 1) \times 4$ bytes, which includes listing $m_i + 1$ IP addresses.

According to the DVMRP draft [19], the sender should use path MTU discovery to determine whether a DVMRP message must be fragmented. When path MTU is unknown, the Requirements for Internet Hosts (RFC 1122) specifies a maximum packet size of 576 bytes. Note that a `NEIGHBORS_REPLY` message do not contain any port nor query numbers. Therefore, a large IGMP reply should be fragmented in several independent IGMP packets having only the source IP in common.

Depending on their system, we notice that routers manage differently the IGMP `NEIGHBORS_REPLY` fragmentation requirement.[5] Indeed, one interesting feature of `mrinfo` is the possibility to partially fingerprint the OS version of the responding routers and thus study their behavior differences. Table 1 provides an insight of the router brand distribution on the data collected by

[4]The version numbers given here correspond to the DVMRP field version returned by `mrinfo` replies.

[5]Note that this fragmentation issue has been identified by Sharma et al. [20]. However, they do not quantify this issue and do not provide any solution except the use of SNMP.



(a) packets size (b) number of IP packets per reply

Figure 1: Fragmentation details

MERLIN. Obviously, the MERLIN view reflects the market: Cisco routers dominates directly followed by Juniper in a much smaller proportion. The “other brands” (i.e., those we were not able to classify) just represent a little bit more than 13% of the total amount of routers we are able to collect. It is worth to notice that the field “version” returned by responding multicast routers should be the current version of DVMRP and a status flag. Thus a normal answer should be: 3.255, with 3 standing for DVMRP version 3 (the last working version before it becomes obsolete) and 255 depicting a normal situation. However, in practice, routers seems to use their own rules: for example, Cisco routers running IOS simply report the IOS version (IOS 11.*, 12.*, 15.*). Junipers routers seem to report a normal 3.255 answer according to our local tests (but some routers of other brands may also fall into this category). The other observed version numbers are a great source of confusion since they have multiple interpretations. For example, versions 3.* can correspond to Cisco routers running IOS XR but may also correspond to JunOS 3.*. In the same way, there exist too much potential collisions on other version numbers to easily distinguish a device from another.

In responding to `mrinfo` probes, *Juniper* routers (version 3.255) with a large number of connections forge a single large IP packet that is “IP fragmented” by the sending interface. Although this behavior is incorrect according to the IETF draft [19], the `mrinfo` client can handle these large responses since IP fragmentation is transparent to it. In contrast, *Cisco* routers with a large number of connections follow the IETF draft recommendations and reply with multiple independent IP packets, with each packet small enough to avoid “IP fragmentation” (we call this behavior “IGMP fragmentation”). In this case, the `mrinfo` client is not able to deal with the multiple received packets: it only processes the first packet (there is no continuation flag forcing the wait for the remaining fragments). Therefore the initial version of the `mrinfo` client is unable to collect the entire interface list returned by large-degree *Cisco* routers.

The router market being dominated by *Cisco*, this brand is the most common in our dataset (see Table 1). To determine the impact of the IGMP fragmentation, we plot the number of concerned routers and the number of fragmented packets. Fig. 1 details how `mrinfo` results may be impacted by fragmentation. Although, the proportion of routers generating fragmentation is quite low ($\sim$ 6% as shown in the CDF given in Fig. 1(b)), they may generate a great number of fragments (between 10 and 470 in 2% of the cases). Indeed, a small proportion of routers generate almost half of the returned traffic (Fig. 1(a)). The quick growth in Fig. 1(a) corresponds to the bound of 576 bytes: it gives the number of packets belonging to a longer message. In practice, generally, routers generating dozens of IGMP fragments do not report interesting topological information: they mostly always report non-publicly routable addresses. However, the IGMP fragmentation problem reduces the effectiveness of the recursion scheme: even a small amount of missed topological data may hide multicast neighbors potentially containing a large set of neighbors and so on.

Finally, we also investigate the problem of IP fragments filtering. Indeed, some ASes filter (i.e., drop) IP fragments because they may hide DDoS attacks. This filtering could cause the loss of large responses from Juniper routers. We performed a set of experiments in order to detect whether ISPs perform filtering of IP fragments and, if so, where those filtering policies are applied.

Based on the set of source IP addresses collected by MERLIN, we consider a single router per /24. This leads to a set of $\approx$ 28,800 routers. We next ping each of those routers. In $\approx$ 94% of the cases, we obtain a response. On this set of responding routers, we send a fragmented ping (i.e., `ping -s 1500`) in order to force the response fragmentation. We obtain a correct reply in $\approx$ 92% of the cases. This result means that a small proportion of routers seems to filter IP fragments. Based on traceroute, we try to understand how and where the filtering seems to be applied along the forwarding path. In half of the cases, routers accept the first fragment but generate an “ICMP TTL exceeded in reassembly” message meaning that the other fragments are filtered. The second half corresponds to cases where all fragments are either filtered in the forward path or in the return path. Using traceroute, we measure that in the vast majority of the cases, routers only filter fragments that are destined to them but do not perform such filtering on transit IP fragments. This result means that we may not retrieve large Juniper routers performing IP fragmentation even when using several vantage points. Indeed, whatever the reply return path, the filtering will be applied at, or directly around, the target (i.e., by the router itself or by edge routers of the destination/targeted AS if we assume that there exists a common filtering policy within the AS). However, we can argue that potentially missed routers mostly only filter incoming IP fragments. Thus, their incoming filter do not prevent them from generating IP fragmented packets when they receive non-fragmented requests such as `mrinfo` probes. Hence, we can claim that the fragment filtering problem seems marginal according to the small number of potentially impacted routers.

2.3.2 Multiplexing

In this section, we focus on performance issues related to large-scale `mrinfo` campaigns. The initial version of the `mrinfo` client works as follow: first, it sends its IGMP query, then it waits for a possible reply during a given time of t seconds. Possibly, it performs up to n retries if no response has been collected within the previous time frame. If we consider a set of targets consisting of m IP addresses, then the whole process may last $t \times n \times m$ seconds in the worst case. A large-scale run of one million targets ($m = 10^6$) with realistic parameters ($t = 2$ and $n = 2$) could last more than 46 days. It may seem like we only need to run multiple `mrinfo` instances on a single vantage point to reduce the running time. However, IGMP does not use ports or query numbers to multiplex incoming/outgoing connections. Therefore, a single computer having only one IP address should not simultaneously run multiple instances of `mrinfo`. Indeed, each parallel instance of `mrinfo` will treat received responses related to other instances as a reply to its last query leading so to confusion.

There is a seemingly obvious workaround that does not work in practice. Consider a reply r and an `mrinfo` instance i. It is easy to force i to ignore r if r is not directly linked to the last query of i: that is, if the source IP of the reply is not equal to the destination IP of the last query, then the `mrinfo` instance i treats reply r as belonging to another instance. However, in practice, a router can reply with an IP address different than the one queried. For example, a router may use its loopback address, the outgoing interface address, or a configured address. When the responding IP address Y does not match the probed IP address X, the `mrinfo` client reports a warning stating that Y has responded "instead of" X. This "instead of" behavior can be normal and is not rare according to our measurements. Roughly 10% of the replies fall in the "instead of" case, all of them involving Juniper routers. Hence, the workaround of checking the responding address will fail for "instead of" responses, since such a response will be ignored by all `mrinfo` instances (including the instance that elicited it).

Note that another type of "instead of" problem occurs when a response arrives late. If the initial `mrinfo` client running sequentially receives a reply after the response timer expires, then the next query can be falsely associated with this late reply. Thus, trying to reduce the timeout period to speed up the campaign can lead to false associations between targets and responses.

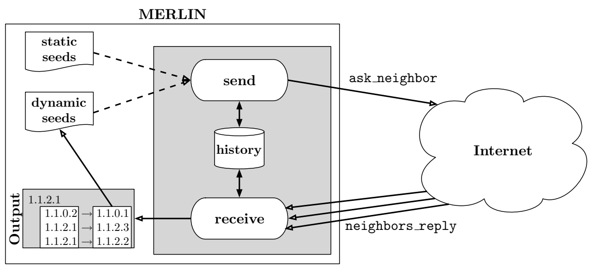


Figure 2: MERLIN architecture

3 MERLIN

As detailed in Sec. 2.3, there does not exist a simple way to fix both issues without impacting the correctness of the probing campaign (except using multihomed vantage points - with multiple IP addresses - or multiplying the number of vantage points). In this section, we describe our new architecture, MERLIN that stands for *MEasure the Router Level of the INternet*. This architecture does not require the use of multiple IP addresses or multiple vantage points while it allows one to fix the issues highlighted in Sec. 2.3. In particular, this new implementation is easily configurable to provide an efficient and network-friendly probing approach: MERLIN minimizes the reprobing risk while it allows one to considerably improve the efficiency of a large scale probing campaign. The basis of the MERLIN architecture is to decouple the sending and receiving processes in order to avoid the use of timers between queries and replies and improve the probing efficiency. With this new scheme, replies are indexed according to the source IP of the reply, so we do not rely on the targeted IP anymore. Furthermore, all replies having the same source IP address are considered as part of a largest message in order to re-assemble IGMP fragmented packets of a given router.

Sec. 3.1 describes in detail the MERLIN architecture while Sec. 3.2 provides some configuration examples and Sec. 3.3 discusses its limitations.

3.1 Architecture

Fig. 2 depicts the MERLIN architecture. The heart of MERLIN is made of two processes: *send*, in charge of sending probes to the network, and *receive*, in charge of processing the replies returned back by routers. These processes are now totally decoupled and the recursion is embedded.

In order to minimize redundancy, the sending process never probes an IP address previously discovered: for efficiency, we use a hash table indexed on local IP addresses of each replying router (the "history" box in Fig. 2). Furthermore, to also minimize the memory consumption, we associate a linked list of IP header and data checksums to each source IP address: a packet is considered as new only if its checksum does not belong to the list of already recorded checksum. It allows one to avoid network duplicates while avoiding, at the same time, dozen of identical messages generated by some end-box routers. Note that MERLIN keeps track of the actual binary reply format such that it is able to differentiate point-to-multipoint links from multiple point-to-point links using the same local address (see `mrinfo` packet format [25]). Moreover, to deal with the IGMP fragmentation issue and again remain light in term of memory consumption, MERLIN uses a timer s to determine when the information related to a given router is ready to be flushed to the output file. If no new fragment associated to a router r has been received during the previous time frame of s seconds, the data structure corresponding to r is freed and the output of r is definitively flushed.

Now, let us describe the basics of the networking processes. The send process is fed by both a static IP address list (called *seeds* on Fig. 2) and a dynamic IP address list obtained from replies. This dynamic list is used for recursion. At the starting of MERLIN, the send process receives IP addresses from the static list. Once replies are collected from the receiving process, the dynamic list is build based on publicly routable IP addresses belonging to the neighbor address list and the recursion is engaged, i.e., the send process gives priority to targets from the dynamic list. Each time the dynamic list is empty (i.e., the current recursion is finished), the send process is again fed with IP addresses from the static list (the initial seeds). Recursion first is a design choice that has been made in order to minimize the probability of reprobing a given router. Moreover, this design choice also ensures to collect a connected part of the probed topology in a short timescale: it allows one to increase the topology consistency in case of topological changes. Indeed, the dynamic of the Internet graph may lead to false connectivity inferences when connected routers are probed in a timescale greater than the one of potential changes.

A key feature of MERLIN is its friendly approach in probing, making it scalable as it avoids reprobing IP addresses previously discovered or already targeted. This is achieved by maintaining information about already processed IP addresses but, also, by slowing down the send process. Indeed, if the time between subsequent probes is too tight, it is very likely to probe the same router many times in case of discovering a highly connected portion of the network. For example, this happen when a pair of routers are connected through multiple logical/physical links or when several routers form a clique. In that case, two or more probes towards the same router can be sent

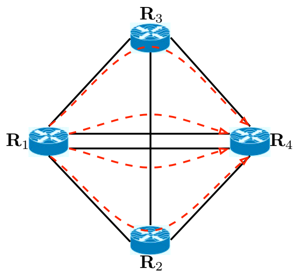


Figure 3: Reprobing risk on R$_4$ - 4 probes may reach it

before receiving its reply. Let us illustrate this situation with Fig. 3: router R$_1$ is able to see R$_4$ through two direct interfaces and is connected to routers R$_2$ and R$_3$. Now, let us imagine that after receiving the interfaces of R$_1$ the send process injects in the network four consecutive probes (within a tight timescale): two towards IP addresses belonging to R$_4$ (it cannot know that those addresses belong to the same router), and two respectively towards R$_2$ and R$_3$. At this step, R$_4$ is already probed two times. Moreover, if its reply is received after the ones of R$_2$ and R$_3$, the recursion will lead to sending two additional probes towards R$_4$ (the ones resulting from R$_2$ and R$_3$ IP neighbors list). This scenario can easily happen if router R$_4$ is slower than R$_2$ and R$_3$ to generate its IGMP response or if forwarding routes fluctuate among those routers. Thus, the only way to prevent routers from that redundant probing is to force waiting a reply using a timer before sending a new request. Sec. 3.2 describes how we calibrate MERLIN to achieve a good tradeoff between an efficient and network friendly probing scheme. Furthermore, note that the topology density (related to highly connected areas) can be exacerbated by the use of VLANs. Indeed, if routers are connected through VLANs, the number of used IP addresses pairs (the logical connections) is greater than the one implied by the physical topology.

The send process of MERLIN considers two probing modes: the *dynamic* and the *static* modes. The dynamic mode occurs with the recursion based on the dynamic list. During this phase, the probe inter-departure time is fixed to a given value α. On the contrary, the static mode corresponds to probing based on the static list. In that case, the inter-departure parameter is fixed to a lower value β: $\beta << \alpha$. To minimize reprobing risk, the sending process prioritizes its treatment tasks as follows: (1) if a new router has been discovered, it marks all its local addresses as already seen, (2) if there exist recursive 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.

Those choices have been made regarding several considerations. The probability that consecutive IP addresses in the static list belongs to the same router is much more lower than for IP addresses belonging to the current dynamic list (so the use of a large timer value is not necessary). Indeed, the static list consists of an unsorted IP address list while the dynamic list is made of IP addresses belonging to a connected part of the Internet. Thus, in the first case, the probability of probing several IP addresses belonging to a given router is really low while, in the second case, it is more likely that consecutive IP addresses in the dynamic list belong to the same router. Furthermore, note that the use of a recursion first approach allows us to reduce the number of duplicate probing when the static list contains consecutive IP addresses belonging to the same router (here the term "consecutive" refers to IP addresses which are probed in a tight timescale). Indeed, our architecture avoids to probe those addresses if they have been already discovered during recursion phases. In practice (see Sec. 3.2.2), most of discovered interfaces are found during this phase.

MERLIN is fully written in C and is freely available on request. It works on Linux and FreeBSD distributions and includes several compilation options to extend its capabilities. For instance, it is possible to force the use of a given IP address for multihomed hosts. It is also possible to forbid the probing and/or indexing of a set of given IP addresses in order to use MERLIN sequentially among a set of vantage points.

3.2 Calibrating MERLIN

3.2.1 Timer

This section experimentally explains our parameter calibration choices. First, we explain how we deal with the fragmentation problem, then we describe how we calibrate MERLIN to perform a good tradeoff between efficiency and friendly approach purpose.

Some routers generate hundred of IGMP fragmented packets just to describe their own interfaces list (see Sec. 2.3.1). To deal with those rare and extreme cases, we need to choose a timer s sufficiently large to ensure the complete response reassembling. In practice, even for routers generating more than hundred of replies (the maximum observed is 470 fragments for a given router), we measure that all responses arrived in the tight timeframe of 0.1 second. However, in order to perform a good tradeoff between CPU and memory use, we decide to set a default timer of $s = 5$ seconds ($s >> 0.1$ to ensure the correct reception of all fragments even with network troubles). Thus, the number of routers flushed in a given timeframe is limited while the number of CPU interruptions remains low and the memory is sufficiently often freed.

In order to investigate the choice of inter-departure parameters, we perform experimental analysis. Fig. 4 has been obtained thanks to our previous `mrinfo-rec` tool. Indeed, we need to link the target IP and the source IP address of a reply. For that purpose, we use a very large timer value (10 seconds) before sending the next probe. We can notice that in the vast majority of the case, responses are returned back to the vantage point in less than 0.5 second: about 99% of replies are collected before the expiration of this timer. Thus, we decide to set $\alpha = 0.5$ second by default to avoid most of the reprobing risk using the recursion mode.

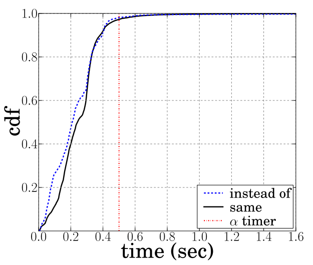


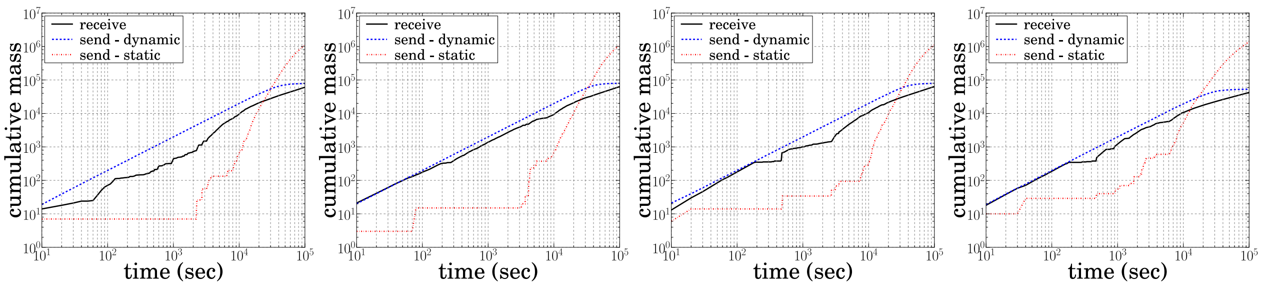
Figure 4: Delay between `mrinfo` probes and replies

The choice of β is made differently because the probability to probe twice or more a given router using the static list is really lower. We decide to set $\beta = 0.05$ second, i.e., at maximum, 20 probes are sent per second. This value offers a good compromise for limiting the rate of the send process while being able to probe more than 1.5M of IPs in less than one day. Sending 20 probes per second, the probability to re-probe the same router in a timescale of 0.5 second is almost insignificant: the probability that two or more IPs - among the 10 IPs probed in 0.5 second - belongs to the same router is really marginal. Indeed, the success rate of the static list decreases over time (because the recursion phase does most of the job - see Sec. 3.2.2 for details) and is, on average, under 2%. Moreover, the probability to find IP addresses belonging to the same router under this low rate is still much more lower because the static list is randomly sorted and the size of a router set of interfaces is really limited compared to the number of IP belonging to the static list (see Fig. 9 in Sec. 4.2.1).

Generally speaking, note that the choice of β and α impacts the duration of the probing campaign. If one chooses to reduce those values to speed up the MERLIN campaign, then one might trigger rate limiter filtering and increase the reprobing risk.

3.2.2 The Power of Recursion

In this section, we study the general behavior of a MERLIN probing campaign. Fig. 5 plots the evolution over time of main MERLIN actions: the number of



(a) San Diego (b) Hamilton (c) Napoli (d) Strasbourg

Figure 5: Dynamic vs. static list - July, 9th 2010

probes sent (from dynamic and static list) and the number of received replies according to the vantage point. Due to space constraints, we only show plots for four vantage points: San Diego, Hamilton, Napoli, and Strasbourg. The horizontal axis gives the time (in seconds) from the starting of the probing until the end. The probing lasted roughly 31 hours and we consider the probing campaign launched on July, 9th 2010. The vertical axis provides the cumulative mass of probes sent and replies received. Finally, it is worth to notice that this a log-log scale, as it highlights more easily the first probing periods.

All those figures reflect the recursion-first nature of MERLIN: during the early moments of a MERLIN measurement campaign, the recursion (i.e., probes sent via the dynamic list) "does the job". Indeed, during the first hour of probing, we can notice that a very low number of static probes are used while the number of received replies is close to the number of recursive probes sent (especially during the first minutes), meaning that we are able to collect large multicast components. However, this success rapidly decreases over time and the use of static probes becomes more and more necessary. After the first hours, the situation completely changes: now, static seeds are often solicited and recursion phases become shorter Thus, after having consumed the largest multicast components retrievable thanks to the target list, MERLIN only finds small sets of isolated routers. Keeping in mind that MERLIN "removes" already discovered IP addresses from the static list, this phenomenon is quite logical: seeds are just used as a new point of departure for recursion but relevant and independent seeds (i.e., those allowing to discover new large connected components) are quickly consumed. In addition, we also observed that during the first hours of the probing period (mainly dependent to the recursion mode), it is likely that each vantage point discovers a common part of the global topology whereas the last hours of the campaign allows them to find isolated and more specific multicast component.

Finally, we also evaluated the average success rate when using seeds coming from the static list versus neighbor IP addresses belonging to the dynamic list. While, on average, we notice that the success rate of the static list is under 2%, the success rate of the dynamic list is greater than 35%. However, it is worth to notice that those results are highly related to our "recursion first" implementation design. On the one hand, the history process strongly reduces the performance of the static list, and on the other hand the recursion mode "steals" responding addresses from the static list.

3.3 Merlin Limitations

This section describes the technical limitations of MERLIN. MERLIN presents numerous advantages compared to the use of `mrinfo` but it also suffers from two problems that follow from its design: handling data from routers with that report IP addresses in common (the anycast addresses problem, Sec. 3.3.1) and determining which probe a response is for if the router replies with a different source address (the decoupling between the targeted and the replying IPs, Sec. 3.3.2). While the first problem may cause false packet assignments or the loss of replies, the second problem prevents us from knowing if the targeted IPs have generated a reply.

3.3.1 Anycast

An anycast address is a publicly routable IP used on different routers. In general, the normal use of such addresses corresponds to a need for reliability and redundancy: it contributes to the robustness of the multicast tree.[6] Most of the time, these addresses are loopback IP addresses that are not used to define a physical connection between routers. In practice, it means that a MERLIN campaign can report two different routers (their list of reported interfaces are different) having the same local IP address that is not related to the physical topology.

[6]We do not consider IP addresses marked as "down" or "disabled" in replies. Indeed, those IP addresses may appear on different routers because they correspond to obsolete configurations and are not used for actual connections. However, some pseudo-anycast IP addresses may also result from false configurations.

Thus, if those routers respond with an anycast IP address, it may impact MERLIN according to two scenarios: if those responses are collected in a tight time scale, i.e., lower than $s = 5$ seconds, the lists of reported interfaces are merged into a single router. Otherwise the second (and possibly following) responses are ignored. Indeed, MERLIN indexes replies according to their responding source addresses instead of using the destination of the probes.

Hopefully, those scenarios occur very rarely: most routers reply through their probed interface. Keeping in mind that the dynamic list corresponds to discovered connections, this list may not generate a problem unless the router responds through its anycast IP address. If the static list contains an anycast IP address, only one of the routers using this IP address will respond according to the forwarding path towards the prefix containing it (i.e., only "instead of" routers may generate this problem). Finally, the rate of the anycast use that we measure on previous `mrinfo-rec` campaigns is very low: less than a dozen of identical multicast IPs appears on several routers compared to the 100,000 IP addresses collected. Hence, we conclude that this problem is really marginal and does not significantly affect MERLIN.

3.3.2 Decoupling

The second problem is due to the fact that using MERLIN, probes and replies are not linked as the send and receive processes are disjoint (see Sec. 3.1). IGMP messages used by MERLIN may be subject to packet loss when routers drop traffic, and there is no underlying acknowledgement mechanism in IGMP. In the original `mrinfo`, it is possible to configure a given number of retries and a timeout between attempts to circumvent the low level nature of the IGMP protocol. With MERLIN, we cannot use that simple mechanism.

A simple way to overcome this limitation would be to launch a new MERLIN instance to reprobe all IP addresses that might have been subject to loss, e.g., the set of IP addresses R resulting from the difference between the probing lists $P = S \bigcup D$ (static and dynamic), and the intersection between the local IP addresses set L found in the first campaign and the probing lists: $R = P \setminus \{L \cap P\}$.

Generally speaking, MERLIN is designed to be driven through a coordinating instance dealing with the multiple vantage points and their associated sets of IP addresses describing their own discovered topology components.

4 Performance Analysis

This section provides statistics and discussions about the MERLIN deployment and the collected data set. As already mentioned in Sec. 2.2, MERLIN has been deployed on six vantage points, three in Europe (Louvain-la-Neuve - Belgium, Napoli - Italy, Strasbourg - France), two in North America (San Diego - USA, Redwood City - USA), and one in Oceania (Hamilton - New Zealand). The static list of seeds is made of 1.6M of seeds as described in Sec. 2.2. Data has been collected with several runs, between July 9th 2010 and July 29th 2010. All data collected has been merged into a single super dataset focusing on relevant and unique information. This dataset gives us 480,000 IP addresses aggregated into almost 50,000 routers scattered in more than 3,000 ASes.

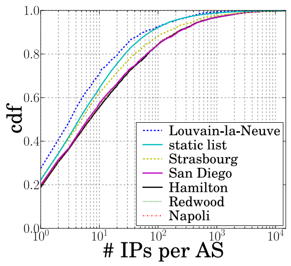


Figure 6: Number of IPs in ASes - July, 25th 2010

Fig. 6 provides some details about the distribution of discovered IP addresses among probed ASes. Most of them come from large Tier-1 ASes (≈ 82%), and their distribution among ASes follows a power law as illustrated in Fig. 6. Generally, most of IP addresses belong to a small subset of well seen ASes while most of discovered AS contain few IPs. When MERLIN discovers less than a dozen of IPs for a given AS, it is likely that it only discovers inter-connection links of this AS, i.e., its boundaries with another AS which is probably more significant in our dataset.

In this section, we first report our efforts to cross-validate the data contained in responses to MERLIN probes. At issue is the frequency of responses that contain addresses that belong to other routers; these addresses might be stale, owing to interfaces being configured with an address that is later shifted to another router, or be anycast addresses. We test the interface addresses returned with *Ally* [11] and *Mercator* [26] probes. Ally infers aliases if a sequence of probes sent to alternating IP addresses yields responses with incrementing, interleaved IP-ID values. Mercator infers aliases when a router responds with a different source address than that probed. More recent tools for alias resolution [12, 27] are more appropriate for constructing a complete router-level graph; Ally lets us carefully probe addresses with a high probability of being aliases without inducing rate limiting.

We tested 41,224 routers; the set consists of routers that reported at least two addresses not inclosed in RFC 1918 prefixes. We were unable to obtain information with Ally or Mercator for 6,135 (14.9%) routers that would allow us to judge the MERLIN response. Of the 35,089 mrinfo routers that we did test, 28,003 (79.8%) were in complete agreement with Ally and/or Mercator techniques. A further 6,747 (16.4%) routers did not have conflicting alias resolution data, but we did not obtain a response for all interfaces. In total, 913 (2.6%) of MERLIN routers had some conflicting alias resolution data.

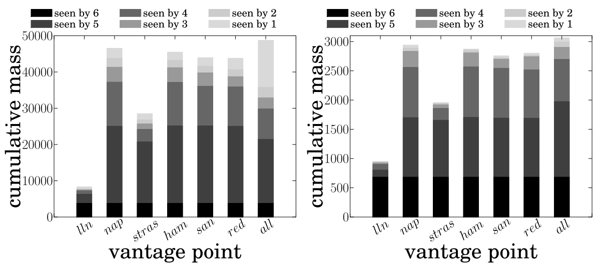
This cross-validation analysis shows us that data collected with MERLIN is highly consistent with results coming from Ally or Mercator. The disagreement cases comes from a combination of Ally's limitations (assuming a shared counter when the counter could be scoped to individual line cards), and assumptions about addresses mapped to a single router (most of those conflicts seem to be due to stale configurations generating pseudo-anycast addresses).

In the following, we investigate the importance of each vantage point (Sec. 4.1). We next evaluate the completeness of the collected dataset (Sec. 4.2).

4.1 Importance of Vantage Point

In this section, we analyze the utility of our set of vantage points. The goal is to emphasize the importance of using several vantage points to avoid IGMP filtering by intermediate networks. A MERLIN probe may be dropped on the forward path, and a IGMP response may also be filtered on the reverse path where the return path differs. Note that there exist two kinds of IGMP filtering behaviors: a multicast router may drop a MERLIN query addressed to it (*local filtering*) or it may drop any MERLIN queries going through it (*transit filtering*). While the local filtering concerns individual routers, transit filtering is more challenging: all requests following a path containing such a filtering router are dropped. In practice, we can distinguish three cases: either a router does not apply IGMP filtering at all, it just applies local filtering, or both local and transit filtering (we assume that cases where routers just apply transit filtering make no sense). Hence, the use of multiple independent vantage points may allow us to increase MERLIN coverage. Indeed, some non filtering routers unreachable via a given vantage point (due to the transit filtering of others) may become reachable through another independent vantage point. More precisely, the term "independent" is related to the AS level graph location of the vantage points: considering a given target r, the more the forwarding paths between the vantage points and r differ, the more likely it is to reduce the impact of IGMP filtering and to increase MERLIN coverage.

Fig. 7(a) shows the utility of each vantage point. For each vantage point, we plot the absolute quantity of routers it discovers, and how many vantage points ob-



(a) routers discovered (b) ASes discovered

Figure 7: Vantage point utility

serve each router. The individual stacks reflect the utility of each vantage point, and the stack labeled "all" refers to the global utility of the union of routers discovered via all vantage points. Interestingly, each vantage point is able to discover between 1,000 and 3,000 unique routers (i.e., they cannot be seen by other vantage points). For the complete set of routers discovered, 30% are discovered by individual vantage points. This proportion is higher than the 15% of routers that belongs to the total intersection ("seen by 6"). This first result highlights the importance of each vantage point: their individual utility cannot be considered as marginal.

Moreover, we can also understand the importance of each vantage point independently. From Fig. 7(a), we notice that the Napoli vantage point is the most efficient, directly followed by the ones in New Zealand, San Diego, and Redwood City. On the contrary, Louvain-la-Neuve and Strasbourg are clearly more subject to IGMP filtering. In all cases, the relative proportion of "seen by n" is roughly uniform among the set of vantage points. It seems that the total number of routers seen through a given vantage point is a sufficient information to understand the importance of a vantage point: each vantage point brings an almost constant number of unique routers while the robustness it provides (routers seen through n points, with $1 < n < 6$) mostly depends on the total of routers it discovers.

We can interpret those results as follows: generally, all vantage points are able to discover routers belonging to Tier-1 ("seen by 6 and 5", because there exist non IGMP filtered paths between the vantage points and Tier-1 ASes). However, their success in probing the network depends on the inter-domain forwarding and filtering policies induced by their providers connectivity. For instance, Strasbourg suffers from the filtering of GEANT while Napoli is able to circumvent it. Thus, although some targeted ASes are close to the vantage points of Strasbourg or Louvain-la-Neuve, they cannot discover them due to the filtering of GEANT while Napoli can. Generally, the further the target, the more likely a filter. Hence, except Strasbourg and Louvain-la-Neuve that are both filtered by one of their main indirect providers (the ones carrying most of their traffic one hop further), the four other vantage points discover a high proportion of common routers while improving the global view due to their ability to better discover the AS graph portion around them.

However, Fig. 7(b) mitigates this first observation. The proportion of AS discovered through only one vantage point is quite low compared to the respective proportion using a per router perspective (Fig. 7(a)). Indeed, the ASes "seen by 1" correspond to small stub or Transit ASes not containing many IPs. This difference means that when a vantage point is able to discover IP addresses belonging to a given AS, it is likely that it will only discover a subset of multicast routers that are `mrinfo` compliant. If one considers the multicast part of an AS as a connected graph, the recursion should allow us to discover this entire graph. However, if some multicast routers do not respond to `mrinfo` (their OS does not activate `mrinfo` capabilities for public users), the recursion may stop facing this wall. Furthermore, even if the static list contains an IP address belonging to the multicast component located at the other side of the wall, the forwarding path used to reach it may be subject to filtering policies. Indeed, considering a given pair (vantage point, AS), the forwarding paths connecting them may be diverse in particular for large AS containing several large prefixes such as Tier-1 and Tier-2 providers. Thus, each vantage point is subject to different filtering policies according to the prefix containing the target.

To conclude, the utility of multiple vantage points using MERLIN is completely different from the one of a tool such as `traceroute` [28]. The utility of using multiple vantage points decreases according to the number of used locations. However, it does not quickly become marginal as each vantage point continues to provide a constant and unique capacity to probe its close environment (Stub and Transit AS within a low number of hops). Further, each vantage point is able to reach prefix subsets of larger AS (Tier-2 and Tier-1 AS) thanks to specific paths allowing it to circumvent IGMP filtering of its other providers. Each vantage point can take benefit of its unique situation in the AS level graph to reach a particular target. In practice, MERLIN should be deployed on several locations well spread around the global AS level graph, and piloted in a way that favor the discovery of new responding routers: multicast neighbors of a new discovered router r may be unreachable through the current active vantage point but may respond using another vantage point not able to reach r.

4.2 Completeness

In this section, we investigate the *completeness* of MERLIN. We understand the completeness of MERLIN on two axes: the *proportion of multicast interfaces* (Sec. 4.2.1), in which we examine the ability of MERLIN to return a complete set of interfaces for a given router, and the *proportion of multicast routers* (Sec. 4.2.2), in which we estimate a lower bound for the proportion of the Internet that is multicast enabled.

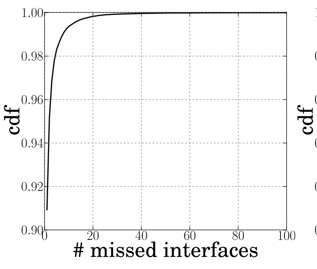
In the following, we assume that a multicast router r reports the same list of interfaces whatever the choice of the targeted IP address as long as it belongs to r. Realistic exceptions may be due to the use of VPN or anycast addresses. If the targeted IP address belongs to the VPN virtual routing table of r, then r only returns the content of the virtual routing table. If the targeted IP address is an anycast address, we cannot predict which router will answer. It depends on the forwarding plane and, consequently, on the vantage point.

Second, a list of multicast interfaces belonging to the same router may be seen by several vantage points and we do not need to keep more than one copy of it, unless the source of the reply is not contained in the list. In this case, we consider the source IP address to be a purely unicast interface belonging to the router. These cases may occur across a single vantage point and are useful to understand the completeness of MERLIN as described in Sec. 4.2.1.

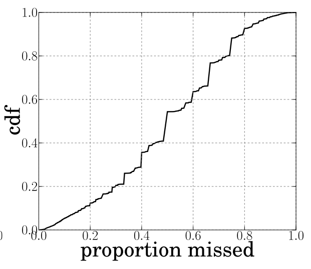
4.2.1 Proportion of multicast interfaces

In this section, we evaluate the ability of MERLIN to return a correct and complete set of interfaces for a given router. Indeed, MERLIN being a multicast tool, by definition, it only reports the multicast interfaces and adjacencies of a given router. In practice, a multicast router can be configured at the interface granularity: each interface can independently support multicast. Furthermore, an ISP may decide to enable multicast only on a given portion of its network. However, if multicast routers do not enable multicast on all their forwarding interfaces, and if the network does not generalize the use of multicast, it may obstruct the multicast tree construction. Indeed, PIM messages generally follow the unicast forwarding plane until the rendez-vous point, and if PIM messages go through a non multicast enabled interface, the multicast tree cannot work properly.

Some exceptions may appear on borders of networks. On the one hand, inside an ISP using a routing protocol such as OSPF, if some routing areas do not require multicast (i.e., there are no multicast clients pending on it), routers do not need to support multicast: only the backbone and the multicast capable areas require it. Thus, an area border router does not need to support multi-



(a) Raw number



(b) Proportion

Figure 8: Interfaces missed for responding routers

cast adjacencies with routers belonging to non multicast area. On the other hand, between Autonomous Systems (ASes), the BGP routing protocol can use specific multicast forwarding entries to disseminate PIM messages. So, it is likely that a multicast AS border router will not enable multicast on all its inter-domain interfaces.

In this section, we try to quantify those missed unicast interfaces. Although MERLIN does not report purely unicast interfaces of the probed router (they do not appear in the interface list), a router can answer via an unicast interface: this IP address is then contained in the source IP field of the response. Furthermore, keeping in mind that most of routers respond through the probed interface, if one probes a purely unicast interface u belonging to a multicast router r, then r is able to answer to MERLIN via u even if it does not support multicast on it. Thus, we are able to provide a lower bound on the quantity of missing interfaces by using source IP addresses (the index of replies) not belonging to the reported multicast alias.[7] The static list containing both multicast and unicast interfaces, and keeping in mind that MERLIN does not avoid the reprobing of a given router indexed on a non reported IP address, we are able to estimate the number of occurrences of such a case. Indeed, if the source u of the reply is not yet indexed but corresponds to an already seen alias, it means that u is a purely unicast interface.

Fig. 8(a) provides the cumulative distribution of the number of missed interfaces per router: they correspond to purely unicast interfaces present in the static list (they are not reported in the multicast alias but we can gather them to it if the router is able to respond through them). In at least 9% of the cases, it seems that MERLIN is not able to collect the entire alias. The largest number of missed interfaces for a single router we faced during our measurements is 88. All interfaces falling in those 9% are unicast interfaces not reported by MERLIN replies in the set of multicast interfaces of a router. If those interfaces were not present in the static list, they would have been missed. Looking at Fig. 8(a), we observe that for most of these cases, less than ten interfaces are missing and can be reported as purely unicast.

[7]This is a lower bound because it is likely that the static list does not contain all interfaces belonging to probed router.

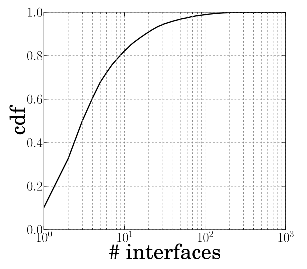


Figure 9: Number of interfaces per router

To better understand the situation, we also plot in Fig. 8(b) the relative proportion of missing interfaces[8], i.e., the number of purely unicast interfaces compared to the total number of IP addresses (both virtually added unicast and reported multicast).

We note that this relative lack is uniformly distributed across the 9% of impacted routers: whatever the level of loss, the occurrence probability remains roughly equal.

4.2.2 Proportion of Multicast Routers

Without having a complete knowledge of the Internet topology, it is difficult to estimate which proportion of the network is multicast enabled (and by extension the subset responding to MERLIN). In this section, we estimate a lower bound of this proportion according to our list of seeds.

Our global static list for seeding MERLIN is made of 1,643,005 IP addresses. Among these targets, 1,223,715 IP addresses come from the Archipelago dataset [5]. We assume that this subset is representative of the active Internet space (e.g., they are well distributed across the Internet). Note that this "hitlist" results from an active traceroute measurement phase allowing to mainly focus on active backbone IP addresses (belonging to routers). Obviously, using an hitlist consisting of randomly chosen IP addresses among the whole Internet address space will not produce equivalent results (see Fan and Heidemann [29] for discussions about passive hitlist efficiency). However, there does not exist any reasons that an hitlist coming from traceroute based measures favors or disfavors the presence of multicast enabled interfaces.

Running a MERLIN campaign specifically targeting those 1,2M IP addresses, we were able to collect responses for 61,988 IP addresses. This number reflects the intersection between the Archipelago seeds and the

[8]Fig. 9 describes the cumulative distribution of the number of interfaces per router: the vertical axis is the cumulative distribution while the horizontal axis, in log-scale, provides the number of interfaces per router. In general (i.e., in 50% of the cases), the routers discovered are pretty small, i.e., less than three active and globally routable interfaces.

set of IP addresses collected by MERLIN. Thus, reported to the Archipelago dataset, both lists share a common subset greater than 5% of the hitlist.

If one considers that the Archipelago hitlist is representative of the whole Internet backbone, one could say that, at least, 5% of the active Internet address space supports multicast and `mrinfo` messages. Indeed, this value can be considered as a lower bound for two reasons: (i) some multicast routers may not respond to `mrinfo` probes, and (ii), even with the use of six vantage points we cannot consider that we avoid all IGMP filtering (Sec. 4.1).

5 Related Work

Compared to traceroute [3] and its variants [4, 5, 6, 30, 11, 7], 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.

The main advantage of MERLIN is that, in a single IGMP reply, a router lists all its multicast interfaces, their IP addresses, and the IP addresses of its neighbor routers. Thus, MERLIN does not suffer from the alias resolution problems affecting traceroute. 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 very small compared to traceroute. Indeed, with MERLIN it is possible to collect the topology of a multicast enabled network by sending a single packet to each router.

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 [16, 6, 31]. Paris Traceroute [4] is able to discover load balancing routers, as well as the set of paths joining those load balancing routers. However, this works mostly for intradomain 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. Furthermore, traceroute based techniques may infer false links if forwarding changes occurs during a trace.

Algorithms based on the *Simple Network Management Protocol* (SNMP) [32] have also been proposed [33, 34]. In some sense, SNMP might be seen as identical to `mrinfo` probing as it allows one to collect information on interfaces directly from the router. It comes with the advantage of not requiring any particular protocol to be deployed. However, the prober must own the SNMP permission on each router.

6 Conclusion

In this paper, we discussed the implementation, the deployment and the validation of MERLIN, a new tool for discovering the Internet topology at the router level. MERLIN, based on a multicast management tool called `mrinfo`, comes with the strong advantage of listing all IPv4 multicast interfaces of a router and its links towards its neighbors. The probing cost associated to `mrinfo` is limited as a single query probe is enough to obtain this information. On the one hand, MERLIN fixes bugs and limitations inherent to `mrinfo`. On the other hand, MERLIN is designed to offer a configurable tradeoff between efficiency and network friendly purposes. The data collected with MERLIN can be used for performing typical topology study [17, 23, 9].

We deployed MERLIN on six machines spread around the world and evaluated its performance. We highlighted the importance of using multiple vantage points in order to circumvent IGMP filtering. Indeed, each vantage point is able to discover a significant portion of unique routers. In addition, we validated and evaluated the completeness of MERLIN: we first perform a cross-validation on reported alias and we investigate the proportion of multicast enabled interfaces and routers in the Internet.

Future work should reveal how we can guide MERLIN vantage points from a coordinating entity in order to improve its coverage and reduce the probing redundancy between vantage points.

References

[1] B. Donnet and T. Friedman, “Internet topology discovery: a survey,” *IEEE Communications Surveys and 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 Communications Surveys and Tutorials*, vol. 10, no. 2, pp. 48–69, April 2008.

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

[4] 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.

[5] 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.

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

[7] 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.

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

[9] 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.

[10] R. Braden, “Requirements for Internet hosts – communication layers,” Internet Engineering Task Force, RFC 1122, October 1989.

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

[12] 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.

[13] M. H. Gunes and K. Sarac, “Resolving IP aliases in building traceroute-based internet maps,” *IEEE/ACM Transactions on Networking (ToN)*, 2009, to appear.

[14] 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.

[15] M. H. Gunes and K. Sarac, “Importance of IP alias resolution in sampling Internet topologies,” in *Proc. IEEE Global Internet Symposium*, May 2007.

[16] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, “Deployment of an algorithm for large-scale topology discovery,” *IEEE Journal on Selected Areas in Communications (JSAC), Sampling the Internet: Techniques and Applications*, vol. 24, no. 12, pp. 2210–2220, Dec. 2006.

[17] 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.

[18] D. Dugal, C. Pignatoro, and R. Dunn, “Protecting the router control plane,” Internet Engineering Task Force, Internet Draft (Work in Progress) draft-ietf-opsec-protect-control-plane-03, August 2010.

[19] 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.

[20] 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.

[21] V. Jacobson, “`mrinfo` ,” 1995, see http://cvsweb.netbsd.org/bsdweb.cgi/src/usr.sbin/mrinfo/?only_with_tag=MAIN.

[22] S. Deering, “Host extensions for IP multicasting,” Internet Engineering Task Force, RFC 1112, August 1989.

[23] 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.

[24] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. van Wesep, T. Anderson, and A. Krishnamurthy, “Reverse traceroute,” in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, June 2010.

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

[26] R. Govindan and H. Tangmunarunkit, “Heuristics for internet map discovery,” in *Proc. IEEE INFOCOM*, March 2000.

[27] K. Keys, Y. Hyun, and M. Luckie, “Internet-scale alias resolution with MIDAR,” February 2010, iSMA Workshop on Active Internet Measurements (AIMS).

[28] P. Barford, A. Bestavros, J. Byers, and M. Crovella, “On the marginal utility of network topology measurements,” in *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)*, Nov. 2001.

[29] X. Fan and J. Heidemann, “Selecting representative IP addresses for Internet topology studies,” in *Proc. ACM/USENIX Internet Measurement Conference (IMC)*, November 2010.

[30] N. Spring, D. Wetherall, and T. Anderson, “Scriptroute: A public internet measurement facility,” in *Proc. USENIX USITS*, March 2002.

[31] 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.

[32] J. D. Case, M. Fedor, M. Schoffstall, and J. Davin, “Simple network management protocol (SNMP),” Internet Engineering Task Force, RFC 1157, May 1990.

[33] S. Avallone, S. D’Antonio, M. Esposito, A. Pescape, and S. P. Romano, “A topology discovery module based on a hybrid methodology,” in *Proc. Inter-Domain Performance and Simulation Workshop (IPS)*, March 2004.

[34] R. Siamwalla, R. Sharma, and S. Keshav, “Discovering internet topology,” Cornell University, Ithaca, NY 14853, Tech. Rep., July 1998.