Zeph & Iris map the internet: A resilient reinforcement learning approach to distributed IP route tracing

Matthieu Gouel, Kevin Vermeulen, Maxime Mouchet, Justin Rohrer, Olivier Fourmaux, Timur Friedman

Zeph & Iris Map the Internet

A resilient reinforcement learning approach to distributed IP route tracing

Matthieu Gouel[*†]
Sorbonne Université, France
matthieu.gouel@lip6.fr

Kevin Vermeulen[‡]
LAAS-CNRS, France
kevin.vermeulen@cnrs.fr

Maxime Mouchet[*†]
Sorbonne Université, France
maxime.mouchet@lip6.fr

Justin P. Rohrer[§]
Naval Postgraduate School, USA
jprohrer@nps.edu

Olivier Fourmaux[*]
Sorbonne Université, France
olivier.fourmaux@lip6.fr

Timur Friedman[*†]
Sorbonne Université, France
timur.friedman@lip6.fr

ABSTRACT

We describe a new system for distributed tracing at the IP level of the routes that packets take through the IPv4 internet. Our Zeph algorithm coordinates route tracing efforts across agents at multiple vantage points, assigning to each agent a number of /24 destination prefixes in proportion to its probing budget and chosen according to a reinforcement learning heuristic that aims to maximize the number of multipath links discovered. Zeph runs on top of Iris, our open-source system for orchestrating internet measurements across distributed agents of heterogeneous probing capacities. We show that carefully choosing which destination prefixes to probe from which vantage point matters for optimizing topology discovery and that a system can learn to improve its assignments based on previous measurements. After 10 cycles of probing, Zeph is capable of discovering 3.3M nodes and 19.8M links in a cycle of 15 hours, when deployed on 5 Iris agents. This is 3 times more nodes and 10 times more links than the existing state-of-the-art production system for the same number of prefixes probed.

CCS CONCEPTS

• **Networks** → **Network measurement**;

KEYWORDS

Active Internet Measurements; Internet Topology;

1 INTRODUCTION

Mapping the internet's topology has been a challenge for more than two decades. Topological knowledge underpins our understanding of issues such as security [39], connectivity [5, 33], and performance [26]. It also leads to better models of the internet that aid in the design of new protocols [38]. The IP-level internet topology, in particular, is an essential input for building other internet datasets [4], such as AS-relationships [19, 23], MPLS tunnel detection [15, 34], alias resolution [27, 29] and IP geolocation [24].

Systems that measure the topology at the IP level, i.e., the addresses and the traceroute [22] style links from one address to the next, face a tension between completeness of discovery and overhead in the number of probe packets sent. Recent high-speed probing tools Yarrp [8], Diamond-Miner [36], and Flashroute [21] are capable of tracing towards all routable IPv4 prefixes from a single vantage point at rates exceeding 100,000 packets per second (pps). Probing at these rates enables the creation of internet topology maps in a reduced time, avoiding staleness of data due to routing changes. However, very few probing agents with such capacity are available to the community; agents deployed on Ripe Atlas [32], PlanetLab Europe [1], or Archipelago (Ark) [10], for instance, are constrained either by machine resources or by operator policy. Nonetheless, probing from multiple vantage points can bring important marginal gains for topology discovery [7]. Therefore, so as to maximize total discovery given a set of vantage points, we are challenged to design an algorithm that intelligently allocates probing directives, i.e., the instructions to an agent to conduct a route trace towards a specific /24 prefix.

This paper presents Zeph, a reinforcement learning algorithm for allocating probing directives to agents, and Iris, the generic distributed measurement orchestration system that supports Zeph.

Our contributions are:

- The Zeph algorithm, that learns how to ameliorate the allocation to agents at multiple vantage points of the destination prefixes that each should probe (Sec. 4). We find that discoveries increase by 57% for the nodes and 90% for the links in 10 cycles of learning, showing the importance of selecting the right prefixes to probe from each vantage point. Moreover, our system discovers 3 times more nodes and 10 times more links for the same number of prefixes in a shorter period of time than the state-of-the-art system Ark [10] (Sec. 6).
- The Iris system, a robust and scalable measurement system, built from free open-source software, that supports multi-vantage-point measurement techniques, and that Zeph can ask to perform single-path or multipath route traces (Sec. 5).
- Publicly available code and data,[1] along with a production deployment of Zeph and Iris to serve the research community with data series and the ability to perform one's own measurements.[2] The results in this paper are fully reproducible on commercial cloud instances via the Jupyter notebooks that we provide.[3]

[*]Lip6 CNRS laboratory, 75005 Paris, France
[†]Lincs laboratory, 75013 Paris, France
[‡]LAAS-CNRS, Université de Toulouse, CNRS, 31400 Toulouse, France
[§]Tancad Lab, Monterey, CA 93943, USA

[1]Free open-source liberally licensed code: https://github.com/dioptra-io
[2]Datasets and measurement service for the research community: https://iris.dioptra.io
[3]Instructions for reproducing this work: https://github.com/dioptra-io/zeph-evaluation

2 RELATED WORK

Measurement systems that have been deployed to map and characterize the internet's IPv4 IP-level topology based on traceroute-like [22] measurements from distributed vantage points around the globe historically include Rocketfuel [31], Dimes [30], and iPlane [28]. Today's production systems include Ripe Atlas [32], which performs single-path Paris Traceroutes from its over 10,000 low-power hardware agents and software agents running at end user's homes and work premises; and M-Lab [17], which issues a multipath Paris Traceroute [35, 37] towards each client that performs an NDT test. Today's longest-running and most used production topology mapping system is Caida's Ark [10], which utilizes agents at over 110 vantage points to perform single-path Paris Traceroutes [6] to one random destination per routed /24 prefix.

Two longstanding challenges face any system that aims to maximize coverage of the internet's paths, i.e., to run at the scale of the entire IPv4 internet: how to trace more efficiently and more rapidly.

More efficient tracing. Doubletree [16] exploited path commonality to terminate measurements when they converged on a previously probed path. iPlane [28] used BGP data to aggregate destinations by common AS path, and Beverly et al. proposed a series of primitives for more efficient mapping (e.g., subnetting inferences) [9]. Our Zeph algorithm aims for a more efficient assignment of probe destinations to probing agents.

High speed topology discovery techniques. Yarrp [8] introduced high speed topology mapping: it encodes sufficient state in each probe packet for the returning ICMP replies, which contain the first bytes of the probe packets, to be self-identifying, thereby eliminating the need to probe slowly in order to associate replies with probes. Whereas Yarrp traces single paths from source to destination, Diamond-Miner [36] added multipath probing. Flashroute [21] reduces the overhead of Yarrp by first estimating the TTL, i.e., the hop count, to the destination. Our production system uses Diamond-Miner high speed probing.

More recent work has focused on the dynamics of internet routes and the challenge this presents to obtaining up-to-date data. Cunha et al. [14] designed DTrack to predict the stability of network paths, helping to determine when to initiate new path measurements and thereby optimize the probing budget. Giotsas et al. [18] designed techniques to detect traceroute staleness, reducing the number of traceroutes to reissue. By tracing quickly, our system aims to keep stale data to a minimum.

3 OVERVIEW

Iris is our new measurement platform (Sec. 5) that exposes a Rest API allowing a client to request that measurements such as Ping, Yarrp [8], or Diamond-Miner [36] be performed from distributed agents, and to obtain the measurement results. We have implemented the Zeph algorithm (Sec. 4) as a Python-based client that pilots Iris so as to survey the IPv4 IP-level topology of the internet, learning to improve the probing directives that it issues over the course of successive cycles of measurements.

4 ZEPH SCHEDULING ALGORITHM

In designing Zeph, we reasoned that the relatively static nature of much routing in the internet [18] would allow a topology discovery system to learn to achieve good node and link coverage on the basis of its experience with the results of its own probing directives (Sec. 4.1). But experience would not be a completely reliable guide due to the existence of routing changes and their unpredictable nature [13, 14, 18], meaning that continuous exploration of new directives would also be required. In considering learning systems that combine experience with exploration so as to select among many choices that offer uncertain rewards, we turned to reinforcement learning [25]. Zeph proceeds across a series of cycles, with the directives for each cycle being based in large part upon the success of the directives that were used in the previous cycle. New directives are also tried out each cycle, with the overall aim being to improve the completeness of coverage over successive cycles. In reinforcement learning terms, the reuse of directives is *exploitation* and the trying out of new directives is *exploration*.

Zeph's challenge is to best use the probing budgets of its agents, i.e., choose which directives to issue to each agent so as to obtain the overall most complete route trace picture possible within a cycle. There are many ways of conceiving of "completeness", and the one adopted here is to maximize coverage of the traceroute-style directed links that are available to be discovered, a directed link consisting in an ordered pair of IP addresses.

Alg. 1 describes the high-level loop of the Zeph algorithm. The parameter ϵ specifies the minimum portion of each agent's probing budget that is set aside for exploration. Each cycle i of Zeph

Algorithm 1: Zeph

Input: ϵ : Fraction of budget per agent reserved for exploration

```
1 for i = 1 to ∞ do
2     R_{i-1} ← ResultsFromPreviousCycle(Iris)
3     A_i ← AgentsWithBudgets(Iris)
      /* Assign exploitation directives to agents A_i          */
4     if i > 1 then
5         D_i ← Exploitation(A_i, R_{i-1})
6     𝔻_i ← PossibleExplorationDirectives(Iris)
7     for a in A_i do
          /* Assign exploration directives to agent a          */
8         D_{i,a} ← Exploration(D_i, 𝔻_{i,a}, ε)
9     Probe(Iris, D_i)
```

involves a series of interactions with the Iris API, culminating (line 9) with Zeph sending Iris a collection of probing directives D_i. These directives are prepared on the basis of the results R_{i-1} of the preceding cycle of probing (line 2). Following the first cycle, which consists in random directives, there are previous results to build upon, and Zeph starts assembling the collection of probing directives on the basis of exploiting those results (line 5). Once the exploitation directives have been assigned, Zeph proceeds agent by agent to round out the assignments with exploration directives (line 8). The remainder of this section describes these steps in detail.

4.1 Agents, directives and results

In each of its cycles i, Zeph instructs Iris's available agents A_i to follow a collection of probing directives D_i the size of which depends on the agent's probing budget. It receives from Iris in return a collection of results, which appears in the subsequent cycle, after i has been incremented, as R_{i-1}.

As Zeph operates over days and weeks, we can expect that the set of Iris agents that are available to it will vary over time; some agents will go down, and others will arrive. As it begins each cycle i, Zeph queries Iris for the currently available set of agents, A_i, along with their associated probing budgets (line 3).

The universe $\mathbb{D}_i$ of possible exploration directives that Zeph obtains from Iris for each cycle i (line 6) potentially includes all /24 prefixes extracted from all public unicast IPv4 address blocks. So as to avoid sending unnecessary probes towards non-routed prefixes, we restrict Zeph to blocks obtained from Oregon Route Views [2].

Zeph only considers ICMP Time Exceeded replies as relevant to its results, as our survey focuses on routing infrastructure, not end-systems. Individual probe replies are assembled into traceroute-style directed links, which are pairs (v_1, v_2) of IPv4 addresses. In the event that one of the two probes did not receive a reply (in common traceroute parlance, a *star*), $(v_1, \texttt{null})$ and $(\texttt{null}, v_2)$ are also considered to be valid links. These links are matched with their initial directives, so that the results that Zeph receives consist of sets of links, each set associated with the agent and the directive that resulted in its being discovered.

4.2 Exploitation

If routing and routers' readiness to reply to probes were to remain unchanged from one cycle to the next, having an agent repeating its directives from the previous cycle would cause it to discover precisely the same set of links as before. Even under these ideal conditions, any given link might be discovered by multiple agents and it might be discovered multiple times by the same agent, and this redundancy potentially leaves room for improvement, as is well known from earlier route tracing work [16, 20]. If the same results can be obtained by executing fewer directives, a portion of some agents' probing budgets can be redirected towards trying out new directives. Zeph therefore sorts each agent's directives, giving highest priority to the directives that, in the prior round, are judged by a heuristic to have made the greatest contribution to link discovery. The aim of this sorting is to discard directives that the heuristic judges to make little or no marginal contribution.

From the results of the previous cycle R_{i-1}, Zeph considers only the agents that are available for the current cycle i. For each such agent, it sorts the directives, and the set of sorted directives for all agents in A_i constitutes the collection D_i (line 5). It uses the Cormode et al. Disk-Friendly Greedy algorithm (DFG) [11] as the heuristic means of sorting the directives, which is an approximation of the greedy algorithm to solve the set cover problem for large datasets. At each iteration, instead of selecting the directive that covers the most uncovered results (classic greedy algorithm), DFG groups the directives into buckets of similar size. Starting from the bucket with the directives that provide the biggest number of results, if the number of results of a directive added to the cover set of results is greater than a parametrized threshold, the results are added to C and the directive is added to D_i. DFG terminates when all prior results have been covered, i.e., $C = R_{i-1}$.

4.3 Exploration

When the time comes for Zeph to designate the exploration directives, the agents that were present in the previous cycle will each have an ordered set of exploitation directives that were assigned by the heuristic described above. To round out these directives, and to assign a complete set of directives to agents that were not present in the previous round, Zeph calls upon the universe of possible exploration directives $\mathbb{D}_i$ (line 6). As previously described, this consists in all of the routable /24 IPv4 prefixes. As opposed to the heuristic employed for choosing exploitation directives, where knowledge about previous results allows a directive chosen for one agent to preclude the choice of a directive for another agent, the exploration choices are made in relative ignorance of their consequences, and are therefore conducted agent by agent, considering each agent $a \in A_i$ separately (line 7).

The parameter ϵ enters into play here, to ensure that this portion of each agent's directives are reserved for exploration. If, perchance, the portion of directives assigned for exploitation exceeds $1 - \epsilon$, a sufficient number of lowest priority exploitation directives are removed to make room for exploration.

Having rounded out the probing budget of each agent with exploration directives (line 8), the cycle's collection of directives $D_i = \bigcup_{a \in A_i} D_{i,a}$ is ready to be sent to Iris (line 9).

5 IRIS MEASUREMENT PLATFORM

The Iris system, shown in Fig. 1, allows us to run Zeph, but it has been designed to run any sort of internet measurement algorithm that requires access to geographically distributed probing agents. At the moment, three tools are available in Iris: Diamond-Miner [36], Yarrp [8], and Ping. More tools can easily be added in the future. Moreover, Iris works the same with IPv4 and IPv6 addresses.

Workflow. A client such as Zeph submits an HTTPS request to run a measurement via Iris's Rest API. The API informs the message broker Redis of the measurement request. The message broker chooses an inactive worker from an available pool. This worker maintains the state of the measurement throughout its life in the system. The worker registers the measurement parameters in the ClickHouse database and asks the agents to perform the measurement. When the measurement is completed by an agent, it sends the results to an object storage MinIO, an open-source alternative to Amazon S3. Then the worker pulls the results from the object storage and inserts the results into the database. When the measurement is finished, the worker updates the measurement state to mark it as ready to be pulled by the client. All of the components described above generate logs that are stored with a monitoring stack built from the combination of Prometheus for storing the system's metrics, Loki for storing the system's logs, and Grafana to allow visualizations of these metrics and logs.

Fig. 1 shows this workflow visually. Arrows symbolize the connections between the components (e.g., the worker connects to the database). Purple shows dataplane flow (e.g., compressed CSV files)

while green shows control plane flow (e.g., measurement parameters to execute a measurement tool on an agent). Yellow shows log and metric collection.

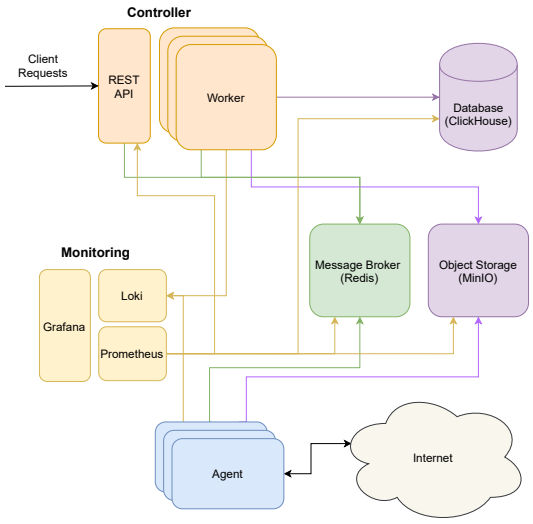


Figure 1: Iris architecture with arrows indicating which component initiates each connection; each box is a self-contained Docker container. Colors indicate the type of data flows: purple for dataplane, green for control plane, and yellow for logs.

5.1 Design considerations

The Iris was designed to meet the following demands:

(1) As Zeph measurements can last for days, Iris needs to be robust to agents crashing, or being unreachable, and provide the ability to restart an agent's measurements if and when it returns (*resilience*).
(2) Zeph will work from as many vantage points as it can access, so Iris should scale well with the number of agents (*agent scalability*).
(3) Zeph supports internet scale high speed topology measurement techniques, generating billions of ICMP replies. Iris should scale well with this amount of data (*data scalability*)
(4) We continue to improve Zeph, so Iris should support easy deployment of new control algorithms and probing software (*continuous delivery*).
(5) To improve affordability and maintainability, Iris should be built as much as possible from free open-source software (*maintainability*).

We chose Docker, and Redis to improve the resilience of the system: each of Iris's components runs in its own Docker container. Moreover, a failed container is automatically reintegrated without external intervention and retrieves the measurements states from Redis. Two aspects of Redis improve the message broker's resilience. First, its *persistence* feature regularly saves its own current state to disk, preserving context in case of failures and restarts. Second, it maintains connection state, which Iris uses to alert workers to the departure of any agent. Any worker with an ongoing measurement stops waiting for that agent to send data, and no further measurements can be requested of the agent until it reconnects to the broker. A new worker is automatically created from the worker pool to carry on the measurement algorithm from the point where the failed worker had left it.

We chose Redis and MinIO for agent scalability: Redis can handle millions of simultaneous connections; MinIO has proven capable, with Zeph, of supporting transfer of data files in the hundreds of gigabytes;

We chose the ClickHouse database for data scalability: it is a database optimized for insert and read operations, which are the only operations Iris perform on the result data. ClickHouse has supported tables containing up to 5 billion rows, and the in-base calculation language provided by ClickHouse's *arrays* feature reduces computation times.

Redis, MinIO and ClickHouse scale "horizontally", meaning that it is possible to deploy multiple message brokers, multiple object storage containers, and multiple replicas of the database to support a larger number of agents. Scaling beyond one instance each was not necessary in order for Iris to support Zeph, but the potential is there.

We chose Docker for continuous delivery: As we move forward, this will ease large-scale deployment by lifting many constraints on the machines and VMs that can host an Iris agent. Also, Iris takes advantage of such containers' ability to run unchanged over a wide variety of operating systems.

Finally, all the components of the system are free and open source, improving maintainability.

6 EVALUATION

There are three main results: (1) Zeph, requesting Diamond-Miner multipath route traces from Iris, provides the most comprehensive view to date of the IPv4 internet in terms of nodes and links discovered in a short period of time. Once it has been trained, a single cycle of Zeph discovers more than 3 times as many nodes and 10 times as many links as does a single cycle of the state-of-the-art Ark platform (Sec. 6.3.2); (2) When compared on single-path route traces, as performed by Ark, Zeph's reinforcement learning approach outperforms Ark's random exploration strategy (Sec. 6.2.1); (3) Zeph saves 50% of the probing budget, dividing the probing time by 2, compared to an exhaustive strategy of tracing multipath routes towards every prefix from every agent (Sec. 6.3), while maintaining nearly the same number of discoveries.

6.1 Vantage points and setup

All of the measurements are run on the EdgeNet [12] platform with nodes hosted in 5 different Google Compute Engine (GCE) zones: `asia-east2-a` (Hong Kong), `asia-northeast1-a` (Tokyo), `asia-southeast1-a` (Singapore), `europe-west6-a` (Zurich) and `southamerica-east1-a` (São Paulo). The measurements can be fully reproduced on nodes in the same locations by applying our open-source evaluation code. The instances use the *standard* network tier which allows the packets to exit to the internet as soon as possible, instead of the default *premium* tier which privileges Google's internal network to the public internet.

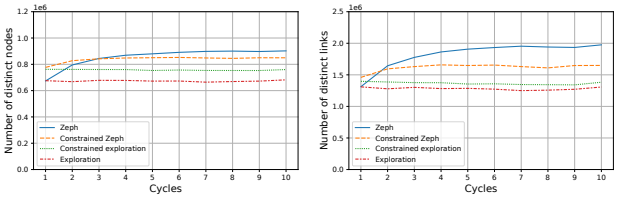
6.2 Topology discovery

We perform two experiments to evaluate Zeph's performance: (1) a comparison of Zeph's prefix allocation strategy with other approaches, performed from the same agents, with common parameter settings; (2) a comparison of the maximum raw discoveries by Zeph on Iris against Ark system for the same number of prefixes probed and the same number of probes sent.

6.2.1 Zeph's reinforcement learning approach outperforms random allocation. We compare Zeph's prefix allocation strategy with that used by the state-of-the-art topology discovery system Ark [10]. Ark applies what we call *constrained random allocation*, which consists in allocating the /24 prefixes of the IPv4 space uniformly at random over the different agents, each prefix being probed by exactly one agent (the constraint). Zeph's reinforcement learning approach splits the probing directives of each agent into two subsets for exploitation (Sec. 4.2) and exploration (Sec. 4.3). Zeph exploration differs from the Ark's random allocation by allowing a same prefix to be probed by multiple agents. We call Zeph's exploration strategy *unconstrained random allocation*.

Experiment. We run 10 cycles of different combinations of probing techniques simultaneously: exploitation and unconstrained random allocation (Zeph with $\epsilon = 0.1$), constrained random allocation (Ark's approach), unconstrained random allocation, and exploitation and constrained random allocation ($\epsilon = 0.1$). Unconstrained random allocation allows us to evaluate Zeph's exploration alone, while exploitation and constrained random allocation allow us to evaluate Zeph's exploration in combination with exploitation.

On August 4th, 2021, we extracted 11.9M /24 prefixes from the routed prefixes provided by Route Views. Dividing these among the 5 agents, at each cycle, and for all four approaches, each agent probes 2.4M prefixes. We run the approaches simultaneously and the agent for each approach probes at 100,000 packets per second using single path tracing [8] with ICMP probes up to TTL 32. Each cycle takes around 15 minutes to complete, running from August 5th to 6th, 2021.

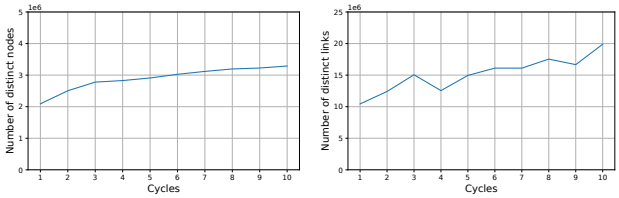


(a) Nodes discovered **(b) Links discovered**

Figure 2: Zeph outperforms random exploration strategies.

Results. Fig. 2 shows the results of the different strategies. The main result is that exploitation together with exploration using unconstrained random allocation (Zeph's approach) outperforms all of the other approaches once Zeph has had ten cycles during which to learn. In particular, it discovers 18% more nodes and 42% more links than constrained random allocation (Ark's approach). The other result is that constrained random allocation outperforms unconstrained random allocation by 12% more nodes and 6% more links over each cycle. This highlights that it is the combination of the exploitation and the unconstrained allocation together that allows Zeph's strategy to perform well.

6.2.2 Zeph/Iris conducting multipath traceroutes perform competitively with respect to the current state-of-the-art internet scale topology discovery system. In the previous section, we have shown that Zeph's prefix allocation strategies outperforms others when limited to the same single-path probing budget. But Zeph and Iris are capable of discovering more than what is shown in Fig. 2. To use their full capacity, we perform 10 cycles of measurement with multipath route traces (i.e., capturing the load-balanced paths) obtained by Diamond-Miner [36] with ICMP probes at 100,000 pps. We retrieved routed prefixes from Route Views on January 21st, 2022 and broke them down into 12M /24 prefixes, each of the five agents receiving a per-cycle budget of 2.4M /24 prefixes. The measurements were gathered from January 22th to 28th, 2022. Each cycle took between 10 hours, 12 minutes (cycle 1) and 15 hours, 14 minutes (cycle 10) to complete.



(a) Nodes discovered **(b) Links discovered**

Figure 3: Node and link discoveries in multipath probing

Fig. 3 shows that Zeph also works with multipath traceroutes: the number of nodes improves by 57% (+1,2M) the number of links improves by 90% (+9.4M) between cycle 1 and cycle 10. Note that an agent crashed at cycle 4, reducing the number of links found in that cycle, but Zeph adapted and the discoveries resumed increasing in cycle 5.

Table 1: Comparison of node and link discoveries between the tenth cycle of Zeph + Iris, and a cycle of the Ark platform for the same number of prefixes probed (12 million, second row) and the same number of probes sent (12 billion, third row).

	Time	Nodes	Links
Zeph + Iris	15h15	3,288,325	19,890,422
Ark (prefixes)	19h12	1,009,738	2,087,903
Ark (probes)	45 days	3,597,042	9,241,146

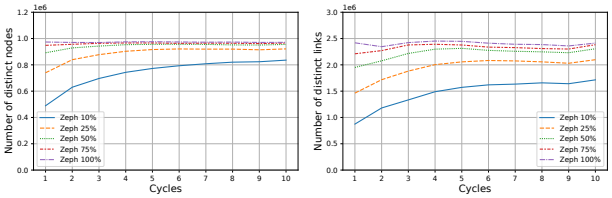
Finally, we compare raw discoveries of Zeph + Iris and Ark platform. Tab. 1 shows the number of nodes and links discovered during the last Zeph cycle and the number and nodes and links discovered by CAIDA's Ark [3] platform (1) when the same number of prefixes is probed, and (2) when the same number of probes is sent. Zeph with Iris takes 20% less time to probe all routed /24

prefixes but discovers more than 3 times more nodes and 10 times more links. Moreover, Ark discovers 10% more nodes and around two times fewer links with the same number of probes sent, but takes 45 days instead of less than 15 hours to do so.

6.3 Zeph probe savings

This section describes the tradeoff between reducing the number of prefixes probed by each agent (and therefore reducing the duration of a cycle and the number of probes sent) and discovery.

6.3.1 Experiment. We collect 10 cycles of 4 Zeph runs where each agent probes 10%, 25%, 50% or 75% of the routed /24 prefixes. In addition, we run an *exhaustive* measurement, with all the agents probing 100% of the routed /24 prefixes. The measurements for each budget were performed from August 2nd to 4th, 2021. We use the same Route Views data as in Sec. 6.2. Each agent runs Yarrp at 100,000 pps with ICMP probes. Depending on the budget, each cycle lasts between 10 minutes and 1 hour and 30 minutes. In this experiment, we reduce the number of prefixes that are probed by each agent to simulate a more constrained budget but keep a high probing rate. We could also reduce the probing rate, but this would have significantly increased the time of the experiment.



(a) Nodes discovered **(b) Links discovered**

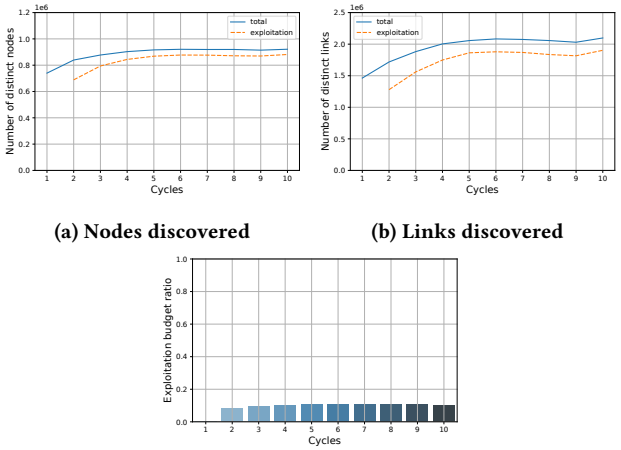
Figure 4: Number of nodes and links discovered over Zeph cycles for different budgets: Zeph with 50% finds almost the same number of nodes and links as with 100% of the budget.

6.3.2 Results. Fig. 4 shows the number of nodes and links discovered by all the agents together for the different budgets. The main result is that Zeph discovers 98% of the nodes and 95% of the links that the exhaustive approach discovers when just 50% of the prefixes are probed by the agents. The 25% (10%) curves show that even with a significantly reduced probing budget, Zeph is able to discover 94% (87%) of the nodes and 86% (71%) of the links. Additionally, reducing the number of probes also reduces the time of a cycle. With 50% (25%, 10%) of the prefixes, 10 cycles took 7.4 hours (3.6, 1.5), compared to the 12.5 hours of the exhaustive approach.

6.4 Reinforcement learning analysis

Finally, we dive into one measurement of Sec. 6.3 where the budget is 25% of the routed prefixes, to understand more about the contributions of exploitation and exploration.

6.4.1 Exploitation and exploration budgets. In choosing a reinforcement learning approach for Zeph, we anticipated that many of each agent's directives could be repeated (exploitation) from one cycle to the next, and that complementing these directives with new ones (exploration) would aid in improving overall discovery. We find that the exploitation directives were indeed capable of discovering most of the links previously discovered, and that exploration did indeed lead over time to better overall discovery.



(a) Nodes discovered **(b) Links discovered**

(c) Fraction of the budget allocated to exploitation

Figure 5: Exploitation counts for 95% of the nodes and 90% of the links discoveries.

Fig. 5 shows the number of nodes (Fig. 5a) and links (Fig. 5b) that are discovered by exploitation and in total. The main result is that reinforcement learning works: exploitation improves over time, going from 739k nodes at cycle 2 to 921k nodes at cycle 10 (+25%) and 1.2M links at cycle 2 to 1.9M links at cycle 10 (+48%). In Fig. 5c, exploitation is responsible for most of the discoveries, i.e., 95% of the nodes and 90% of the links. Interestingly, although we allocated 90% of the budget to exploitation, Zeph actually used only 10% for it. We interpret this result as a consequence of the high redundancy of internet paths, and leave the study of the optimal value of ϵ for future work.

7 CONCLUSION AND FUTURE WORK

Zeph is a new algorithm for distributed tracing at the IP level of the routes that packets take through the IPv4 internet. It learns the probing directives to allocate to the vantage points in order to maximize topology discovery. Zeph is platform agnostic, and independent of the probing tool used and the agent's capacities.

Iris is a distributed internet measurement system based on a modern resilient architecture that exposes an API that allows various algorithms, including Zeph, to be run. Together, Zeph and Iris discover 3 times more nodes and 10 times more links than the state-of-the-art Ark platform for the same number of prefixes probed.

All of the code of Iris and Zeph and data of the evaluation are publicly accessible and we now offer regular Zeph internet topology data series to the community and the ability to perform one's own measurements.

In future work, we will extend Zeph to IPv6 and analyze in depth the dynamics of the internet topology.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers and the *CCR* editor, Steve Uhlig. Ethan Katz-Bassett and Robert Beverly were very helpful with their suggestions. Jack Brassil helped through his support for EdgeNet. Kevin Vermeulen completed some of this work at Columbia University. Support for Matthieu Gouel, Maxime Mouchet, Olivier Fourmaux, and Timur Friedman came in part from a French Ministry of Defense university research grant; for Kevin Vermeulen from U.S. National Science Foundation (NSF) grant Nº 1836872; and for Justin P. Rohrer from NSF grant Nº CNS-1855614. The views and conclusions are those of the authors and should not be interpreted as representing the official policies or position of either the French government or the U.S. government; or the French Ministry of Defense or the U.S. National Science Foundation.

REFERENCES

[1] [n. d.]. PlanetLab Europe. https://planet-lab.eu/. ([n. d.]). Accessed February 2, 2022.

[2] 2004. Oregon Route Views. http://routeviews.org/. (2004). June 8, 2004; accessed February 2, 2022.

[3] 2008. The CAIDA UCSD IPv4 Routed /24 Topology Dataset. https://www.caida.org/catalog/datasets/ipv4_routed_24_topology_dataset/. (2008). February 1, 2008; version of July 8, 2020.

[4] 2014. The Impact of the Archipelago Measurement Platform. https://www.caida.org/projects/ark/impact/. (2014). July 3, 2014; version of November 15, 2019.

[5] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. 2012. Anatomy of a Large European IXP. In *Proc. ACM SIGCOMM Conf. (SIGCOMM '12)*. https://doi.org/10.1145/2342356.2342393

[6] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. 2006. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '06)*. https://doi.org/10.1145/1177080.1177100

[7] Paul Barford, Azer Bestavros, John Byers, and Mark Crovella. 2001. On the Marginal Utility of Network Topology Measurements. In *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW '01)*. https://doi.org/10.1145/505202.505204

[8] Robert Beverly. 2016. Yarrp'ing the Internet: Randomized High-Speed Active Topology Discovery. In *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '16)*. https://doi.org/10.1145/2987443.2987479

[9] Robert Beverly, Arthur Berger, and Geoffrey G. Xie. 2010. Primitives for Active Internet Topology Mapping: Toward High-Frequency Characterization. In *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '10)*. https://doi.org/10.1145/1879141.1879162

[10] Kimberly Claffy, Young Hyun, Ken Keys, Marina Fomenkov, and Dmitri Krioukov. 2009. Internet Mapping: From Art to Science. In *Proc. 2009 Cybersecurity Applications Technology Conf. for Homeland Security (CATCH)*. https://doi.org/10.1109/CATCH.2009.38

[11] Graham Cormode, Howard Karloff, and Anthony Wirth. 2010. Set Cover Algorithms for Very Large Datasets. In *Proc. ACM Intl. Conf. on Information and Knowledge Management (CIKM '10)*. https://doi.org/10.1145/1871437.1871501

[12] Berat Can Şenel, Maxime Mouchet, Justin Cappos, Olivier Fourmaux, Timur Friedman, and Rick McGeer. 2021. EdgeNet: A Multi-Tenant and Multi-Provider Edge Cloud. In *In Proc. ACM Intl. Workshop on Edge Systems, Analytics and Networking (EdgeSys '21)*. https://doi.org/10.1145/3434770.3459737

[13] Italo Cunha, Pietro Marchetta, Matt Calder, Yi-Ching Chiu, Bruno V. A. Machado, Antonio Pescapè, Vasileios Giotsas, Harsha V. Madhyastha, and Ethan Katz-Bassett. 2016. Sibyl: A Practical Internet Route Oracle. In *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI '16)*.

[14] Ítalo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. 2014. DTRACK: A System to Predict and Track Internet Path Changes. *IEEE/ACM Trans. on Networking* 22, 4 (2014), 1025–1038. https://doi.org/10.1109/TNET.2013.2269837

[15] Benoit Donnet, Matthew Luckie, Pascal Mérindol, and Jean-Jacques Pansiot. 2012. Revealing MPLS Tunnels Obscured from Traceroute. *ACM SIGCOMM Computer Communications Rev.* 42, 2 (Mar. 2012), 87–93. https://doi.org/10.1145/2185376.2185388

[16] Benoit Donnet, Philippe Raoult, Timur Friedman, and Mark Crovella. 2005. Efficient Algorithms for Large-Scale Topology Discovery. In *Proc. ACM SIGMETRICS Conf. (SIGMETRICS '05)*. https://doi.org/10.1145/1064212.1064256

[17] Constantine Dovrolis, Krishna Gummadi, Aleksandar Kuzmanovic, and Sascha D. Meinrath. 2010. Measurement Lab: Overview and an Invitation to the Research Community. *ACM SIGCOMM Computer Communications Rev.* 40, 3 (Jun. 2010), 53–56. https://doi.org/10.1145/1823844.1823853

[18] Vasileios Giotsas, Thomas Koch, Elverton Fazzion, Ítalo Cunha, Matt Calder, Harsha V. Madhyastha, and Ethan Katz-Bassett. 2020. Reduce, Reuse, Recycle: Repurposing Existing Measurements to Identify Stale Traceroutes. In *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '20)*. https://doi.org/10.1145/3419394.3423654

[19] Vasileios Giotsas, Matthew Luckie, Bradley Huffaker, and kc claffy. 2014. Inferring Complex AS Relationships. In *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '14)*. https://doi.org/10.1145/2663716.2663743

[20] R. Govindan and H. Tangmunarunkit. 2000. Heuristics for Internet map discovery. In *Proc. IEEE INFOCOM '00*. https://doi.org/10.1109/INFCOM.2000.832534

[21] Yuchen Huang, Michael Rabinovich, and Rami Al-Dalky. 2020. FlashRoute: Efficient Traceroute on a Massive Scale. In *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '20)*. https://doi.org/10.1145/3419394.3423619

[22] Van Jacobson. 1988. 4BSD routing diagnostic tool available for ftp. Email 8812201313.AA03127@helios.ee.lbl.gov to the IETF and end2end-interest e-mail lists. (1988).

[23] Yuchen Jin, Colin Scott, Amogh Dhamdhere, Vasileios Giotsas, Arvind Krishnamurthy, and Scott Shenker. 2019. Stable and Practical AS Relationship Inference with ProbLink. In *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI '19)*.

[24] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. 2006. Towards IP Geolocation Using Delay and Topology Measurements. In *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '06)*. https://doi.org/10.1145/1177080.1177090

[25] Volodymyr Kuleshov and Doina Precup. 2014. Algorithms for multi-armed bandit problems. (2014). arXiv:1402.6028

[26] Matthew Luckie and Robert Beverly. 2017. The Impact of Router Outages on the AS-Level Internet. In *Proc. ACM SIGCOMM Conf. (SIGCOMM '17)*. https://doi.org/10.1145/3098822.3098858

[27] Matthew Luckie, Robert Beverly, William Brinkmeyer, and kc claffy. 2013. Speedtrap: Internet-Scale IPv6 Alias Resolution. In *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '13)*. https://doi.org/10.1145/2504730.2504759

[28] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. 2006. IPlane: An Information Plane for Distributed Services. In *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*.

[29] Pascal Mérindol, Benoit Donnet, Jean-Jacques Pansiot, Matthew Luckie, and Young Hyun. 2011. MERLIN: MEasure the router level of the INternet. In *Proc. Conference on Next Generation Internet Networks (EURO-NGI '11)*. https://doi.org/10.1109/NGI.2011.5985865

[30] Yuval Shavitt and Eran Shir. 2005. DIMES: Let the Internet Measure Itself. *ACM SIGCOMM Computer Communications Rev.* 35, 5 (Oct. 2005), 71–74. https://doi.org/10.1145/1096536.1096546

[31] Neil Spring, Ratul Mahajan, and David Wetherall. 2002. Measuring ISP Topologies with Rocketfuel. In *Proc. ACM Sigcomm Conf. (SIGCOMM '02)*. https://doi.org/10.1145/633025.633039

[32] RIPE NCC Staff. 2015. RIPE Atlas: A Global Internet Measurement Network. *The Internet Protocol Journal* 18, 3 (Sept. 2015), 2–26. http://ipj.dreamhosters.com/wp-content/uploads/2015/10/ipj18.3.pdf

[33] James P.G. Sterbenz, Egemen K. Çetinkaya, Mahmood A. Hameed, Abdul Jabbar, Shi Qian, and Justin P. Rohrer. 2011. Evaluation of network resilience, survivability, and disruption tolerance: Analysis, topology generation, simulation, and experimentation. *Telecommunication Systems* 52, 2 (Dec. 2011), 705–736. https://doi.org/10.1007/s11235-011-9573-6

[34] Yves Vanaubel, Jean-Romain Luttringer, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. 2019. TNT, Watch me Explode: A Light in the Dark for Revealing MPLS Tunnels. In *Proc. Network Traffic Measurement and Analysis Conference (TMA '19)*. https://doi.org/10.23919/TMA.2019.8784525

[35] Darryl Veitch, Brice Augustin, Renata Teixeira, and Timur Friedman. 2009. Failure control in multipath route tracing. In *Proc. IEEE INFOCOM '09*. https://doi.org/10.1109/INFCOM.2009.5062055

[36] Kevin Vermeulen, Justin P. Rohrer, Robert Beverly, Olivier Fourmaux, and Timur Friedman. 2020. Diamond-Miner: Comprehensive Discovery of the Internet's Topology Diamonds. In *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*. https://www.usenix.org/conference/nsdi20/presentation/vermeulen

[37] Kevin Vermeulen, Stephen D. Strowes, Olivier Fourmaux, and Timur Friedman. 2018. Multilevel MDA-Lite Paris Traceroute. In *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC '18)*. https://doi.org/10.1145/3278532.3278536

[38] Walter Willinger, David Alderson, and John C Doyle. 2009. Mathematics and the internet: A source of enormous confusion and great potential. *Notices of the American Mathematical Society* 56, 5 (2009), 586–599. https://www.ams.org/notices/200905/rtx090500586p.pdf

[39] Zheng Zhang, Ying Zhang, Y. Charlie Hu, Z. Morley Mao, and Randy Bush. 2008. Ispy: Detecting Ip Prefix Hijacking on My Own. *ACM SIGCOMM Computer Communications Rev.* 38, 4 (Aug. 2008), 327–338. https://doi.org/10.1145/1402946.1402996