Latency-Based Anycast Geolocation: Algorithms, Software and Datasets

Extended Technical Report (January 18th, 2016)

Danilo Cicalese*, Diana Joumblatt*, Dario Rossi*, Marc-Olivier Buob†, Jordan Augé†, Timur Friedman†
*Telecom ParisTech, Paris, France – first.last@telecom-paristech.fr
†UPMC Sorbonne Universités, Paris, France – first.last@lip6.fr

***Abstract*—Use of IP-layer anycast has increased in the last few years: once relegated to root and top-level DNS servers, anycast is now commonly used by Content Distribution Networks. However, existing measurement techniques to identify and enumerate anycast replicas exploit specifics of the DNS protocol, which limits their applicability to this particular service. With this paper, we not only propose and thoroughly validate a protocol-agnostic technique for anycast replicas discovery and geolocation, but also provide the community with open source software and datasets to replicate our experimental results, as well as facilitating the development of new techniques such as ours.**

In particular, our proposed method achieves thorough enumeration and city-level geolocalization of anycast instances from a set of known vantage points. The algorithm features an iterative workflow, pipelining enumeration (an optimization problem using latency as input) and geolocalization (a classification problem using side channel information such as city population) of anycast replicas. Results of a thorough validation campaign show our algorithm to be robust to measurement noise, and very lightweight as it requires only a handful of latency measurements.

We implement the technique in an open source software that is able to process previous measurement campaign, as well as conduct new measurement campaigns (from RIPE Atlas). We also release datasets of our thorough validation campaign, that we carried out from multiple measurement infrastructures (PlanetLab, RIPE Atalas), targeting several anycast destinations (including DNS and CDN servers). We complement such datasets not only with public available information concerning location of DNS and CDN servers at time of measurements, but also with measurement ground truth obtained exploiting DNS/CDN protocol specific information.

The framework composed of the software, dataset and ground truth we release, provides both a useful service for pratictioners, as well as a complete test suite for researchers. The former can readily use our proposed technique without having to worry about deployment of multiple vantage points, whereas the latter can focus their effort on the proposal of new techniques for anycast geolocation, directly comparing results to ground truth and to other techniques such as ours.

I. Introduction

IP-layer anycast [1] allows a group of replicas to offer the same service using a shared IP address from geographically distinct locations around the globe. Inter-domain routing directs the traffic destined to an anycast address to the topologically closest replica (in BGP terms). Anycast is an appealing solution as it is very simple to deploy (i.e., avoiding the need to manage some custom and complex application-layer solution), provides users with enhanced quality of experience (e.g., for services where we want to cache content close to the user), while also providing several advantages for the service provider (i.e., load balancing among replicas, robust to DDoS attacks in reason of geographic traffic confinement, etc.). Therefore, many important Internet services [2] use anycast nowadays, to reduce response times and mitigate the effects of server failure and denial of service attacks. While historically IP anycast has been mostly relegated to DNS, we observe an increasing tendency of anycast CDN services: e.g., EdgeCast [3] and CloudFlare [4], that advertise to serve respectively 1.5 billion objects per hour representing the 4% of the whole Internet traffic[1] and over 2 million Web sites[2], are good examples well testifying this trend.

With this rise of anycast usage, there is a concomitant need to understand anycast services [5]–[20]. ISP providers have a large commercial stake in handling traffic, and for efficient operation they need to where these flows are directed to – which is more ambiguous in the case of anycast than with unicast. Businesses that rely upon services that are delivered via anycast need adequate troubleshooting tools. The locations from which services are provided are of interest to scientists in a broad span of fields. Unfortunately, the publicly information available regarding anycast replicas placement is often outdated [6], if available at all [5], which raises the need for tools capable of automatic, service-independent, lightweight *detection, enumeration and geolocation* of anycast replicas placement. Yet the current state of the art consists of techniques that are either limited to detection [5] or enumeration [6], but are not capable of replicas geolocation. Additionally, most of the work on anycast [6]–[11], including enumeration techniques such as [6], leverage DNS protocol

[1] http://www.edgecast.com/company/news/edgecast-continued-growth
[2] https://www.cloudflare.com/features-cdn

information for the enumeration, so that they fail with any other popular anycast services such as, e.g., CDN.

We recently proposed iGreedy [21], a service-independent techniques able to detect, enumerate and geolocate anycast replicas based on a handful of latency measurement from distributed vantage points. This work extends [21] in several ways, not only by a thorough validation of the technique itself, but also providing the community with open source software and datasets to replicate our experimental results, as well as facilitating the development and validation of new techniques. Especially, at [22] we provide the community with:

- the *iGreedy technique* for lightweight service-agnostic anycast discovery, capable of accurately enumerate (>75% recall) and geolocate (>75% true positive geolocation) replicas with a handful of latency measurements;
- its *thorough validation*, using multiple targets pertaining to different services (DNS and CDN) from two measurement infrastructures (RIPE Atlas and PlanetLab);
- an *open-source implementation* of the technique, able to operate on offline datasets, as well as to generate new datasets (from RIPE Atlas);
- a *simple environment* to visualize on a map the results of detection and classification algorithms, including but not limited to our, and compare them against ground truth;
- *ground truth* that we have assembled regarding DNS (via standard CHAOS queries) and CDN (via inspections of HTTP headers, a novel contribution on its own);
- a *dataset* comprising exhaustive latency measurements from two measurement infrastructures (RIPE Atlas, PlanetLab) towards anycast addresses in our ground truth.

With this "test suite"[3], practitioners are equipped with a tool to unveil and troubleshoot anycast replicas they possibly encounter in their daily operations. Similarly, researchers can develop other anycast identification algorithms, testing whether they are better than iGreedy at, with low overhead, determining whether an IP address is anycast, how many replicas stand behind the address, where they are located, etc.

The remainder of this paper is organized as follows. We first overview the current state of the art in Sec. II and specify our objectives in Sec. III. We next describe our dataset, including the ground-truth in Sec. IV. Our own solution of the anycast geolocation problem, namely iGreedy, is described in Sec. V. Results based on the dataset and software that we release at [22] are shown a glance in Sec. VI and completed by a thorough sensitivity analysis in Sec. VII. Examples of applications of our technique, each having a high practical interest, are given in Sec. VIII, after which conclusive remarks are gathered in Sec. IX. Finally, the Appendix reports example of usage of the tool, which has since been extended to perform live measurement from RIPE Atlas.

[3]It is worth to point out that code and datasets are available in the same tarball at this direct link [23]

II. State of the art

Anycast server enumeration and geolocalization is part of a broader effort from the research community to geographically map the Internet infrastructure and identify the various components of the physical Internet [24]. While latency-based unicast geolocation [25], [26] is well studied, the same does not hold for anycast, where triangulation technique locating unicast instances at the intersection of several measurement do not apply. We now review the state of the art, organized along several sub-categories related to our work.

A. Unicast geolocation

Unicast geolocation is a well investigated research topic: numerous techniques based on latency measurements [25]–[27] and databases [28], [29] have been proposed for unicast geolocation. Yet, database techniques are not only unreliable with unicast [29], but also with anycast, since they advertise a single geolocation per IP. Similarly, latency-based techniques [25]–[27] use triangulation, and geolocate unicast addresses at the *intersection* of multiple latency measurements from geographically dispersed vantage points. However, this assumption no longer necessarily holds for anycast as we will illustrate in the next section.

B. Unicast infrastructure mapping

Unicast infrastructure mapping studies, the last in line being [18], [30], leverage EDNS-client-subnet (ECS) extension requests to geolocate servers: (millions of) requests are sent with different client IPs from one VP to unveil (thousands of) unicast IP addresses corresponding to PoPs of major over-the-top operator. However, ECS support is becoming widespread to enhance the user online experience but is not yet pervasive, and additionally ECS technique fails with anycast (notice indeed that [18] reports only a single location for Edgecast, which instead has a few tens of PoPs, geographically dispersed worldwide).

C. Anycast characterisation

Research on anycast has so far prevalently focused either on architectural modifications [10], [14], [31], [32] or on the characterization of existing anycast deployments, which is close to our work and that we compactly summarize in Table I. Overall, a large fraction of these studies quantify the performance of anycast in current IP anycast deployments in terms of metrics such as proximity [8], [10]–[12], [33], affinity [7], [8], [10]–[13], [34], availability [8], [9], [11], [13], and load-balancing [11]. Interestingly, while the body of historical work targets DNS, more recent work [13], [14] has tackled investigation of anycast CDN performance (e.g., client-server affinity and anycast prefix availability for the CacheFly CDN).

TABLE I
STATE OF THE ART IN ANYCAST MEASUREMENT RESEARCH

	[5]	[6]	*This work*	[7]	[8]	[9]	[10]	[11]	[12]	[13]
Platform(#VPs)	Renesys monitors	PL (238), Netalyzr (62k), rDNS (300k)	*PL (300), RIPE (6k)*	End-hosts $O(100)$	PL (300)	DNSMON (77)	PL (129)	rDNS (20K)	C,F,K root	Renesys monitors
Technique	BGP vs. traceroute	DNS CHAOS +traceroute	*Latency probes*	DNS CHAOS	DNS CHAOS	DNS CHAOS +BGP	DNS CHAOS	DNS queries	pcap	BGP
Targets	IPv4 prefixes	F root, TLDs, AS112	*F,I,K,L root, EdgeCast CloufFlare CDNs*	C,F-K,M root	B,F,K root, TLDs	C,F-K,M root	C,F-K,M root, AS112	F,J root, AS112		1 CacheFly prefix
Detect	✓	✓	✓							
Enumerate		✓	✓							
Geolocalize			✓							
Proximity					✓		✓	✓	✓	
Affinity				✓	✓		✓	✓	✓	✓
Availability					✓	✓		✓		✓
Loadbalance								✓		

D. Anycast infrastructure mapping

Fewer techniques instead exists that allow to detect, enumerate or geolocate anycast replicas, and that are thus closest to this work. In terms of methodologies, as reported summarized in Table I, anycast infrastructure mapping has so far employed the following measurement techniques: (i) issuing DNS queries of special class (CHAOS), type (TXT), and name (host-name.bind or id.server [35]), (ii) BGP feeds, and finally active (iii) traceroute or (iv) ICMP latency measurement. Specifically, [6] employs (i) and (iii), while [5] leverages (ii) and (iii) and our proposal [21] uses exclusively (iv).

E. Anycast detection

In particular, *detection of anycast prefixes* is tackled in [5] by leveraging latency measurement (from distributed traceroute agents) and BGP routing information (from looking glass and public routers). BGP information is helpful since when the transit tree of an IP prefix has multiple domestic ISPs that are located in disjoint geographic locations, this IP prefix is likely anycasted. Latency measurement complement this information inferring if an IP prefix is anycast by detecting speed-of-light violations, as in this work. Techniques used in [5] have the merit of being protocol-independent, yet they do not to enumerate (let alone to geolocate) replicas as we do in this work.

F. Anycast enumeration

Protocol specific information is instead leveraged in [6] to *enumerate DNS anycast replicas.* As pointed out in [6], unfortunately not even all anycasted DNS servers reply with their identifier to CHAOS-class queries, and in case they do, the replies do not always follow a common naming standard. Additionally, [6] show that in-path proxies may modify such replies and therefore propose two new techniques to distinguish among servers within an anycast group. These techniques consist in either augmenting DNS CHAOS TXT queries with traceroute or modifying existing anycast servers to reply to special DNS IN TXT queries. Despite being a valuable tool for the domain name system, the main drawback of DNS-based technique is their narrow field of applicability with respect to [5], [21].

G. Anycast geolocation

Finally, with the exception of our iGreedy proposal [21] that this work extends, *no other technique exists that is capable of lightweight and protocol-independent anycast replicas geolocation.* The technique is lightweight as it relies on a handful of latency measurement, and is reliable and robust in spite of noisy measurement: indeed, latency measurement are used for detection and enumeration purposes, whereas geolocalization depends on reliable side-channel information (e.g., such as city population).

III. PROBLEM DEFINITION

Our open source test suite allows anyone not only to (i) perform anycast measurement with the iGreedy technique, that represents the current state of the art, but also to (ii) try out his or her anycast identification algorithm on ground truth data that we have established, and compare the performance of their algorithm against iGreedy. This section frames the problem at high level, overviewing at the input available to these algorithms, and presenting useful design guidelines to advance the state of the art. We defer details about the dataset (Sec. IV), our proposed algorithm (Sec. V), its performance (Sec. VI) and sensitivity (Sec. VII) to later sections.

A. Latency-based algorithms

The algorithms that can be evaluated in our framework suite are ones that solve the problem using latency-based measurements. As depicted in Fig. 1(a), for any application of the algorithm there will be some number M of measurement agents launching latency measurement towards each of the target addresses (e.g., with ICMP ping or other protocols). Details about the measurement infrastructure and campaigns are reported in Sec. IV-A and Sec. IV-B respectively, while their impact of iGreedy performance is the object of Sec. VII-C.

Measurement agents are located at a different vantage point somewhere in the world, at a known (and reliable) position expressed as latitude and longitude $(\mathrm{lat}, \mathrm{lon})$. As early introduced, the anycast identification problem consists of three subproblems: (i) given a target IP address, t, *detect* if it is anycasted, (ii) *enumerate* the replicas that are offering the anycasted service, and (iii) *geolocalize* those replicas. We now separately consider each subproblem.

B. The anycast detection subproblem

We assume that any latency-based detection algorithm will generate a number $2 \leqslant N \leqslant M$ of disks for a given target IP address t. Each disk is a circle that is centered on a vantage point in which, according to speed of light calculations, t must lie. If there is any pair of disks that do not overlap, this is proof that t is an anycast address, as illustrated in Fig. 1(b) and introduced more formally in Sec. V-A. On the other hand, if there is a unique area in the world on which all of the disks overlap then, while t *might* be anycast, we cannot prove this with the evidence at hand and so we assume t to be unicast.

For a given address that truly is anycast, loosely speaking the more disks that are generated, the more chances there are to correctly determine this fact. The choice of vantage points (their locations and the spacing between them) will also matter. It might be that for a given set of vantage points, even conducting measurements from all of the vantage points will not be sufficient to determine that some anycast addresses are such (e.g., when vantage points are few and close, or when the latency noise is large so that all disks overlap). But if there are non-overlapping disks, it suffices to correctly choose two vantage points in order to make the detection. A similar technique is employed by [5] to detect anycast replicas.

C. The anycast replica enumeration subproblem

Intuitively, if the observation of a pair of non-overlapping disks allows to detect an address as anycast, the observation of several disks that do not overlap among them allows to further enumerate distinct replicas. Given N disks, there are multiple ways to choose a set of $K \leqslant N$ non-overlapping disks such that the addition of any of the disks outside of this set would result in an overlap. Our enumeration technique is discussed in Sec. V-B

Ultimately, the enumeration problem is better framed in terms of an *optimization problem*, in which an optimal solution consists in identifying the largest number (all if possible) of replicas. The enumeration subproblem may use the same disks as used for the detection problem (where only a pair suffices to trigger detection), or additional disks (aiming for a full enumeration). At a minimum, a number of disks equal to the number of anycast replicas is required in order to enumerate all of them. In practice, it might not be possible to enumerate all replicas depending on the set of vantage points available in the measurement infrastructure. Sensitivity to the measurement infrastructure is assessed in Sec. VII-C.

D. The anycast replica geolocalization subproblem

For each of the K non-overlapping disks generated in the previous step, a replica must lie somewhere within that disk. The geolocalization problem can thus be thought as a *classification problem*, in which we select, from a set of discrete locations within each disk, the most likely position of the anycast replica in that disk. Of course, in this stage not only latency measurement, but also any pertinent information can be leveraged by algorithms, such as, for instance, known landmass areas (replicas are unlikely to be out at sea) and locations of major metropolitan areas (replicas might be situated close to these, in order to promote low latencies to large numbers of users).

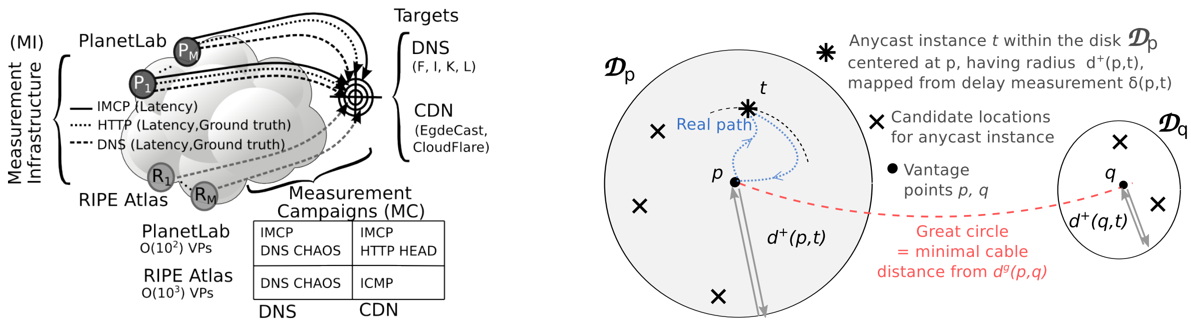
In Fig. 1(b) such discrete locations are indicated with crosses within the disk: we discuss how we build a reliable ground truth for the position of such crosses in Sec. IV-E. Our geolocation technique is described in Sec. V-C and a sensitivity of its performance is presented in Sec. VII-A.

E. Beyond the state of the art

Desirable properties of algorithms outlined above can be summarized as (i) reliable detection, (ii) complete enumeration, (iii) accurate geolocation, (iv) protocol independence and (v) low-overhead. Properties (i)-(iii) are specific to each algorithm, and this paper illustrates especially iGreedy enumeration and geolocation performance. The remaining properties are instead intrinsic to our problem formulation, since algorithms are given just (iv) RTT latency measurement (v) in the order of 100-1000 vantage points.

Notice especially that the (iv) protocol independence requirement suggests the use of ICMP to obtain latency samples. Indeed, given that no a priori information on the service running on an anycast host can be assumed, soliciting response on specific transport layer ports (e.g., UDP 53 for DNS or TCP 80 for CDN) would likely only obtain service-specific response (i.e., conditioned to the availability of that anycast service on the target under test). Conversely, ICMP based latency measurement are not affected by this per-service bias. We further discuss quality of latency samples in Sec. IV-B and the impact that latency noise has on iGreedy in Sec. VII-C.

Finally, in terms of (v) overhead, we remark that the amount of probe traffic in our datasets is much lower to what considered in recent studies employing from 20k vantage points [11],


(MI)
PlanetLab
Measurement Infrastructure
IMCP (Latency)
HTTP (Latency,Ground truth)
DNS (Latency,Ground truth)
RIPE Atlas
Targets
DNS
(F, I, K, L)
CDN
(EgdeCast, CloudFlare)
Measurement Campaigns (MC)
PlanetLab O(10^2) VPs
RIPE Atlas O(10^3) VPs
IMCP DNS CHAOS
IMCP HTTP HEAD
DNS CHAOS
ICMP
DNS
CDN
Anycast instance t within the disk $\mathcal{D}_p$ centered at p, having radius $d^+(p,t)$, mapped from delay measurement δ(p,t)
Candidate locations for anycast instance
Vantage points p, q
Real path
$d^+(p,t)$
$d^+(q,t)$
Great circle = minimal cable distance from $d^g(p,q)$
$\mathcal{D}_p$
$\mathcal{D}_q$

(a) (b)

Fig. 1. Synoptic of (a) anycast measurement scenario and (b) anycast instance detection via latency measurements

to soliciting responses from about to 300k recursive DNS resolvers plus 60k Netalyzr datapoints [6]. In our framework, algorithms employs as few as 1/100 of the Netalyzr (or 1/1000 of the recursive DNS) data points: while challenging, our results show that fairly complete enumeration and correct geolocation are achievable even with few latency samples.

IV. Datasets

As summarized in Fig.1(a), we run a number of *measurement campaigns* (MC) to provide latency measurement from a relatively low number (hundreds to thousands) of agents situated at different vantage points around the world. Location and number of agents depend on the *measurement infrastructure* (MI) used in the campaign. Since our MCs target known DNS and CDN anycast services, we can build reliable *ground truth* (GT) using protocol-specific information. In particular, building a ground truth for the anycast CDN service is, to the best of our knowledge, a novel contribution on its own – especially, since GT is highly more valuable with respect to *publicly available information* (PAI).

This section discusses the MI, MC, PAI and GT datasets that we provide in our test suite, and that we used to obtain iGreedy results discussed in the reminder of the paper.

A. Measurement infrastructures (MI)

A fairly large number of *measurement infrastructures* (MIs) exist in the current Internet. As the techniques we discuss in this paper can be applied from any such MI, it is thus interesting to analyze the MI candidates, which we list in Tab. II. Given our aim, we put special emphasis on the MI footprint and coverage, and additionally annotate with metadata concerning the geolocation accuracy of individual *vantage points* (VPs).

The table reports a number of open (e.g., RIPE Atlas, PlanetLab, Archipelago, MLab, etc.) and some proprietary (e.g., Dasu, SamKnows, etc.) measurement infrastructures. Due to obsolescence[4] of such information, we prefer to "round" numbers collected from the respective pages, to imply that only orders of magnitude are (at time of writing) or will be accurate (at time of reading). Notice also that while in this work we are mostly concerned with geolocation accuracy of MI vantage points, a recent and more complete survey of measurement infrastructures is also available for the interested reader [39].

The geolocalization of each vantage point in such MIs is typically reported by the person who hosts the measurement agent, and these reports can be verified through unicast geolocalization services [25]–[29], to check for initial accuracy and to catch cases when an agent is moved to another location. For instance, PlanetLab Europe performs an additional validation of the VP location with Spotter [37]. Geolocalization of RIPE Atlas VPs is checked through MaxMind [36], and tags additionally report information about the accuracy of geolocalization for (a growing number of) VPs. In few cases we spotted inconsistent location of VPs, that we validate via manual inspection[5].

In this work, we use RIPE Atlas [40] and PlanetLab [41] which are interesting due to their complementarity aspects. On the one hand, RIPE Atlas has a larger footprint in terms of number of VP (over 20 times larger than PlanetLab) as well as a better geographical (10 times) or AS (5 times) coverage, from which we expect RIPE Atlas to provide a more exhaustive coverage than PlanetLab. On the other hand, RIPE Atlas but is more constrained than PlanetLab in the type and rate of measurement that can be performed: for instance, no HTTP measurement are allowed from RIPE Atlas, limiting the space of actions in the CDN case (see later Sec. IV-E). While the use of even further open MI (such as MLab, Archipelago, etc.) would be of course interesting, we remark these platform to have a comparatively smaller footprint than PlanetLab and RIPE Atlas, which thus already constitute an interesting and relevant starting point. The impact of MI on iGreedy is assessed in Sec. VII-C.

[4]Quoting CAIDA's Measurement Infrastructure Comparison Webpage [38] "*This list was compiled in 2004 and is no longer being maintained. This page is made available for historical purposes*"

[5]As a side effect of this work, we contributed to correct the geolocalization of some of them contacting the infrastructure maintainers. Annotations about inconsistent vantage points locations are available at [22]

TABLE II
INTERNET MEASUREMENT INFRASTRUCTURES: FOOTPRINT AND GEOLOCATION META-DATA

Infrastructure	**Footprint and coverage**			**Geolocation Metadata**				**Used in**
	VP	**AS**	**Country**	**Information**	**Granularity**	**Accuracy info**	**Validation**	**this work**[†]
RIPE	6k	2k	150	(lat,lon)	Individual VP	14% of the VP	MaxMind [36]	✓
PlanetLab	250	180	30	(lat,lon)	Individual VP	n.a.	Spotter [37] (for EU)	✓
Archipelago	100	100	40 (1/3 in US)	City-level	Individual VP	n.a.	Oral communication	⋆
MLab	1.5k	35	30 (1/2 in US)	(lat,lon)	Individual VP	n.a.	Single owner	⋆
Dasu	100k	1k	150	Continent	Aggregated	n.a.	n.a.	✗
SamKnows-FCC'13	190k			US State	Aggregated	n.a.	n.a.	✗
SamKnows-OFCOM'14	2k			Rural vs Residential	Individual VP	n.a.	n.a.	✗

†Legend: used in this work (✓); open so possible in principle (⋆); proprietary so not possible (✗)

B. Measurement campaigns (MC)

As illustrated in Fig. 1(a) from the RIPE Atlas and PlanetLab MIs, we perform several *measurement campaigns* (MCs) destined to multiple target IP addresses, representative of different services. Specifically we target DNS root servers F, I, K and L and additionally two representative addresses of the EdgeCast and CloudFlare CDN.

For each vantage point p and target t, what can be easily measured is the round trip time delay $RTT_i(p,t)$ that the i-th packet sample took to travel from p to the closest instance of t and back to p. As algorithms require the one-way propagation delay, we estimate it as $\delta_i(p,t) = RTT_i(p,t)/2$: halving the round trip time, we make the worst case assumption of maximal distance from the vantage point (since forward and backward paths are not necessarily symmetric).

It is also well known that measured latency samples can vary from packet to packet, due to different paths [42], queuing delay [43], protocols [44] or even flow-id for the same protocol [44]. To partly compensate for these potential sources of bias we (i) use a minimum operation $\delta(p,t) = min_i RTT_i(p,t)/2$ over multiple P samples to removing noise due to variable RTT components (e.g., queuing delay) and (ii) use RTT samples gathered from different protocols (e.g., ICMP, DNS/UDP and HTTP/TCP). Concerning the latter point, we leverage different campaigns over different application layer protocols, such as DNS and HTTP, which are anyway needed to build the ground truth described in Sec. IV-E. In the case of DNS, $RTT_i(p,t)$ samples include the response time of DNS servers (yet another small but variable component that the minimum operation attempts at filtering). In the case of HTTP, $RTT_i(p,t)$ represent the TCP three-way handshake time.

We shall see in Sec. VII-C that the number of P measurement, or the protocol they are gathered with have no noticeable impact on the performance of the algorithm we propose. To understand why this happens, it is worth recalling that (i) at the speed-of-light, packets travel about 100Km in 1ms and that (ii) recent measurement work [43] has shown that access links can queue several seconds worth of delay, well in excess of the Earth to Moon distance [45]. It follows that geolocation algorithms must cope with noise (due to protocol bias or individual samples variability) as otherwise even slight inaccuracies of the latency estimation can translate into fairly large errors for the geolocation problem. iGreedy thus prefer to leverage side-channel information (such as city population) to factor out latency measurement noise.

C. Publicly available information (PAI)

Publicly available information (PAI) about our target IP addresses is generally available through some Website. While PAI is of course valuable, it however adds additional challenges and ambiguities. In some cases, PAI *comprises* a larger set of replicas with respect to those actually visible from the VPs, which happens for instance in countries with low MI vantage point densities (e.g., China and African continent). In this case, discrepancies between an algorithm results and the PAI are tied to the measurement infrastructure, as opposite to the algorithm. Considering the PAI as reliable would in this case mistakingly increase the amount of False Negative classifications for the algorithm.

In other cases, the opposite is true: i.e., PAI comprises a *smaller* set with respect to the set of replicas actually seen from the VP, which happens whenever the Webpage content is outdated. One example is worth making to anecdotally assess the amount of discrepancy between PAI and measurement in the case of DNS root servers. DNS operators maintain an official website [46] with maps annotated with the number and geographic distribution of deployed sites around the world. According to [6], in 2013 PAI of root server E was advertising a single (unicast) location, despite their DNS state of the art method was able to enumerate 9 distinct locations. In 2014, at the time of our [21] experiments 12 anycast locations were advertised, but our PlanetLab and RIPE measurements were able to collectively discover over 40 replicas. Considering the PAI as reliable would in this case mistakingly increase the amount of False Positive classifications for the algorithm. A telling example of the manual validation concerns root server L, advertising a replica at SGW, which corresponds to an airport in Canada. However, this (spurious) instance was incoherent with the measurement from over 100 vantage points, that were (correctly) locating the vantage point in Singapore. PAI information do not report any airport in Canada

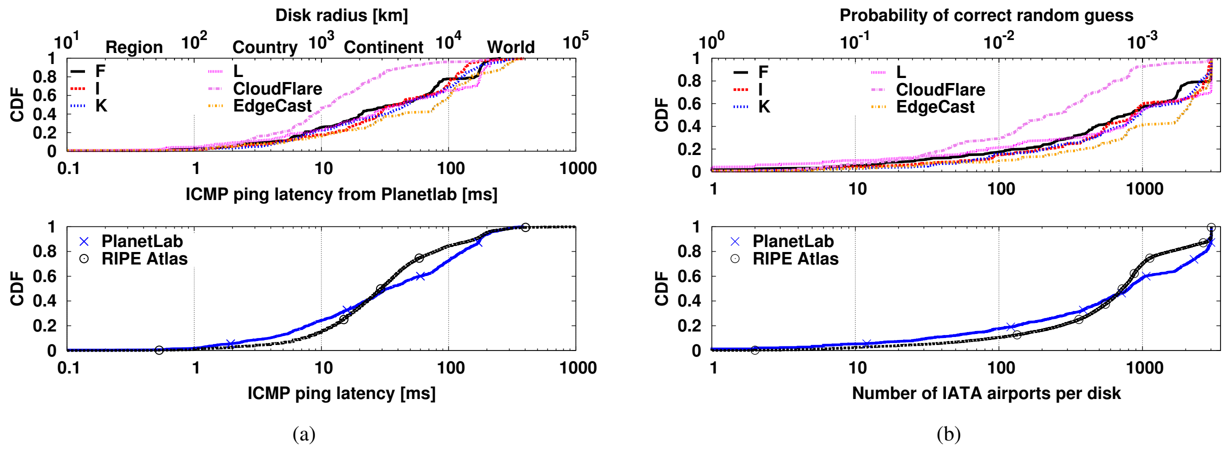


Fig. 2. Dataset statistics: Per-target and Per-infrastructure of (a) Cumulative distribution function (CDF) of minimum latency over all protocols and (b) Complementary CDF (CCDF) of the number of IATA airports in each disk

but does in Singapore, confirming the hypothesis that protocol specific information is configured by humans and still possibly subject by errors, although unfrequented they can be.

These conflicting situations cannot of course be determined by solely relying on PAI and rather call for more accurate alternative methods.

D. *Ground truth (GT)*

For each target IP address in our dataset, we provide geolocation *ground truth* (GT), which is non ambiguous and solves the aforementioned issues with PAI. GT is assembled by (i) performing additional experiments that exploit protocol specific information and (ii) manually validating this new information against PAI and latency measurement. In the case of IP addresses of root DNS servers, we use DNS CHAOS requests as in [6], whereas we use HTTP requests in case of CDN IP addresses to reliably extract geolocation information as described in the following.

This is somewhat similar to the case of traffic classification, where protocol specific information (e.g., Deep Packet Inspection) is used to build a ground truth against which classification algorithms that *do not* rely on such protocol specific information can be tested [47]. However, it is useful to stress that collection of protocol specific informaiton is used only in the validation phase, but is not necessary for the correct execution of the iGreedy algorithm.

CDN Ground truth. To collect CDN ground truth, we issue HTTP HEAD requests towards CloudFlare and EdgeCast to solicit a reply from the destination servers from PlanetLab. Unfortunately, it is not possible to issue HTTP HEAD requests from RIPE since the RIPE API does not provide HTTP support due to legal reasons (e.g., RIPE Atlas could be otherwise used as a proxy for accessing content restricted in some countries). It follows that from RIPE we are only able to issue ICMP measurements, whereas from PlanetLab we perform both ICMP and HTTP queries.

After manual inspection of the HTTP headers, we find that the HTTP reply headers contains meta-information about the servers location. Specifically, we observe that CloudFlare uses a custom `CF-RAY` header that uniquely identifies the server answering the HTTP request, whereas EdgeCast encodes such information in the standard `Server` header. Pairing such measurement data with PAI allows to reliably determine GT information.

CloudFlare encodes the server name directly as IATA airport codes, whose status is published at [48]. At the time we run our measurement campaign, EdgeCast used instead a mix of IATA codes and pseudorandom string for server names, publishing the servers list along with their geographical locations and the strings used to identify them at [49]. Currently, the page URL [50] as well as the information format has changed: while this is not surprising, it also testify that the collection effort of GT (and PAI) synchronously with MCs is far from being a trivial tasks. This further stresses the values of dataset sharing, which let the community capitalize on the effort of individual groups.

DNS Ground truth. To build a reliable DNS ground truth, we issue distributed IPv4 DNS queries of class CHAOS, type TXT, and name hostname.bind [35] to DNS root servers F, I, K, and L that are operated by ISC, Netnod, RIPE NCC, and ICANN respectively. We use both RIPE and PlanetLab to collect DNS replies to our queries: in the case of PlanetLab, we issue new ICMP and DNS measurement, wheras we rely in the case of RIPE of DNS measurement performed by the full set of vantage points that are already published by RIPE.

As in the previous case, pairing protocol-specific replies with PAI allows to reliably determine GT information in the DNS case. Indeed, despite CHAOS replies do not follow a standard format, GT is relatively easy to manually validate since some operators name servers in their infrastructure after IATA airport codes (e.g., AMS, PRG in root severs F and L respectively) or IXPs short names (e.g., AMS-IX, BIX, MIX in root server K). In other few cases (e.g., root server I), operators use arbitrary codes, but make publicly available a list that maps site codes to locations. In sporadic cases, multiple CHAOS names are located in the same city: as we are interested in locating geographically distinct replicas, as opposite to enumerating the number of physical or virtual servers operating on a site, we coalesce all replicas located in the same site.

E. Dataset at a glance

We depict some relevant properties of our dataset in Fig.2. Notably, we report the Cumulative Distribution Function (CDF) of the latency samples coalescing ICMP, DNS and HTTP protocols altogether (left) for different targets (top) and measurement infrastructure (bottom). Since each latency sample translate into a disk in the Earth surface, we report the disk radius in the top-x axis for reference: it can be seen that the large majority of latency samples, being larger than 10 ms, map to disks that are big enough to cover a whole country.

As we have seen in the previous section, ground truth for both DNS and CDN is expressed by the very same DNS and CDN operators in terms of IATA airport codes. We report the CDF of the number of IATA airports per disk in the right plot of Fig.2. Intuitively, the number of airports inside such disks can give a rough estimation of the difficulty of finding the correct replica location. for reference, top x-axis reports the probability of a random guess, which is inversely proportional to the number of airports in the disk. From the CDF in the bottom plot of Fig.2(b), one can notice that only about 10% of all disks contain less than 100 airports, or otherwise stated naïve guess has lower than 1/100 chances of success in roughly 90% of the cases: it follows that *anycast geolocation algorithms should thus be designed to be inherently robust to noise in latency measurement.*

A final point is worth clarifying. Our dataset also includes a set of over 3,000 airports, out of the about 7,000 available airports with a IATA code. The dataset associate airports with several properties, such as its latitude and longitude, the name of the city it serves, the population of the city, the airport "type" (taking values such as Small/Medium/Large Airport, Heliports, Closed Airports, Seaplane Bases, Balloonports, etc.) available in open databases. Of the about 7,000 airports with a IATA code, only about 500 airports worldwide fall in the large category – hence, the wide majority of airports in our dataset are of the "medium" type.

Notice that the type attribute correlates to the size of the airport, not to the size of the city population. Additionally, multiple airports of several type can be associated to the same city. Taking Paris as an example, the datasets[6] lists several airports (namely CDG, ORY, PAR, BVA and LBG). While Paris is a city with a given population (of about 2 million inhabitants) these airports have a variable size. Moreover, while CDG is very often use in naming severs, both CDG and ORY are "large" airports, that are sometimes found as well. PAR is a virtual airport, i.e., a label for any Paris airport to simplify searching flights. There are also less known airports such as BVA (essentially, low cost flights) and LBG (famous for international airshow every two years) airports, categorized as medium. The mapping between city and IATA is therefore a many-to-one mapping: given our target city-level geographic accuracy in this paper, this is not problematic, but shall deserve further attention to provide finer-grain geolocation (i.e., sub 100km radius or building-level [51]).

V. iGreedy Algorithm and implementation

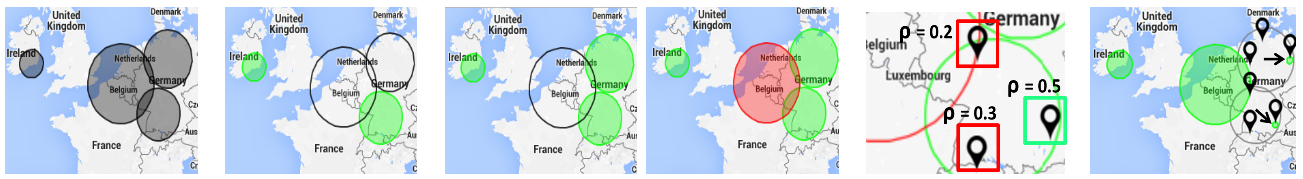
Our test suite also comprises an open source implementation of the iGreedy technique we originally proposed in [21], of which we illustrate the inner working with the help of Fig. 3. The implementation can operate on historical data (i.e., the dataset early described, corredated with ground truth) or issue new measurement from RIPE Atlas (in which case the provided GT is no longer up-to-date).

In a nutshell, (a) we first perform latency measurement to a given IP and map them to a disks centered around the VP, that by design contains at least one anycast instance; (b) if two such disks do not intersect, we can infer that VPs are contacting two different replicas, as is the case for the green disks in Fig. 3(b); (c) we provide a conservative estimate of the minimum number of anycast replicas by solving a Maximum Independent Set (MIS) problem, using a greedy 5-approximation algorithm that operates on disks of increasing radius size as in Fig. 3(c); (d) in each disk, we geolocate the replica at city-level granularity with a maximum likelihood estimator biased toward city population; and finally, (e) we coalesce the disks to the classified cities, which reduces disk overlap and allows iteration of the algorithm until convergence, thus increasing the recall (i.e., number of replicas discovered) along each iteration. We now describe the steps in more details.

A. Detection

At a logical level, prior to enumerating anycast instances, we must detect whether there are indeed anycast replicas behind a given unicast IP address. This can be done so by detecting speed-of-light violations in our dataset by comparing latency measurements δ to the expected propagation time due to speed-of-light considerations as in [5]. As early illustrated in Fig. 1(b), we consider pairs of latency measurements $\delta(p,t)$ and $\delta(q,t)$ for the same target. We map measurements to

[6]The airport dataset is available along with our provided software in `./igreedy-1.0/datasets/airports.csv`



(a) **Measurement**: Map RTT samples to disks centered around VPs

(b) **Detection**: Non-overlapping disks imply speed-of-light violation

(c) **Enumeration**: Solving a Maximum Independent Set (MIS) problem yields non-overlapping disk, each containing a different replica (two steps shown)

(d) **Geolocalization**: Maximum likelihood classification problem (city-level)

(e) **Iteration**: Collapse disks around geolocalized replicas until convergence

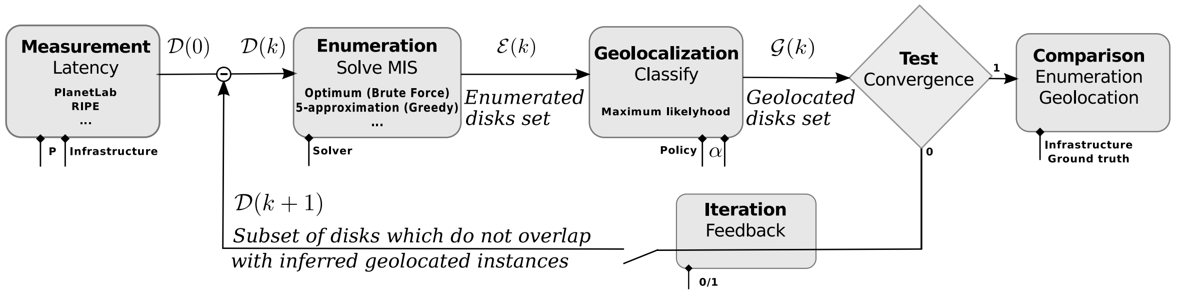


Fig. 3. Synoptic (bottom) and illustration (top) of the iGreedy anycast detection, enumeration and geolocalization algorithm

disks $\mathcal{D}_p$ whose radius equals $d^+(p,t) = \delta(p,t)/3$, where the constant 3 lumps altogether several factors (such as the speed-of-light in a fiber medium, the fact that fiber deployment is subject to physical constraints, etc.), of which a good overview is given in [52].

Specifically, given two vantage points p, q we compute their geodesic distance $d^g(p,q)$ according to Vincenty's formulæ. As packets cannot travel faster than light, if

$$d^g(p,q) > d^+(p,t) + d^+(q,t) \geq d(p,t) + d(q,t) \quad (1)$$

then disks $\mathcal{D}_p$ and $\mathcal{D}_q$ do not overlap, indicating that p and q are in contact with two separate anycast replicas.

Some remarks are in order. First, (1) compares distances with homogeneous dimensions, that are however gathered with different techniques. Note that considering the geodesic distance $d^g(p,q)$ between vantage points yields a *conservative lower bound* to the expected propagation time between p and q, as a packet will not travel along a geodesic path but will follow a path shaped by physical and economic contraints (i.e., the geography of fiber deployment, optoelectronic conversion, BGP routing, etc.). Instead, since latency is not only due to propagation delay, $d^+(p,t)$ conversely *aggressively upper bounds* the distance that a packet may have traveled during $\delta(p,t)$.

As the the inequality is violated only when the conservative lower bound exceeds the upper bound, it follows that (1) is conservative in detecting anycast instances and, assuming correct geolocation of VPs, by definition avoids raising false positive anycast instances (i.e., flagging as anycast a truly unicast target). Notice that in the iGreedy implementation, the detection step illustrated in Fig. 3(b) is a side product of the enumeration, and is thus not explicitly accounted for in the workflow of Fig. 3: i.e., in case two or more anycast replicas are enumerated, this also implies true positive anycast detection. Of course, while false negatives are possible on a single inequality (i.e., flagging as unicast a truly anycast target), their odd decreases using multiple vantage point pairs.

B. Enumeration

While the detection criterion expressed by (1) is not particularly novel [5], we are the first to leverage a full set of distributed measurements in the study of anycast deployment and its geographical properties as illutrated in Fig. 3(c). Distributed measurement allow indeed to infer more sophisticated properties than mere binary detection, such as the count of anycast replicas, or their position.

It is possible that multiple anycast instances may be located within a given disk. Although the aim of anycast is to offer services from distinct locations, the locations may be distinct from an IP routing point of view but not distant geographically from each other. Therefore, our technique can only provide a lower bound of the number of anycast instances that correspond to our observations.

MIS formulation. To achieve our enumeration goal and simplify the geolocalization step, we model the problem as a *Maximum Independent Set (MIS)*. Our aim is to find a maximum number of vantage points (and corresponding disks) for which we are confident they contact distinct anycast instances (an instance being included in the disk). To do so, we select a maximum subset of disks $\mathcal{E} \subset \mathcal{D}$ such that:

$$\forall \mathcal{D}_p, \mathcal{D}_q \in \mathcal{E}, \qquad \mathcal{D}_p \cap \mathcal{D}_q = \emptyset \quad (2)$$

The enumeration problem is thus solved by the subset $\mathcal{E}$, whose cardinality $|\mathcal{E}|$ corresponds to the minimum number of instances that avoid latency violations, and which represents thus a plausible explanation to our observations. Notice that $|\mathcal{E}|$ is a lower bound on the number of anycast instances, since due to the conservative definition of (1) we might have removed disks that overlap due to noisy measurements. Additionally, each disk of $\mathcal{E}$ yields a coarse location of anycast replicas useful starting point for refinement in the geolocalization step.

Efficient MIS solution. Although the MIS problem is NP-hard, it can be solved in finite time for small number of vantage points with a brute force approach. This allows us to compare the solution of known greedy approximate solutions: while a simple greedy strategy has poor performance in general ($(M-1)-$approximation) the situation improves by simply sorting disks in increasing radius size (5-approximation), with complexity $O(Mlog(M) + M(M-1)/2)$ determined by the comparisons. The greedy pseudocode is trivial, but reported here for completeness.

Algorithm 1 Greedy 5-approximation to MIS for anycast instances enumeration

```
Require: A set of disk $\mathcal{D}$
Ensure: A set of disk $\mathcal{E}$ such as $\forall p, q \in \mathcal{E}, \mathcal{D}_p \cap \mathcal{D}_q = \emptyset$
  Initialization: sort disks in $\mathcal{D}$ by increasing radius size
  Initialization: $\mathcal{E} \leftarrow \emptyset$
  for all disk $\mathcal{D}_d$ of $\mathcal{D}$ do
    for all disk $\mathcal{D}_e$ of $\mathcal{E}$ do
      if $\mathcal{D}_d \cap \mathcal{D}_e = \emptyset$ then
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{\mathcal{D}_e\}$
      end if
    end for
  end for
```

We point out that more refined solutions do exist [53], [54], that achieve $(1 - \frac{2}{k}) - OPT$ performance at the price of a non marginal computational complexity $M^{O(k^4)}$, where k is a tunable parameter. Roughly, the main idea in [53], [54] is to slice the original dataset into several layers, with disks into layer ℓ having diameter d satisfying $1/(k+1)^\ell \geq d > 1/(k+1)^{\ell+1}$. The more the layers, the closer the approximation to the optimum, the longer the computational time. Notice also that k must be greater than 2 to apply the slicing.

As we will show later, the greedy solution often performs well in practice and is often comparable to the brute force solution, which means that more refined solution are not worth for this problem. Since computational time of a greedy approximation for O(100) vantage points is in the order of O(100ms), whereas brute force solution is O(1000sec), a simple greedy MIS solver has an undoubt practical appeal.

Alternative formulation. A final remark is worth making. Had our original goal been limited to the *enumeration* of anycast instances a *Boolean satisfiability (SAT)* formulation would have been more appropriate. A geometric interpretation of our problem would be then as follows: if we represent a anycast instance using a cross, SAT consists in placing a minimum number of crosses such that all the disks $\mathcal{D}_p$ contains at least one cross. A benefit of this formulation is that SAT upper bounds the number of instances returned by MIS, since MIS removes overlapping disks from the set: hence, a SAT formulation would increase the recall of anycast instances. At the same time, the SAT formulation would make the geolocalization harder and is thus not adapted to our goals: indeed, as several realizations having exactly the same number of anycast instances could satisfy this problem, SAT does not easily allow to determine in which circles intersection the anycast instances are located. Releasing datasets as open source should facilitate implementation of alternative techniques for replicas enumeration such as the one just illustrated.

C. Geolocalization

Our aim being to provide geographic locations at city granularity, we need to refine the preliminary location that is output by the enumeration algorithm. We opt for city granularity for two reasons. First, recall that a 1 ms noise in latency measurement corresponds to an increase in the disk size by 100 km. It follows that great trust should be put in latency measurements to achieve finer-grained geolocalization. Second, notice that the ground truth provided by DNS and CDN measurement is already provided as IATA airport codes. City-level granularity naturally allow to assess the correctness of our geolocalization.

Classification formulation. As opposed to classical approaches that operate in the geodesic (or Euclidean) space by constructing density maps of likely positions (see references in [26]), or assessing target location to be the center of mass of multiple vantage points [30], we transform the geolocalization task into a classification problem as in [27]. Specifically, since our output is a geolocalization at city level granularity, we shift the focus from identifying a geographical locus $(lat, lon) \in \mathcal{D}_p \subset \mathbb{R}^2$ to identifying which among the cities $c \in \mathcal{C} \subset \mathbb{N}$ contained in the disk $(lat_c, lon_c) \in \mathcal{D}_p$ is most likely hosting the anycast instance. This focus shift greatly simplifies the problem in two ways: first, it significantly reduces the space cardinality ($\mathbb{R}^2$ to $\mathbb{N}$); second, it allows us to further leverage additional information with respect to delay or distance measurements, namely the city population.

Geolocalization step outputs IATA airport codes as shorthand for cities. For each of the non-overlapping disks of the enumeration phase, some of the over 7,000 airports codes may be contained in the disk. Aside from the trivial case where a single airport is contained in the disk, in the general case multiple airports $\{A_i\} \in \mathcal{D}_p$, represented as a crosses in Fig. 1(b), are contained in any given disk. The output of the geolocalization phase can thus be expressed with disk-airport pairs $\mathcal{G} = \{(\mathcal{D}_i, A_i)\}$ according to the notation of Fig. 3(f).

Classification criterion. To guide our selection of the most likely location of a site, we employ two metrics, namely: (i) the distance between the airport and the disk border $d(p,t) - d(p,A_i)$ and (ii) the population c_i of the main city c_i that the airport A_i serves. Our intuition in using (i) the distance between an airport and the border of a disk is that the larger the distance, the nearer is the airport from the vantage point. Here, we use geographic proximity from the vantage point as a proxy for topological proximity in the routing space. Our reasoning for (ii) extends previous work [27], which argues that IPs are likely to be located where humans are located: in other words, due to the distribution of population density, large cities represent the likely geolocalization of single-host unicast IP addresses. We further argue that, since anycast replicas are specifically engineered to reduce service latency, they ultimately have to be located close to where users live: hence the bias toward large cities is again likely to hold for server side anycast IPs as well. Hence, we include only airports that are categorized as “medimum” and “large” in the database, but exclude airports categorized as “small”, because we are only interested in locations that are densely populated. Our IATA dataset contains a total of over 3000 airports.

For a given disk $\mathcal{D}_p$ we compute the likelihood of each airport $\{A_i\} \in \mathcal{D}_p$ for all airports in the disk, illustrated in Fig. 3(d) and defined as:

$$p_i = \alpha \frac{c_i}{\sum_j c_j} + (1-\alpha) \frac{d(p,t) - d(p,A_i)}{\sum_j d(p,t) - d(p,A_j)} \tag{3}$$

where $\sum_i p_i{=}1$ follows from the normalization over all airports $\{A_i\} \in \mathcal{D}_p$ of the c_i contribution (population of the main city served by airport A_i) and of the $d(p,t) - d(p,A_i)$ contribution (the distance of the airport i from the disk border). The parameter $\alpha \in [0,1]$ tunes the relative importance of population vs distance in the decision, in between the distance-only ($\alpha = 0$) vs city-only ($\alpha = 1$) extremes. Likelihood for three cities is examplified in Fig.3(d). Based on the p_i values, we devise two maximum likelihood policies that return either (i) a single $A_i = \text{argmax}_i p_i$ or (ii) all locations (A_i, p_i) annotated with their respective likelihoods. These policies involve a trade off, as returning all locations increases the average error (since in case $\text{argmax}_i p_i$ is correct, it pays the price of incorrect answers for $1 - p_i$), whereas returning a single location possibly involves a larger error. As we shall see, the city population has sufficient discriminative power alone, so that the simplest criterion of picking the largest city is also the best:

$$A_i = \text{argmax}_i c_i / \sum_j c_j \tag{4}$$

D. Iteration

Recall that the enumeration step lower bounds the number of instances, due to the possibility of overlapping disks. Now, consider that the geolocalization decision in effect transforms a disk $\mathcal{D}_p$, irrespective of its original radius, into a disk $\mathcal{D}'_p$ centered around the selected airport with arbitrarily small radius (in this work, we conservatively shrink disks to a 100Km radius). Hence, we argue that, provided the geolocalization technique is accurate, it would be beneficial to transform the original set of disks $\mathcal{D}$ by (i) remapping $\mathcal{D}_p$ to $\mathcal{D}'_p$ and (ii) excluding from $\mathcal{D}$ those disks that contain any of the geolocalized cities $\mathcal{D}'_p$. This case is illustrated in Fig. 3: the red-shaded disk overlapping in the previous enumeration step Fig. 3(c), no longer overaps after geolocalization of the two circles in Fig. 3(d), so that it can be considered in the next iteration Fig. 3(e). Denoting with $\mathcal{A}(k)$ the subset of airports geolocalized up to step k, and with $\mathcal{G}(k)$ the geolocalization at step k (considering a single airport selected per disk for the sake of simplicity), defined as $\mathcal{G}(k) = \{(\mathcal{D}_i, A_i) \in \mathcal{E}(k) \times \mathcal{A}(k))\}$, we have that the dataset $\mathcal{D}(k+1)$ as input to the numeration problem at step $k+1$ can be written as $\mathcal{D}(k+1) = \mathcal{D}(k) \backslash \{\mathcal{D}_i : \exists (\mathcal{D}_i, A_i) \in \cup_{i=1}^{k} \mathcal{G}(i)\}$. Iterations continue until no further disk can be added that does not overlap. At each iteration, the set of geolocalized cities grows, so that the set of disks that no longer overlap diminishes, which keep the running time reasonably bounded. Note that iterative operations can be employed irrespectively of the underlying solver (i.e., brute force, greedy, etc.). Notice also that iteration “couples” the analysis of the geolocalization and enumeration performance, as the input to the latter is modified by the former. We analyze coverage benefits and additional complexity of iterative workflow in Sec. VII-B.

VI. Results at a glance

We now run the open source iGreedy implementation on the dataset[7] provided to the community, reporting at a glance illustrative examples and key performance indicators. Results in this section are gathered with the “largest city” criterion of (4) – we defer justification of this choice to Sec. VII, where we perform a thorough assessment of iGreedy robustness.

A. Example of results

Geographical maps (using a GoogleMaps interface) are among the outputs directly available in the open source iGreedy implementation. For the sake of illustration, Fig. 4(a) depicts an Euro-centric view of geolocalization of root server L replicas: the map reports vantage points (black dots) and the results of iGreedy as shaded disks that contain either correct ✓ (True Positive, TP) or erroneous $\times$ (False Positive, FP)

[7]It is worth recalling that datasets are released in the same tarball of the iGreedy code [23].

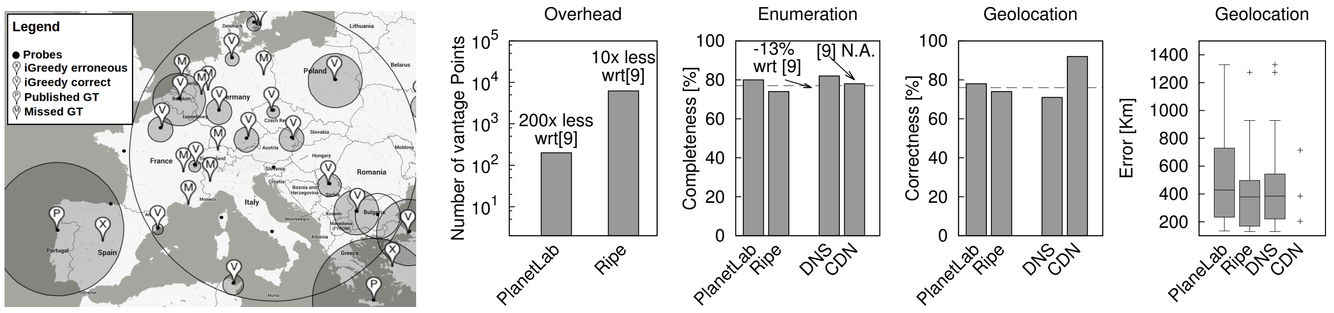


Fig. 4. iGreedy results: (a) Illustrative example of root server L and (b) compact summary of geolocation and enumeration performance across MI (PlanetLab, RIPE) and breakdown of PlanetLab targets across services (DNS, CDN). Note that only 3 replicas are at erroneous location in the PlanetLab CDN case, illustrated as points in the geolocalization error boxplot.

geolocalization markers (and in the last case the location of the ground truth P as well) according to the earlier discussed ground truth. The map additionally reports missed M instances (False Negative, FN), whose position is known through public available information. Such instances were missed either because (i) they are not observed in our measurement, which is the cause of the large majority of this misses or because (ii) they are observed in disks that overlap (represented as circles with no shading). Additionally, notice an example of vantage points that our iterative workflow allows to include: i.e., disks of the Bruxelles and Paris vantage points intersect. Finally, observe that population bias yields to misclassification for the point located in Porto, Portugal: this vantage point exhibits a relatively large latency (6 ms) to hit a target also located in Porto, so that the disk is large enough to include Madrid (population of 3.3M) which is an order of magnitude more populated than Porto (population of 250K). The distance between Madrid and Porto is 420Km, which is just 10% above the median geolocation error of iGreedy.

B. Comparison with the state of the art

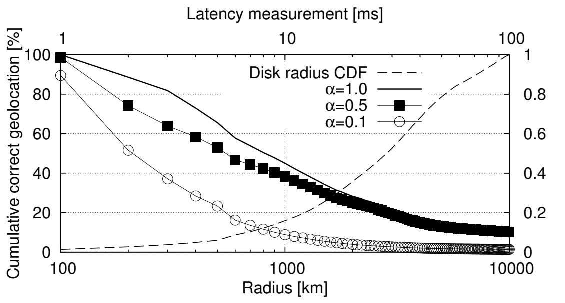
We separate analysis of iGreedy enumeration and geolocation as follows. Enumeration aims at *completeness*, i.e., assessing the number of disks $|\mathcal{E}|$ contacting different replicas that iGreedy is able to recollect, independently whether the geolocation succeed. We normalize the number of replicas to the number in the ground truth obtained with protocol specific information and denote this ratio $|\mathcal{E}|$/GT as enumeration completeness. Geolocalization instead aims at *correctness*, so that it is important to assess the amount of geolocation that correctly matches the ground truth, which can be expressed as the True Positive Rate or Precision = TP/(TP+FP).

Notice that the enumeration results of [6] are *directly quantitatively* comparable, as [6] employs F and other root servers as a case study, albeit the measurement timeframe and infrastructure differ. Conversely, since iGreedy is the only technique able of anycast geolocation, we do not have any candidate technique to directly[8] compare with in this case, and thus make our dataset open to promote and facilitate future comparisons.

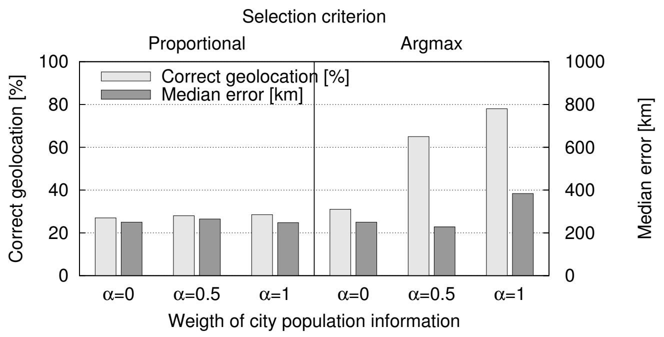
At a glance, Fig. 4(b) shows that, with respect to [6], iGreedy: (a) reduces the measurement overhead by several orders of magnitude, (b) has comparable yet lower enumeration recall. Additionally, while [6] technique is limited to DNS and does not allow geolocalization, iGreedy: (c) is protocol protocol agnostic and (d) is able to correctly guess anycast instance location about 3/4 of the time. Finally, we can see that results are (e) qualitative consistent across service and measurement infrastructure. As for (a), we indeed notice that [6] employs 62K vantage points (the Netalyzr dataset), i.e., about 200$\times$ larger than PlanetLab and 10$\times$ larger than RIPE Atlas. As for (b), since we observe enumeration performance that are worse than that of [6] but anyway comparable, this intrinsically means that the datasets used in [6] are highly redundant (e.g., including multiple trials from the same users; or affected by popularity of Netalyzr in a given geographical region). The very same observation also holds for the MI we use in this paper (e.g., RIPE has several hundreds monitors in Paris, but iGreedy uses at most one of them), so that we explore this issue further in Sec. VII-C. As for (c), iGreedy is protocol-agnostic because only latency measurements are needed, that can be gathered from service-independent protocols such as ICMP. As for (d), we report that geolocation is correct in 78% of the cases, and that the median error distribution of the remaining 22% of the cases is 384 Km.

Making a parallel to unicast geolocation, it is worth noticing that error magnitude in iGreedy is similar to that of

[8]In a sense, [6] also provides geolocation in CHAOS replies, which is the very same information we use to build the ground truth: however, its use is limited to DNS and is conditioned to having location information encoded in server names, which we have seen to apply to only part of DNS root servers. Similarly, our previous work [21] compares geolocalization results of the *unicast* Client-Centric Geolocalization (CCG) [30] (which are however *only qualitatively* comparable, as they additionally target the Google infrastructure). As qualitative comparison may lead to misinterpretation, we prefer to avoid reporting it here (which is available to the interested reader in [21])



(a) Geolocalization over individual disks (RIPE, PlanetLab)



(b) iGreedy sensitivity (PlanetLab)

Fig. 5. Tuning iGreedy classification: (a) cumulative true positive geolocalization over all disks, with argmax selection for different weighting factor α. (b) iGreedy performance for different selection policies and weighting factor settings.

unicast techniques: e.g., without (with) filtering large delay samples, [30] reports a 556Km (22Km) median error. An additional similarity with is worth stressing, quoting [30] “A disadvantage of our geolocation technique is that large datacenters are often hosted in remote locations, and our technique will pull them towards large population centers that they serve. In this way, the estimated location ends up giving a sort of logical serving center of the server, which is not always the geographic location.”: the very same hold here for iGreedy.

VII. Sensitivity analysis

We now thoroughly validate iGreedy (e.g., classification in Sec. VII-A, solver in Sec. VII-B) and assess its robustness to measurement campaigns and infrastructures (Sec. VII-C).

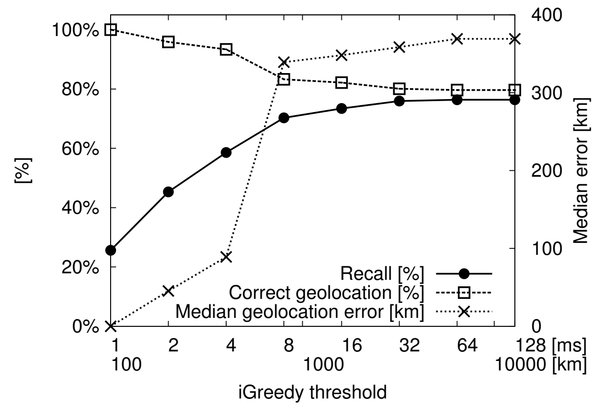
A. Impact of classifier

Weighting factor α. It could be argued that, at least for small disks, latency information could bring useful information to improve classification accuracy. To assess this, we first consider individual disks $\mathcal{D}_p$, where we apply the geolocation criterion interpolating distance and population information via $\alpha \in \{0, 0.5, 1\}$ in (3). As we consider all disks irrespectively if they will be selected in the iGreedy enumeration phase, this information is valuable for all algorithms. Fig. 5(a) reports geolocation accuracy over all measurement platforms, targets and protocols: in particular, the plot shows the average correct geolocation ratio, cumulated over all disks up to a given size. As it can be seen from Fig. 5(a) the probability of correct geolocation is upper-bounded by $\alpha = 1$: this suggests that city population has discriminative power useful for any anycast geolocation algorithm in general, and ultimately corroborates the simple “largest city” criterion of (4).

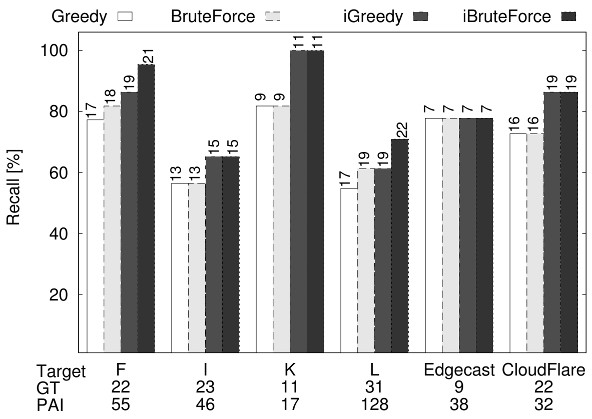
Given that geolocation is still erroneous in about 20% of the cases, this is an area for future improvement, using either complementary measurement (e.g., traceroute) or side-channel information (e.g., Internet exchange points maps [51]). Of course, we believe that improving geolocation accuracy of the classification step is an important point, as there is much room for improvement from the coarse-grained 100km-radius accuracy iGreedy aims at providing, towards the fine-grained building-level accuracy provided by [51]. Indeed, on the one hand one could argue that an anycast geolocation service is not useful given the accuracy does not reach acceptable levels (e.g., say 95%). On the other hand, one could also argue that even small improvements can accumulate over time and provide eventually an acceptable solution. Our hope is that this paper, along with the software and the dataset we provide, can facilitate this second option to let the community collectively refine the existing service to an acceptable level.

Selection policy. We now turn our attention to iGreedy, where only a subset of all the above disks are selected by the algorithm: over this set of disks, we study combination of the selection policy (i.e., argmax vs proportional) and weighting factor ($\alpha \in \{0, 0.5, 1\}$). For any given disk $\mathcal{D}_p$, let us denote with A^{GT} the airport code given by the ground truth, and further denote with A_i the different airports that are located in $\mathcal{D}_p$. Considering the argmax policy, in case $A^{GT} = A_i$ (with i such that $\text{argmax}_i\, p_i$), the classification is accounted as correct and does not count in the error statistics. In case $A^{GT} \neq A_i$, then the classification is erroneous, and off by a distance $Err = d(A_i, A^{GT})$. In the proportional policy instead, the classification is accounted as correct only for p_i (i.e., proportionally to the percentage of time the correct instance would be selected). The geolocalization error for this instance is then computed over all airports inside the disk, and weighted according to the respective likelihood of each airport $Err = \sum_j d(A_j, A^{GT})p_j$.

In short, the tradeoff is here between optimistically returning a single location and possibly have large maximum error (argmax) vs conservatively bounding the maximum error but increase the average error (proportional). Results are reported in Fig. 5(b), from which it is easy to gather that, given that



(a) Thresholding input data (PlanetLab and RIPE)



(b) Impact of MIS Solver (PlanetLab)

Fig. 6. iGreedy Sensitivity: (a) Thresholding input data (PlanetLab and RIPE) and (b) Impact of MIS Solver (PlanetLab)

iGreedy preferably select smaller circles, and in reason of the good discriminative power of the city population, argmax is largely preferable with respect to the proportional policy. As early noticed performance are already similar for $\alpha > 0.5$, although $\alpha = 1$ confirms to be the best setting, leading furthermore to a very simple geolocalization criterion.

B. Impact of input data and solver

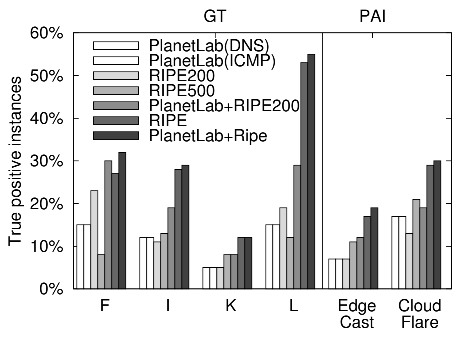
Latency measurement provides an input that is fed to a MIS solver for the enumeration phase. Given the large uncertainty in accurate geolocalization of replicas in large disks, it could be tempting to filter out latency measurement exceeding a configurable threshold. This is interesting not only to bound the maximum error, but also since a smaller dataset can reduce the computational time spent in the solution of the MIS.

Filtering input data. We thus start by filtering latency measurements fed to the MIS, by setting a maximum threshold: i.e., measurement larger than the threshold will be discarded, which explicitly upper-bounds the maximum error. Intuitively, this yields to a tradeoff between accuracy and completeness: indeed, each disk by definition contains at least an anycast instance, and despite the larger the disk, the harder the geolocation, however discarding large disks potentially discards useful information. Fig. 6(a) show average performance over all dataset, as a function of thresholding: clearly, small thresholds negatively affect recall, which is undesirable; instead, even for for very large thresholds (e.g., over 100ms or 10000Km radius where geolocalization almost certainly fails), the completeness saturates, the correct geolocation stabilizes and the geolocation error does not increase (despite iterations). This desirable property follows from the fact that iGreedy *order disks by size* and thus *implicitly filters* out the largest circles from the solution. Hence, we do not recommend thresholding measurements to be fed to iGreedy, which is inherently robust to outliers – irrespectively of their nature such as BGP path inflation, queuing delay, etc. [52].

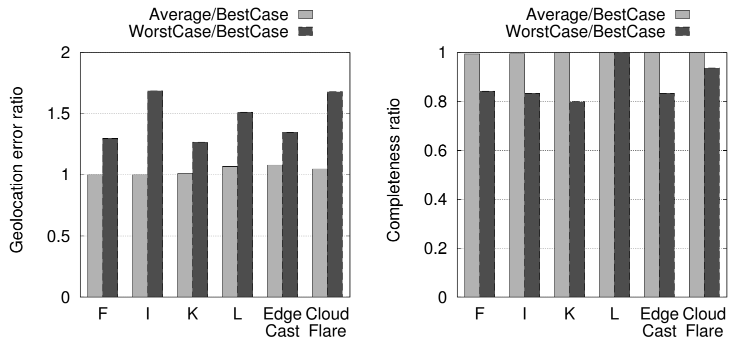
MIS Solver. Fig. 6(b) reports statistics about the enumeration of our targets from PlanetLab with different solvers. For each solver, the picture reports the completeness $|\mathcal{E}|$/GT; each bar annotates the number $|\mathcal{E}|$ of anycast replicas found, and the x-label is annotated with GT and PAI information for that target. It can be seen that the greedy solution is as good as that of the brute force solution (I, K, EdgeCast, CloudFlare) or anyway comparable (F, L). More interestingly, the iterative workflow produces benefits that are sizeable and consistent across datasets and solvers. Note also that most of the replicas are found in the first iteration, which keeps the number of iterations (and associated computational complexity) limited.

Solver selection is also affected by computational complexity: under this perspective, the running time of iGreedy (*hundreds of milliseconds*) is orders of magnitude smaller with respect to that of the brute force approach (*hundreds to thousands of seconds*). Indeed, that while we were able to obtain brute force solution on the PlanetLab dataset, its cost is prohibitive for the larger RIPE Atlas datasets. Thus, it also follows that, while refined solutions do exist [53], [54], they are not appealing due to the good enough performance and short running time of the iGreedy solver.

Iteration. As depicted in Fig.3(f), there are two loops: an inner loop (to solve a single MIS enumeration step) and an outer loop (the iterative feedback of iGreedy or iBruteForce). Clearly, while the complexity of the inner loop depends on the solver it is interesting to analyze the impact of the outer loop iteration. Notice that a partial answer to this question is already available from Fig. 6(b): indeed, comparing a MIS solver with its iterative iMIS version (e.g., Greedy vs iGreedy or BruteForce vs iBruteForce) one can gather an *upper bound*



(a) Sensitivity to the measurement infrastructure and protocol



(b) Sensitivity to the number of samples per vantage point

Fig. 7. Sensitivity of iGreey to the measurement infrastructure and campaigns: (a) impact of vantage point combinations and network protocols; (b) best, average and worst-case results with $P = 100$ measurement per vantage point.

TABLE III
OVERHEAD OF ITERATIVE WORKFLOW

Target	Elapsed time (s)		Iteration
	Greedy	iGreedy	overhead
F	0.10	0.15	+50%
I	0.08	0.12	+50%
K	0.07	0.09	+20%
L	0.09	0.15	+66%
EdgeCast	0.09	0.09	+0%
CloudFlare	0.11	0.15	+36%

to the number of loops that are executed.

Notice that the picture reports a label on top of each bar, which represents the number of replicas discovered: since *at least one* replica is discovered in a new loop (else the loop terminates), then *at most* a number of loops equal to the number of newly discovered replicas through the iteration can be executed. At the same time, this estimation *upper bounds* the number of loops since it is also possible that after the classification steps (and the disk shrink), multiple non-overlapping disks are found in a single iteration. Notice further the upper bound is anyway pretty low (e.g., 0 in Edgecast, 2 for I and K and 3 for CloudFlare for both Greedy and BruteForce). It follows that the number of outer iterations is typically upper-bounded by 2-3.

To estimate the additional complexity due to iteration, we have to consider that the duration of the overall workflow is not dominated by the first iteration, and that rather each iteration can be considered to have an approximately fixed time, since all disks need to be considered at each new outer interation in the MIS solution. We point out that albeit an optimization is possible (i.e., removing the already geolocalized disks) its impact is expected to be pretty low: indeed, this would amount at removing some few tens of disks out of several hundreds (PlanetLab) or several thousands (RIPE Atlas) vantage points, so that we do not implement it.

Overall, we can safely upper-bound the running time of iGreedy to about twice as much the running time of the simple MIS greedy solver. However, in practice average runtime increase is significantly more contained. Considering the PlanetLab infrastructure for the sake of simplicity, Tab.III reports the running time (in seconds, averaged over 10 repetition) for non-iterative (Greedy) vs iterative (iGreedy) versions of the algorithms executed on a Intel i7 running 64-bits Linux OS with kernel version 3.17. The relative runtime (last column, denoted as overhead) is also reported with respect to the Greedy baseline. It can be seen that, in practice, iteration increases computational complexity by less than 50% on average.

C. Impact of measurement infrastructure and campaign

We assess the impact of MI and MC by (i) considering different combinations of vantage points from multiple MIs, and by performing latency measurement (ii) with different protocols as well as (iii) with a varying number of samples for the same protocol.

Vantage points. RIPE Atlas and PlaneLab have largely different characteristics concerning AS and country coverage, as well as vantage points footprint. We thus expect MI to play a paramount role in determining the results. Indeed, lack of VPs in an area where anycast instances lay will likely result in false negative (in case of disks overlap) or false positive (in case of a single large disk covering the area). To reduce the intrinsic VP redundancy, we also consider smaller subsets in the case of RIPE Atlas, since as previously noticed, there is an intrinsic redundancy in VP placement (e.g., we count over several hundreds RIPE Atlas hosts in Paris, albeit at most one can be useful for our purposes).

We thus consider (i) PlanetLab, (ii) a selection of 200 RIPE Atlas VPs that are at least 100 km distant from each, (iii) a random selection of 500 RIPE Atlas VPs, (iv) the combination of PlanetLab and RIPE200, (v) the combination of PlanetLab and the full RIPE dataset. Notice that, when combining MIs, our methodology seamlessly combine DNS (from RIPE) and ICMP (from PlanetLab) measurement. Given that HTTP measurement (hence, GT information) is not available from RIPE, when combining PlanetLab and RIPE CDN measurements we compare results with the PAI.

Some remarks are in order. First, as expected, the set of VP plays a paramount role: increasing the number of VPs increase the number of valid instances that iGreedy is able to find. Second, notice that even simple selections that avoid replicas in the same city (RIPE200) are better than random selection (RIPE500). Third, combining PlanetLab and RIPE increases the enumerated replicas – which holds even considering the full set of RIPE VPs and despite PlanetLab has about 20$\times$ less VPs than RIPE. Combination of VPs is done offline: since the APIs of these MIs are totally different, it is not easy to control a single experiment jointly using VPs from both platforms. Yet, these results suggest that it would be desirable to leverage protocols such as LMAP [55] or mPlane [56] to more systematically exploit both platforms in a unified interface.

Network protocol. RTT latency can be obtained with several protocols other than network-layer ICMP measurements. As a matter of fact, a side-product of our DNS and HTTP ground-truth measurement campaigns, we obtain delay measured with application-layer protocols. Notice that for DNS CHAOS requests, RTT latency include the server response time, whereas for HTTP HEAD we rely on the TCP three-way handshake duration (as including the server response, would overestimate the distance by about a factor of two). Of course, GT campaign only offer a single latency sample, which is however not expected to have a major impact as just outlined. By running iGreedy over latency samples gathered by these different protocols, *we do not observe any quantifiable difference in the results* (for completeness, this absence of variability is illustrated in Fig.7(a) in the PlanetLab DNS vs ICMP case, but we experimentally confirm it to hold over all PlanetLab targets and protocols combination). While robustness stem from the fact that iGreedy do not geolocalize based on latency measurement, this reinforces the soundness of ICMP choice to jointly achieve protocol-independence and correctness at the same time.

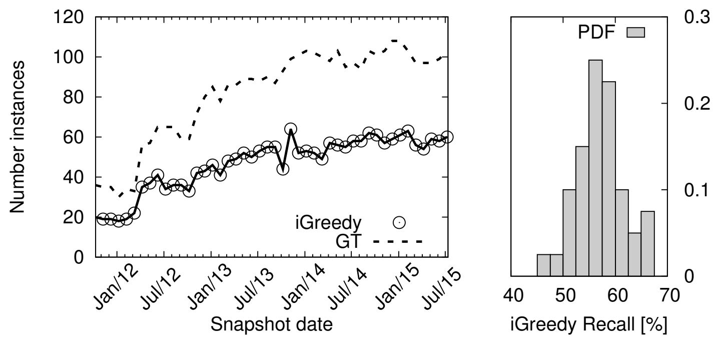
Varying number of samples. Results with varying number of P latency measurements are reported in Fig.7(a). Specifically, we run a measurement campaign gathering 100 ICMP samples per VP-target pair, and then consider: (i) the best case where $\delta^{BC}(p,t) = min_i \delta_i(p,t), \forall p,t$; (ii) the worst case where $\delta^{WC}(p,t) = max_i \delta_i(p,t), \forall p,t$; (iii) the typical case where $\delta^{TC}(p,t)$ are extracted at random from the $\delta_i(p,t)$ population. We next run iGreedy over such sets, and compare the performance over these sets: notice that while best and worst cases are deterministic, for the typical case we consider 100 such sets, so to obtain performance that are statistically representative of an average case. To quickly give an idea of the robustness of iGreedy performance to *real* input data noise, Fig.7(a) reports the mean error (left) and the completeness (right) for the worst and average cases, normalized over the best case as a reference. As it can be seen, performance deteriorates slightly even in the unlikely worst case where noise is maximum for all vantage points. More interestingly, performance of the *average case are almost indistinguishable from the best case*: this suggests that *even less than a handful of measurements per vantage point* is sufficient to correctly geolocalize anycast replicas with iGreedy.

VIII. Applications

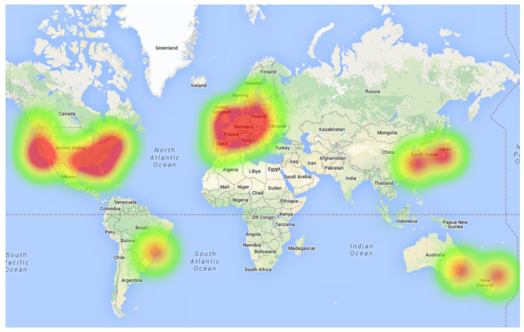
The anycast enumeration and geolocation technique we propose can be a useful starting point for several application having high practical relevance, illustrated in Fig. 8.

Longitudinal study. iGreedy can be used to perform a longitudinal study of anycast infrastructure evolution. RIPE Atlas provides the possibility to retrieve historical measurement data, and has been issuing periodic DNS CHAOS requests to all root servers since October 2011. We obtain monthly snapshots of these DNS measurements: for the sake of illustration, we select the most challenging case represented by the root server L, which has the largest deployment, and run iGreedy on each monthly snapshots. Since measurements have been performed with DNS CHAOS, we are able, for each snapshots, to build a reliable ground truth against which we can compare iGreedy results. Fig. 8(a) depicts the time evolution of the number of replicas in each snapshot, along with the number of instances correctly geolocated by iGreedy, and is completed with an PDF of the recall in the different snapshots. Notice that in these snapshots not only the L root server deployment, but also RIPE Atlas have changed, which also contributed to the temporal evolution: hence, results reported in Fig. 8(a) are of anecdotal relevance (i.e., single target; RIPE infrastructure evolution; etc.), but nevertheless testify the feasibility of anycast infrastructure mapping over historical data.

Anycast census. A complementary viewpoint to temporal evolution of a specific deployment is represented by a broad spatial analysis of anycast deployments. We depict in Fig. 8(b) the current state of our ongoing effort to perform full IPv4 anycast censuses from PlanetLab [2]. Shortly, for each IP/24 subnet, we first find a responsive target IP address, toward which we perform ICMP latency measurement from all PlanetLab hosts, running iGreedy on the resulting dataset. Given the sheer size of our target set, we run in this case a custom efficient scanner [57] that stores measurement as binary data, and analyze



(a) Longitudinal study (Evolution of root server L from RIPE)



(b) Spatial study (IPv4 census from PlanetLab)

Fig. 8. Applications of iGreedy: (a) longitudinal study of DNS root server L evolution using historical RIPE measurement and (b) density map of anycast replicas over a full IPv4 census from PlanetLab.

the dataset with a low-level C iGreedy implementation (not available at [22] for the time being). We however make results of this census browsable at [22]: for the time being, the website contains results for the top 100 anycast ASes comprising 897 IP subnets, for which we find 11,598 replicas that are localized in 71 cities of 36 countries. It is worth putting these results in perspective with [5], which use latency-based technique to perform anycast detection from distributed Renesys monitors: despite the measurement period and infrastructure differ, it is interesting to observe that that they detect 593 anycast prefixes (0.13% of the global IPv4 routing table) used mostly for CDN, DNS, and DDoS protection services. Of these prefixes, 88% are /24, 3% are /23, and 9% are larger (they suggest that larger prefixes may be anycast in part, which can be due to BGP prefix aggregation). In reason of its lightweight, iGreedy is amenable to continuously perform such censuses, extending the longitudinal studies to the spatial dimension.

BGP hijack detection. As IP-level anycast is realized through announcement of the same BGP prefix from multiple points, the iGreedy technique could be used to perform or assist BGP hijacking detection – as anycast and hijacks are indeed "syntactically" equivalent with respect to a router speaking the BGP "lingo". Despite a large literature on the topic [58], to the best of our knowledge most work exploits features related to AS-paths, whereas latency information such as the one we propose here has so far been ignored. Closest work in this context is [59] where suspicious announces trigger traceroute measurements from RIPE Atlas, which provides additional information to determine whether the announce is a hijack. Technique such as iGreedy could be run reactively as in [59], gaining in timeliness with respect to executing a full traceroute.

Troubleshooting. Finally, iGreedy could be useful in general, adding e.g., a relevant feature for troubleshooting [60], including e.g., ensuring reachability of specific anycast replicas, or detecting unexpected affinity between a specific replica and (a faraway) vantage point. Additionally, some inference techniques can be applied only on the unicast context [61], [62], where authors generally have to resort to some heuristic to discard suspiciously anycasted instances: in this context, iGreedy could either automatically validate the assumption, or raise a flag forbidding to use such unicast-only techniques in case of positive detection.

IX. Conclusion

Use of anycast has increased in the last few years, venturing out of the DNS realm and revealing a sizeable footprint in, e.g., CDN services. At the same time, measurement techniques for anycast infrastructure discovery are either protocol agnostic but limitedly offer detection capabilities [5], or offer enumeration capabilities but are limited to DNS [6].

The contributions of this paper are to provide the researchers and practitioners communities with (i) iGreedy, a tool able of lightweight, protocol-agnostic anycast replicas enumeration and geolocation on the one hand, and with (ii) a suite of datasets and software, to facilitate the development, validation and comparison of new techniques on the other hand.

The key to iGreedy performance is the formalization of the *enumeration step* as a Maximum Independent Set problem to maximize the replica coverage; and formalization of the *geolocalization step* as a classification problem, that leverages city population to factor out noise in the latency measurements. iGreedy leverages latency measurement from any protocol, and can seamlessly integrate measurements from heterogeneous protocols. Being inherently robust to outliers make the technique extremely lightweight: even a single latency sample per vantage point, from a few hundreds vantage points suffices to provide satisfactory enumeration and geolocation performance. In reason of its lightweight and its low computational complexity, the technique is amenable to continuous large scale measurements, which is hopefully helpful in refining our understanding of the Internet.

APPENDIX

For completeness, we provide an overview of the capabilities of the version of iGreedy released at time of writing. We expect to release an improved version of the code soon (thorugh GitHub) that will provide new features with respect to those described in this section; at the same time, the code will remain backward compatible so the information in this section will not become out-dated by future releases.

This section is loosely based on the README file released in the tool [23]. Briefly, the tool allows to (i) analyze existing measurement or (ii) generate and analyze new measurement (iii) visualize the measurement on a GoogleMap. The package also contains (iv) datasets corredated with ground-truth to assess the accuracy of the tool. Notice that the released version does not allow to generate new ground-truth, as this step often involves some level of manual verification (e.g., recall the SGW example in the Ground Truth section), and is thus difficult to fully automate.

A. Installation and configuration

iGreedy should run out of the box: iGreedy is written in Python (2.7 flavor) and there is no Python depenedency which we are aware of: all the code you need is in the `code/` folder of the tarball.

While running iGreedy on the provided datasets does not require any special configuration, however to launch new measurement from RIPE Atlas you need to (i) have a RIPE Atlas account (ii) have enough credits (iii) configure your authentication. Measurement are launched by `code/RIPEAtlas.py` which is going to read your RIPE Atlas key from the `datasets/auth` file.

Finally, we assume that the chrome (or chromium) Webbrowser is installed (which in our experience renders the Google maps better), at it is automatically launched whenever users request to display results in a map.

B. Usage

iGreedy faciliates interaction by providing some command line parameters to either process existing measurement from an input file (`-i input`) or to launch new measurement toward a target alive IP address (`-m target`). This choice is the only mandatory parameter (`-i input|-m target`).

In case Ground Truth (GT, preferable) or Public Available Information (PAI, in lack of GT) is available, then it can be specified (`-g groundtruth`). The format of GT or PAI files is very simple: GT files correlate IATA code seen by each vantage point (each line has two fields following a `hostname iata` format); PAI files only contain a list of `iata` airport codes, one per line, associated with the target (e.g., on a public webpage).

The output files (CSV and JSON) can be customized (`-o outputfile`), possibly triggering in-browser graphical rendering of the geographical map (`-b`). The code allows to tune the disk threshold (by default set to ∞) and the latency vs population weight α (by default set to $\alpha = 1$). Changing the default settings is discouraged from a performance and accuracy perspective, but it can be nevertheless useful for verification or understanding of the underlaying algorithm.

C. Examples: process historic measurement

We now provide a list of examples to process historic measurement released in the code. For instance, to run iGreedy on the F root server of the provided dataset:

```
./igreedy -i datasets/measurement/f-ripe
```

To run iGreedy over the F root server dataset, showing results on a map (opening your browser):

```
./igreedy -i datasets/measurement/f-ripe -b
```

To run iGreedy over the F root server dataset, showing results and ground truth on a map (opening your browser):

```
./igreedy -i datasets/measurement/f-ripe \
  -g datasets/ground-truth/f-ripe -b
```

To run iGreedy over the EdgeCast dataset measured (from RIPE Atlas), using publicly available information (from Webpages):

```
./igreedy -i datasets/measurement/edgecast-ripe \
 -g datasets/public-available-information/edgecast
```

To run iGreedy over the CloudFlare dataset measured (from PlanetLab), using ground truth information (from PlanetLab):

```
./igreedy -i datasets/measurement/cloudflare-planetlab \
 -g datasets/ground-truth/cloudflare-planetlab
```

D. Examples: run and process new measurements

The released version allows to run iGreedy from RIPE Atlas. The version we use internally allows to run measurement from RIPE Atlas or PlanetLab. As getting a RIPE Atlas key is much easier than obtaining, configuring and maintaining a PlanetLab slice (on which we surely wouldn't want to provide support or assistance), we do not intend to let the version of the code supporting multiple MIs public anytime soon (but feel free to contact us via email if you are an experienced PlanetLab user and want to profit of iGreedy results gathered on this platform as well).

Configuring RIPE Atlas key. Simply put your RIPE Atlas key in this file: `datasets/auth`. Voilà, iGreedy is now ready to run new RIPE Atlas measurement with your account.

Configuring RIPE Atlas vantage points. As the sensitivity in Sec.VII-C illustrated, iGreedy results are highly dependent on the vantage point selection: a bad vantage point selection (e.g., few vantage points, or bad geographical coverage) is expected to yield bad enumeration/geolocation results. Vantage points selection is made by modifying the content of the `datasets/ripe-vps` file. Depending on the aim (eg. detection, geolocation, census) each experiment may need a careful assessment of the tradeoff coverage-number of RIPE Atlas credits.

To simplify bootstrap, we provide two examples configurations that are rather conservative in the number of probes but useful for anecdotal use of detection (`datasets/ripe-vps.rand10`) and enumeration/geolocation (`ripe-vps.suggested200`). It is worth stressing that the set of RIPE Atlas vantage points used by default (`datasets/ripe-vps`) is conservative (10 random probes `ripe-vps.rand10`) and useful at most for detection (and to avoid burning all your credits with a single "for" loop. The set of `ripe-vps.suggested200` is again very conservative and useful for familiarizing with the tool before launching a measurement campaign. We count on releasing slightly larger but more accurate defaults on the website.

Hello, Anycast world. Finally, you are ready to run new measurement with iGreedy. For instance, to run iGreedy on the F root server 192.5.5.241,

```
./igreedy -m 192.5.5.241 -b
```

note that the set of new measurements is saved in `datasets/measurement` for further post-processing. Have fun! Danilo and Dario

GRAZIE

This work has been carried out at LINCS (http://www.lincs.fr), with funding from a Google Faculty Research Award and from the European Union under the FP7 Grant Agreement n. 318627 (Integrated Project "mPlane"). Results in this paper and open source dataset made available to the community would have not been possible without RIPE Atlas and PlanetLab.

REFERENCES

[1] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," IETF RFC 1546, 1993.
[2] D. Cicalese, J. Auge, D. Joumblatt, T. Freeman, and D. Rossi, "Characterizing IPv4 Anycast Adoption and Deployment," in *ACM CoNEXT*, 2015.
[3] http://www.edgecast.com.
[4] http://www.cloudflare.com.
[5] D. Madory, C. Cook, and K. Miao, "Who are the anycasters," Nanog, 2013.
[6] X. Fan, J. S. Heidemann, and R. Govindan, "Evaluating anycast in the domain name system." in *Proc. IEEE INFOCOM*, 2013.
[7] P. Boothe and R. Bush, "DNS Anycast Stability: Some Early Results," CAIDA, 2005.
[8] S. Sarat, V. Pappas, and A. Terzis, "On the use of anycast in DNS," in *Proc. ICCCN*, 2006.
[9] D. Karrenberg, "Anycast and bgp stability: A closer look at dnsmon data," Nanog, 2005.
[10] H. Ballani and P. Francis, "Towards a global IP anycast service," in *Proc. ACM SIGCOMM*, 2005.
[11] H. Ballani, P. Francis, and S. Ratnasamy, "A measurement-based deployment proposal for ip anycast." in *Proc. ACM IMC*, 2006.
[12] Z. Liu, B. Huffaker, M. Fomenkov, N. Brownlee, and K. C. Claffy, "Two days in the life of the DNS anycast root servers." in *Proc. of PAM*, 2007.
[13] M. Levine, B. Lyon, and T. Underwood, "Operational experience with TCP and Anycast," Nanog, 2006.
[14] A. Flavel, P. Mani, D. A. Maltz, N. Holt, J. Liu, Y. Chen, and O. Surmachev, "Fastroute: A scalable load-aware anycast routing architecture for modern cdns," *connections*, vol. 27, p. 19, 2015.
[15] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van der Merwe, "Anycast cdns revisited," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 277–286.
[16] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van Der Merwe, "A practical architecture for an anycast cdn," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 4, p. 17, 2011.
[17] Z. Al-Qudah, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van der Merwe, "Anycast-aware transport for content delivery networks," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 301–310.
[18] F. Streibelt, J. Böttger, N. Chatzis, G. Smaragdakis, and A. Feldmann, "Exploring edns-client-subnet adopters in your free time," in *Proc. ACM IMC*, 2013.
[19] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known content network (cn) request-routing mechanisms," Tech. Rep., 2003.
[20] M. Calder, E. Katz-Bassett, R. Mahajan, and J. Padhye, "Analyzing the performance of an anycast cdn," in *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015.
[21] D. Cicalese, D. Joumblatt, D. Rossi, M.-O. Buob, J. Augé, and T. Friedman, "A fistful of pings: Accurate and lightweight anycast enumeration and geolocation," in *Proc. IEEE INFOCOM*, 2015.
[22] http://www.enst.fr/~drossi/anycast.
[23] http://goo.gl/7ESrCR.
[24] R. Durairajan, S. Ghosh, X. Tang, P. Barford, and B. Eriksson, "Internet atlas: A geographic database of the internet," in *HotPlanet*, 2013.
[25] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of internet hosts." in *Proc. ACM IMC*, 2004.
[26] B. Eriksson and M. Crovella, "Understanding geolocation accuracy using network geometry," in *Proc. IEEE INFOCOM*, 2013.
[27] B. Eriksson, P. Barford, J. Sommers, and R. Nowak, "A learning-based approach for IP geolocation," in *Proc. of PAM*, 2010.
[28] Y. Shavitt and N. Zilberman, "A geolocation databases study," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 10, 2011.
[29] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "IP geolocation databases: Unreliable?" *ACM SIGCOMM CCR*, 2011.
[30] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, and R. Govindan, "Mapping the expansion of google's serving infrastructure," in *Proc. ACM IMC*, 2013.
[31] M. J. Freedman, K. Lakshminarayanan, and D. Mazières, "Oasis: Anycast for any service," in *Proc. USENIX NSDI*, 2006.
[32] D. Katabi and J. Wroclawski, "A framework for scalable global ip-anycast (gia)," in *Proc. ACM SIGCOMM*, 2000.
[33] L. Colitti, "Measuring anycast server performance: The case of k-root," Nanog, 2006.
[34] B. Barber, M. Larson, and M. Kosters, "Traffic source analysis of the j root anycast instances," Nanog, 2006.
[35] S. Woolf and D. Conrad, "Requirements for a Mechanism Identifying a Name Server Instance," RFC 4892 (Informational), 2007.
[36] https://www.maxmind.com.
[37] S. Laki, P. Mátray, P. Hága, T. Sebok, I. Csabai, and G. Vattay, "Spotter: A model based active geolocation service," in *Proc. IEEE INFOCOM*, 2011.
[38] http://www.caida.org/research/performance/measinfra/evaltable.xml.
[39] V. Bajpai and J. Schonwalder, "A survey on internet performance measurement platforms and related standardization efforts," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1313–1341, 2015.
[40] https://atlas.ripe.net.
[41] https://www.planet-lab.org.
[42] 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 IMC*. ACM, 2006, pp. 153–158.

[43] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, “Netalyzr: illuminating the edge network,” in *Proc. ACM IMC*, 2010.

[44] C. Pelsser, L. Cittadini, S. Vissicchio, and R. Bush, “From paris to tokyo: On the suitability of ping to measure latency,” in *Proc. ACM IMC*, 2013, pp. 427–432.

[45] C. Chirichella and D. Rossi, “To the moon and back: are internet bufferbloat delays really that large,” in *IEEE INFOCOM Workshop on Traffic Measurement and Analysis (TMA’13)*, April 14-19 2013.

[46] http://www.root-servers.org.

[47] S. Valenti, D. Rossi, A. Dainotti, A. Pescape, A. Finamore, and M. Mellia, “Reviewing traffic classification,” in *Data Traffic Monitoring and Analysis: From measurement, classification and anomaly detection to Quality of Experience*, M. M. Ernst Biersack, Christian Callegari, Ed. Heidelberg, Germany: Springer (LNCS 7754), 2013, ch. 6.

[48] http://www.cloudflarestatus.com.

[49] http://status.edgecast.com.

[50] http://status.verizondigitalmedia.com/.

[51] V. Giotsas, G. Smaragdakis, B. Huffaker, and M. Luckie, “Mapping peering interconnections to a facility,” in *ACM CoNEXT*.

[52] A. Singla, B. Chandrasekaran, P. B. Godfrey, and B. Maggs, “The internet at the speed of light,” in *ACM HotNets’XIII*, 2014.

[53] K. Jansen, “Approximation algorithms for geometric intersection graphs,” in *Graph-Theoretic Concepts in Computer Science*, 2007.

[54] T. Erlebach et al., “Polynomial-time approximation schemes for geometric intersection graphs,” in *SIAM Journal on Computing*, 2005.

[55] M. Linsner, P. Eardley, T. Burbridge, F. Sorensen, “Large-scale broadband measurement use cases,” *IETF RFC7536*, 2015.

[56] B. Trammell, P. Casas, D. Rossi, A. Bar, Z. Ben-Houidi, I. Leontiadis, T. Szemethy, and M. Mellia, “mPlane: an Intelligent Measurement Plane for the Internet,” *IEEE Commu. Mag.*, pp. 148–156, May 2014.

[57] P. Casoria, D. Rossi, J. Aug, M.-O. Buob, T. Friedman, and A. Pescape, “Distributed active measurement of internet queuing delays,” in *Proc. of PAM, Extended Abstract*, 2014.

[58] G. Huston, M. Rossi, and G. Armitage, “Securing BGP – A literature survey,” *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 199–222, 2011.

[59] “Detecting bgp hijacks in 2014,” in *NoSuchCon (NSC#2)*, 2014.

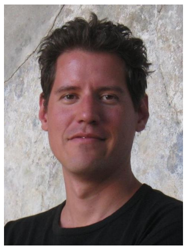
[60] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown, “A survey on network troubleshooting,” Technical Report TR12-HPNG-061012, Stanford University, Tech. Rep., 2012.

[61] R. Beverly and A. Berger, “Server siblings: Identifying shared ipv4/ipv6 infrastructure via active fingerprinting,” in *Passive and Active Measurement*, 2015, pp. 149–161.

[62] R. Ensafi, J. Knockel, G. Alexander, and J. R. Crandall, “Detecting intentional packet drops on the Internet via TCP/IP side channels,” in *Passive and Active Measurement*, 2014, pp. 109–118.

Danilo Cicalese received his MSc from Universitá Federico II, Naples, Italy. He is currently enrolled at a PhD program at Telecom ParisTech and UPMC. His research interests include traffic measurement and analysis.

Diana Zeaiter Joumblatt received her MSc and PhD degrees from Université Pierre et Marie Curie (UPMC) in 2008 and 2012 respectively. She was a postdoctoral researcher at Telecom ParisTech for 18 months starting February 2014. Her research interests include Internet measurements and user quality of experience.



Dario Rossi received his MSc and PhD degrees from Politecnico di Torino in 2001 and 2005 respectively, was a visiting researcher at University of California, Berkeley during 2003-2004, and is currently Professor at Telecom ParisTech and Ecole Polytechnique. He has coauthored over 150 conference and journal papers, received several best paper awards, a Google Faculty Research Award (2015) and an IRTF Applied Network Research Prize (2016). He is a Senior Member of IEEE and ACM.

Jordan Augé received his Engineering degree from ENSIIE in 2004, and his PhD from Telecom ParisTech jointly with Orange Labs in 2014. He held research engineer positions in University of Cambridge, UPMC/LIP6, and Telecom ParisTech. He is currently a PostDoc at Cisco Systems France working on Information Centric Networking and more particularly on the performance evaluation of mobile architectures.

Marc-Olivier Buob received his Engineering degree from ENSIIE in 2005, and his PhD from University of Angers jointly with Orange Labs in 2018. Since 2015, he is research engineer at Alcatel Lucent. His research fields include Internet measurements, routing protocols, path algebras and graph theory.

Timur Friedman received the AB degree in philosophy from Harvard University, the MSc and PhD degrees in computer science from the University of Massachusetts Amherst, in 1995 and 2001, respectively. He is currently an Associate Professor at Université Pierre et Marie Curie (UPMC), working on Internet traffic measurement and testbed federation.