Extracting Routing Events from Traceroutes: a Matter of *Empathy*

Marco Di Bartolomeo, Valentino Di Donato, Maurizio Pizzonia, Claudio Squarcella, Massimo Rimondini

***Abstract*—With the increasing diffusion of Internet probing technologies, a large amount of regularly collected traceroutes are available for Internet Service Providers (ISPs) at low cost. We show how it is possible, given solely an arbitrary set of traceroutes, to spot routing paths that change similarly over time and aggregate them into inferred routing events. With respect to previous works, our approach does not require any knowledge of the network, does not need complex integration of several data sources, and exploits the asynchronicity of measurements to accurately position events in time. The formal model at the basis of our methodology revolves around the notion of *empathy*, a relation that binds similarly behaving traceroutes. The correctness and completeness of our approach are based on structural properties that are easily expressed in terms of empathic measurements. We perform experiments with data from public measurement infrastructures like RIPE Atlas, showing the effectiveness of our algorithm in distilling significant events from a large amount of traceroute data. We also validate the accuracy of the inferred events against ground-truth knowledge of routing changes originating from induced and spontaneous routing events. Given these promising results, we believe our methodology can be an effective aid for troubleshooting at the ISPs level. The source code of our algorithm is publicly available at https://github.com/empadig.**

***Index Terms*—Internet routing, network monitoring, traceroute, empathy, root-cause analysis, routing events, Internet measurement platforms.**

I. Introduction

One of the primary goals of a network operator is to ensure its network works as expected. Since misbehaviors can happen for a variety of reasons, constant monitoring is performed by operators to timely detect problems and limit users complaints. Directly monitoring the health of each network element works in many situations, but may fall short when the element itself lacks the necessary agent support or is not under the operator's control.

Many works show methods to detect and localize faults, like disconnections or service degradation. The spectrum of the proposed solutions is quite broad: they can focus on enterprise networks, Internet backbone or access networks, they can exploit control-plane and/or data-plane sources, they can exploit active and/or passive measurements, etc. The quality of these solutions is usually evaluated by estimating their capability not to miss occurred events and not to report fictitious events. To reduce errors, a typical approach is to increase the information to analyze considering several kinds of data sources and to apply several detection techniques at the same time. However, this comes with an increase in complexity of the whole system.

Currently, there is a growing trend in the network measurement landscape: the availability of Internet measurement platforms, which are infrastructures composed of a large number of small low-cost devices (called *probes* or *monitors*) that can independently perform several kinds of measurements towards a large number of targets. These devices have been deployed all over the world at strategic locations within access and backbone networks and behind residential gateways [2]. This trend is also supported by a standardization process [3] and by an increasing request from regulators for reliable third-party measurements [4]. These projects are designed to scale on the amount of deployed probes. Hence, they provide a source of data, of unprecedented size and diversity, with possibly Internet scope, that might have a large impact on fault detection.

Many measurement platforms can perform traceroutes, which allow operators to gather information about the path traversed by traffic from any probe towards any host in the Internet. Advantages of performing this kind of monitoring are manifold: 1) traceroutes are end-to-end measurements which are not easy to obtain by other monitoring approaches; 2) traceroutes can provide information about portions of the network outside the direct control of an operator, helping in the assignment of responsibility for network misbehaviors; 3) traceroutes might detect failures that other approaches may overlook (some examples of "silent failures" are provided in [5]); 4) the support needed for traceroutes is quite basic and widespread.

Measurement platforms run continuously and therefore produce a huge amount of data. However, extracting meaningful information from a large number of traceroutes is a challenging task.

In this paper we introduce a novel methodology and an algorithm that enable the analysis of large collections of traceroute measurements in search of significant changes, with the intent of easing management and troubleshooting. Our methodology takes as input a set of traceroutes, identifies paths that evolve similarly over time, and reports them aggregated

M. Di Bartolomeo, V. Di Donato, M. Pizzonia and C. Squarcella are with the Department of Engineering, Roma Tre University, 00146, Rome, Italy (e-mail: dibartolomeo@ing.uniroma3.it, didonato@ing.uniroma3.it; squarcel@ing.uniroma3.it; pizzonia@ing.uniroma3.it).

M. Rimondini is with the Network Design & Engineering group of Unidata S.p.A., 00148, Rome, Italy (e-mail: m.rimondini@unidata.it)

A preliminary version of this paper appeared in [1].

into inferred events (e.g., routing changes, loss of connectivity), augmented with an impact estimate and a restricted set of IP addresses that are likely close to the cause of the event (a piece of information similar to those provided by tomography-based techniques [6]). The methodology, as well as its correctness, is founded on the notion of *empathy*, a relation that binds similarly behaving traceroutes, which are therefore a good evidence of the same network event. Our approach does not need a-priori knowledge of the network topology, does not assume a stable routing state, and does not impose restrictions on the schedule of traceroutes, which may be collected asynchronously and at arbitrary intervals. Instead, it takes advantage of asynchronous measurements to improve the timeliness and precision of event detection, and is resistant to measurement errors (e.g., due to software errors or routing anomalies), which in most cases only generate fictitious events with a small impact that are easy to filter out.

We provide experimental evidence of the effectiveness of our approach by running our algorithm on data collected by large-scale measurement infrastructures, such as RIPE Atlas, and by comparing the inferred events with ground truth derived from induced routing changes or third-party information.

The rest of the paper is organized as follows. In Section II we review related contributions. In Section III we describe our network model and the fundamental properties of the empathy relation. In Section IV we introduce a methodology and an algorithm, based on empathy, to infer events and report relevant data about them. In Section V we report on some applicability considerations. In Section VI we analyze the results of the application of our methodology to real-world data and finally in Section VII we draw conclusions and present ideas for future work.

II. Related Work

An Internet measurement platform is an infrastructure of dedicated probes that periodically perform network measurements such as ping or traceroute on the Internet. A number of such platforms have emerged in the last decade and a comprehensive survey can be found in [2]. Some examples include SamKnows [7], RIPE Atlas [8], PlanetLab [9] and CAIDA Ark [10], and a standardization effort to make these platforms interoperable is also ongoing, as attested by [3], [11], [12]. Data produced by these platforms have been used in the past in a variety of contexts and have been combined with control-plane data by a number of systems mostly aimed at detecting and localizing path changes and various types of anomalies in the network [13]–[16]. In the following we review the most relevant contributions in this ecosystem classifying them based on their objectives, data sources and target networks.

a) Systems to detect and localize Internet interdomain path changes: The algorithm presented in [17] aims at detecting interdomain path changes in the Internet considering RIPE RIS [18] and the Oregon RouteViews project [19] as data sources. The proposed technique is based on a previously published method presented in [20]. Feldmann et al. propose in [21] a novel technique to localize interdomain path changes and therefore identify the autonomous system(s) responsible for the changes and for their propagation. Subsequently Teixeira et al. [22] explain why considering public BGP data only is not sufficient to find the root cause of routing changes. A decade after, Javed et al. introduce PoiRoot [23], a novel system to detect interdomain path changes that combines BGP feeds coming from RouteViews with active traceroute measurements performed via PlanetLab nodes.

b) Systems to detect and localize faults in the Internet: Fault detection and localization at the BGP level has been extensively studied as attested by a recent survey appeared in [24]. In this context, it is often the case that BGP routing data are combined with active measurements in such a way to better quantify and characterize the scope of the inferred anomalies. Hubble [13] for example, is a system that monitors BGP feeds but also looks at ping measurements performed by PlanetLab probes. PlanetSeer [14] and LIFEGUARD [15] are two systems that passively monitor traceroutes to detect anomalous behavior, and then coordinate active traceroutes to confirm and further investigate the anomalies. NetDiagnoser [16] instead, combines traceroutes and BGP feeds, along with information extracted from BGP Looking glasses.

c) Systems to detect and localize faults in known networks: Systems listed in this section still deal with fault detection and localization but are mostly targeted to specific types of networks for which the topology is, at least partially, known in advance. Controlled environments also offer the possibility to install software on the agents and therefore to collect finer-grained information on the connectivity status. SCORE [25] and MaxCoverage [26] are for example two systems aimed at the backbone of an ISP. The former is based on risk models whereas the latter adopts a spatial-correlation approach. Sherlock [27] is suited for large enterprise networks and requires each host to install a software agent in such a way to analyze the incoming and outgoing packets and infer the structure of the network. Gestalt [28] tries to adapts to different types of networks and combines the best elements of existing techniques to obtain a higher localization accuracy and a lower running time with respect to the alternatives. Another set of contributions that notably assume at least a partial knowledge of the network topology, goes under the name of binary tomography. The binary tomography approach, firstly proposed for trees [6] and then extended to general topologies [16], [29], has applicability problems which have been discussed in [30]–[32]. A natural extension to this approach has been successfully implemented in NetDiagnoser [16].

Finally, a relevant contribution that is outside the previous classification appeared in [33]. In this paper, the authors search for patterns in traceroute data collected by RIPE Atlas probes. They do so by developing a measure for the differences between successive traceroutes and then by using this measure to cluster routing changes across all the considered vantage points. The idea of this work is somewhat similar to ours, but the paper focuses more on providing meaningful interpretations of traceroute behavior rather than on easing network management and troubleshooting.

In contrast with current literature we propose a methodology to search for significant path changes in the Internet at the

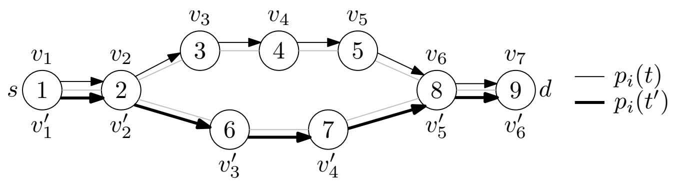


Fig. 1: Example of two traceroute paths from s to d collected at different time instants t and t'. Gray lines are network links.

IP level. Our technique takes as input a set of traceroutes only, does not assume any knowledge on the topology of the network, and does not impose any restrictions on the schedule of the considered traceroutes.

III. The Empathy Relationship

In this section, we describe the model we use to analyze traceroute paths, which is at the basis of our event inference method. We also introduce several assumptions that make our approach easier to understand and allow us to take advantage of several properties. We illustrate in Section VI how, even under these assumptions, interesting results can be obtained with real-world data, and discuss in Section V their impact on the practical applicability of our approach.

Informally speaking, we introduce the concept of *empathic* traceroutes, which are traceroutes that happen to change similarly at roughly the same time, even if measured between unrelated sources and destinations. As explained in the following, when an event occurs (e.g., a router or link fault), a set of empathic traceroute changes can be observed and they are pairwise empathic. We call *impact* the size of this set, since, under certain conditions, detailed in Section V.8, a higher number of changed traceroutes is associated with more relevant events.

Let $G = (V, E)$ be a graph that models an IP network: vertices in V are network devices (routers or end systems), and edges in E are links between devices. Some devices in V, called *network probes* or *sources*, periodically perform traceroutes towards a predefined set of *destinations*. We assume that each traceroute is acyclic (otherwise there is evidence of a network anomaly). We assume each traceroute measurement to be not affected by any ongoing routing change. This is equivalent to consider it instantaneously performed when its execution ends. For this reason, in the following, we always formally associate a single instant of execution for each traceroute.

Let $i = (s, d)$, where $s \in V$ is a source and $d \in V$ is a destination. We call i a *source-destination pair*, or *sd-pair*. A *traceroute path* $p_i(t)$ measured at time t by s towards d is a sequence $\langle v_1\ v_2\ \dots\ v_n \rangle$ such that $v_1 = s$, $v_j \in V$ for $j = 1, \dots, n$, and there is an edge in E for each pair (v_k, v_{k+1}), $k = 1, \dots, n-1$. While we include the source in $p_i(t)$, the destination may not appear because a traceroute may end at an unintended vertex different from d. For convenience, let $V(p)$ be the set of vertices of path p.

Now, consider two traceroute paths $p_i(t) = \langle v_1\ v_2\ \dots\ v_n \rangle$ and $p_i(t') = \langle v'_1\ v'_2\ \dots\ v'_m \rangle$ between the same source-destination pair $i = (s, d)$, with $t' > t$, and assume that

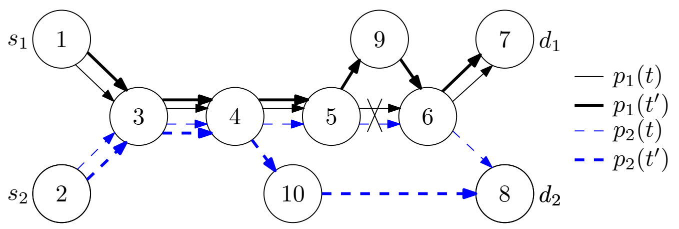


Fig. 2: Example of empathy relations. Link $(5, 6)$ fails, and we have $(s_1, d_1) \overset{\text{pre}}{\sim}_t (s_2, d_2)$ but $(s_1, d_1) \overset{\text{post}}{\not\sim}_t (s_2, d_2)$.

$p_i(t) \neq p_i(t')$. Fig. 1 shows two such paths: $i = (1, 9)$, $p_i(t) = \langle 1\ 2\ 3\ 4\ 5\ 8\ 9 \rangle$, and $p_i(t') = \langle 1\ 2\ 6\ 7\ 8\ 9 \rangle$. Since the path from s to d has changed between t and t', we call the pair consisting of $p_i(t)$ and $p_i(t')$ a *transition*, indicated by τ_i, and say that it is *active* at any time between the *endpoints* t and t', excluding t'. To analyze the path change, we focus on the portion of the two paths that has changed in the transition: let $\delta^{\text{pre}}(\tau_i)$ indicate the shortest subpath of $p_i(t)$ that goes from a vertex u to a vertex v such that all the vertices between s and u and between v and the end of the path are unchanged in $p_i(t')$ and are in the same order. If there is no such v (for example because the destination is unreachable at t or t'), $\delta^{\text{pre}}(\tau_i)$ goes from u to the end of $p_i(t)$. Referring to the example in Fig. 1, it is $\delta^{\text{pre}}(\tau_i) = \langle 2\ 3\ 4\ 5\ 8 \rangle$. We define $\delta^{\text{post}}(\tau_i)$ as an analogous subpath of $p_i(t')$. Referring again to Fig. 1, it is $\delta^{\text{post}}(\tau_i) = \langle 2\ 6\ 7\ 8 \rangle$. In principle, $\delta^{\text{pre}}(\tau_i)$ may have several vertices in common with $\delta^{\text{post}}(\tau_i)$ besides the first and the last one: we still consider $\delta^{\text{pre}}(\tau_i)$ as a single continuous subpath, with negligible impact on the effectiveness of our methodology. The same applies to $\delta^{\text{post}}(\tau_i)$.

We can now introduce the concept of empathy, that determines when two traceroute paths exhibit a similar behavior over time. Consider two transitions τ_1, with source s_1 and destination d_1, and τ_2, with source s_2 and destination d_2, such that both transitions are active between t and t', $t' > t$, at least one has an endpoint in t, and at least one has an endpoint in t'. We say that (s_1, d_1) *and* (s_2, d_2) *are pre-empathic at any time* $t \leq \hat{t} < t'$ if the portions of $p_1(t)$ and $p_2(t)$ that change during τ_1 and τ_2 overlap, namely $V(\delta^{\text{pre}}(\tau_1)) \cap V(\delta^{\text{pre}}(\tau_2)) \neq \emptyset$. Intuitively, traceroute paths relative to pre-empathic sd-pairs stop traversing a network portion that they shared before an event occurred. Similarly, we say that (s_1, d_1) *and* (s_2, d_2) *are post-empathic at any time* $t \leq \hat{t} < t'$ if $V(\delta^{\text{post}}(\tau_1)) \cap V(\delta^{\text{post}}(\tau_2)) \neq \emptyset$. Post-empathy captures a different kind of path change: traceroute paths of post-empathic sd-pairs start traversing a common portion that they did not use before the event occurred. Fig. 2 shows two traceroute paths p_1, from s_1 to d_1, and p_2, from s_2 to d_2, that change between t and t' due to the failure of link $(5, 6)$. Considering the corresponding transitions τ_1 and τ_2, we have $\delta^{\text{pre}}(\tau_1) = \langle 5\ 6 \rangle$, $\delta^{\text{post}}(\tau_1) = \langle 5\ 9\ 6 \rangle$, $\delta^{\text{pre}}(\tau_2) = \langle 4\ 5\ 6\ 8 \rangle$, and $\delta^{\text{post}}(\tau_2) = \langle 4\ 10\ 8 \rangle$. Since $\delta^{\text{pre}}(\tau_1)$ and $\delta^{\text{pre}}(\tau_2)$ share vertices 5 and 6, (s_1, d_1) and (s_2, d_2) are pre-empathic between t and t', whereas (s_1, d_1) and (s_2, d_2) are not post-empathic despite the fact that $p_1(t')$ and $p_2(t')$ share a subpath. Indeed, p_1 and p_2 behave similarly before the link fails and change to two independent routes afterwards.

In order to understand how empathy is useful to infer network events, we need to formally introduce the notion of event, qualifying it as physical to distinguish it from events inferred by our algorithm. We call *physical event* at time $\bar{t}$ the simultaneous disappearance of a set $E^{\downarrow}$ of links from E (*down event*) or the simultaneous appearance of a set $E^{\uparrow}$ of links in E (*up event*), such that:

- either $E^{\downarrow} = \emptyset$ or $E^{\uparrow} = \emptyset$ (a physical event is either the disappearance or the appearance of links, not both);
- $E^{\downarrow} \subseteq E$ (only existing links can disappear);
- $E^{\uparrow} \cap E = \emptyset$ (only new links can appear);
- $\exists v \in V \mid \forall (u, w) \in E^{\downarrow} : u = v$ or $w = v$, and the same holds for $E^{\uparrow}$ (all disappeared/appeared edges have one endpoint vertex in common). Vertex v is called *hub* of the event (an event involving a single edge (u, v) has two hubs: u and v; any other event has a unique hub).

When the type of an event is not relevant, we indicate it as $E^{\downarrow\uparrow}$. This event model captures the circumstance in which one or more links attached to a network device fail or are brought up, including the case in which a whole device fails or is activated. Such events may be caused, for example, by failures of network interface cards, line cards, or routers, by accidental link cuts, by provisioning processes, and by administrative reconfigurations, if they fit this model. Congestion may be detected as an event if it makes a balancer shift traffic away from a set of links. Failures or activations of links that do not have a vertex in common are considered distinct events. We only consider *visible* physical events, namely those that cause at least a transition to occur. Moreover, we assume that every transition comprises at least one edge involved in a physical event, an assumption that is long-argued in the literature about root-cause analysis (see, e.g., [23]) and yet we deem reasonable because our goal is to detect events, not reconstruct their original cause. Given a physical event $E^{\downarrow\uparrow}$ occurred at time $\bar{t}$, we define the *scope* $S(E^{\downarrow\uparrow})$ of $E^{\downarrow\uparrow}$ as the set of sd-pairs $i = (s, d)$ involved in the transitions that are active at $\bar{t}$. The sd-pairs of the scope are pairwise empathic. We also call *impact* of $E^{\downarrow\uparrow}$ the cardinality of $S(E^{\downarrow\uparrow})$.

In our model, there is a strong relationship between the occurrence of a physical event and the existence of a set of sd-pairs that, when that physical event occurs, are all pairwise empathic. We exploit this relationship in our inference algorithm shown in Section IV and we formally state and prove it in Theorems 1 and 2. Since not all the assumptions of our model hold in real networks, the effectiveness of our algorithm, when run on real data, depends on how small are the errors introduced by deviations of real networks from our model. In Section V, we discuss some of these deviations. The experimental evaluation of Section VI is intended to provide evidences that, in practice, errors are small.

IV. Seeking Events: Methodology and Algorithm

In this section, we describe our inference algorithm for detecting routing events. The algorithm takes as input a set of traceroute paths, and produces as result a list of inferred events, each described by a tuple $(t_1, t_2, S, \Pi, \mathit{type})$, where,

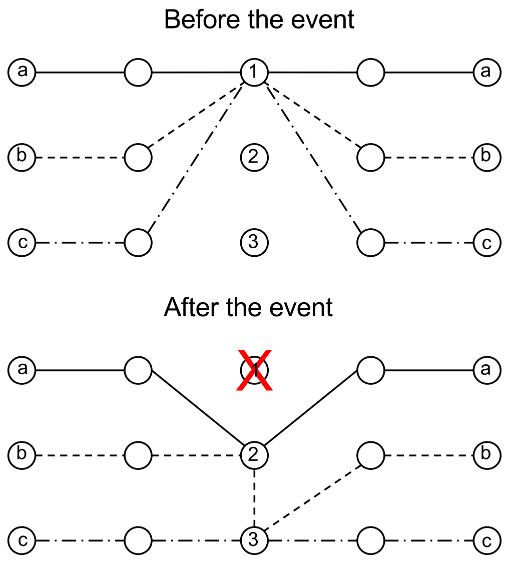


Fig. 3: An example of down event where the hub is the node 1. The event affects three sd-pairs.

i) $[t_1, t_2]$ is an interval of time in which the event is supposed to have occurred, ii) S is a set of sd-pairs affected by the event (the scope), iii) Π is a set of IP addresses that, after the event has occurred, (dis)appeared in all the traceroutes performed between sd-pairs in its scope and that contains the hub of a physical event, and iv) $\mathit{type} \in \{\mathsf{up}, \mathsf{down}, \mathsf{unknown}\}$ is the type of the event.

We refer to the model illustrated in the previous section, considering the general case of non-synchronized traceroute measurements. That is, for an sd-pair $i = (s, d)$ traceroute paths $p_i(t)$ are only available at specific time instants $t \in \mathbb{R}$ that depend on s (if probes are synchronized via NTP, whose precision is high enough for our needs, we can refer to a universal clock). As we will show later, unsynchronized traceroutes can improve the accuracy of the interval reported by our algorithm for an inferred event. For convenience, for a transition τ_i we define the *changed set* $\Delta(\tau_i)$ consisting of *extended addresses*, namely IP addresses in $V(\delta^{\mathrm{pre}}(\tau_i))$ labeled with a tag pre and IP addresses in $V(\delta^{\mathrm{post}}(\tau_i))$ labeled with a tag post.

Our algorithm consists of three phases.

Phase 1 – Identification of transitions: in this phase, for each sd-pair i, input samples $p_i(t)$ are scanned and all transitions τ_i, with the corresponding $\Delta(\tau_i)$, are identified. As an example, consider the network event shown in Figure 3 where three sd-pairs a, b, and c are associated with three transitions, which are the consequence of a physical down event with hub 1, denoted τ_a, τ_b, and τ_c, respectively. The upper part of Figure 4 shows the three transitions represented as segments terminating at the transitions' endpoints, and the corresponding changed sets (IP addresses are represented as numbers).

Phase 2 – Construction of candidate events: in this phase, the algorithm tracks the evolution of empathy relationships between sd-pairs involved in transitions. As detailed in Fig. 5, the algorithm linearly sweeps on time instants corresponding

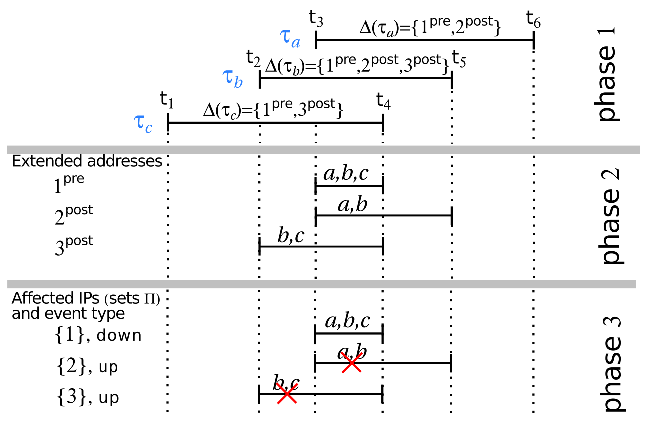


Fig. 4: Sample outputs of the various phases of our algorithm run on the measurements related to event shown in Figure 3.

```
Input: a set $T$ of transitions
Output: a set $CEvents$ of candidate events, namely tuples $(t_1, t_2, S, \mathcal{A})$
    indicating time intervals $[t_1, t_2]$ in which all the sd-pairs in $S$ are pre-
    empathic or post-empathic with each other and all the corresponding
    transitions $\tau_i$ have extended address $\mathcal{A}$ in their changed set $\Delta(\tau_i)$.
 1: Let $t_\mathcal{A}$ be a mapping from each extended address $\mathcal{A}$ to a timestamp,
    initially set to $-\infty$.
 2: Let $S_\mathcal{A}$ be a mapping from each extended address $\mathcal{A}$ to a set of sd-pairs,
    initially empty.
    ▷ For each $\mathcal{A}$, variables $t_\mathcal{A}$ and $S_\mathcal{A}$ correspond to queues of length three.
      Each assignment to $t_\mathcal{A}$ or $S_\mathcal{A}$ enqueue a new value discarding the
      oldest one. We denote the two old values in a queue by $prev(\cdot)$ and
      $pprev(\cdot)$, with $pprev(\cdot)$ being the oldest (if unset, these functions
      return $\emptyset$ for $S_\mathcal{A}$ and $-\infty$ for $t_\mathcal{A}$).
    ▷ The algorithm sweeps through transitions endpoints keeping the
      following invariants: (1) each $S_\mathcal{A}$ is the union of sd-pairs of transitions
      $\tau_i$ active right after the current endpoint $t$, such that $\mathcal{A} \in \Delta(\tau_i)$ and
      (2) each $t_\mathcal{A}$ is the last timestamp when $S_\mathcal{A}$ changed.
 3: $CEvents = \emptyset$
 4: for $t$ endpoint of transitions in $T$, in time order do
 5:     for $\tau$ transition in $T$ ending at $t$ do
 6:         for $\mathcal{A}$ extended address in $\Delta(\tau)$ do
 7:             Remove the sd-pair of $\tau$ from $S_\mathcal{A}$
 8:             $t_\mathcal{A} = t$
 9:             if $|pprev(S_\mathcal{A})| \leq |prev(S_\mathcal{A})|$ then
10:                 Add $(prev(t_\mathcal{A}), t_\mathcal{A}, prev(S_\mathcal{A}), \mathcal{A})$ to $CEvents$
11:             end if
12:         end for
13:     end for
14:     for $\tau$ transition in $T$ starting at $t$ do
15:         for $\mathcal{A}$ extended address in $\Delta(\tau)$ do
16:             Add the sd-pair of $\tau$ to $S_\mathcal{A}$
17:             $t_\mathcal{A} = t$
18:         end for
19:     end for
20: end for
21: return $CEvents$
```

Fig. 5: Inference algorithm, Phase 2.

to transition endpoints and, for every instant t (line 4) and every transition that ends or starts at t (lines 5 and 14), for each extended address $\mathcal{A}$ (lines 6 and 15), updates a set $S_\mathcal{A}$ of sd-pairs i corresponding to active transitions that are empathic with each other because they have $\mathcal{A}$ in their changed set $\Delta(\tau_i)$ (lines 7 and 16). Sets $S_\mathcal{A}$, as well as the time instants $t_\mathcal{A}$ at which they are updated, are kept in special variables which allow access to the last 3 assigned values. Access to the values assigned before the current one is denoted by $prev(\cdot)$ and $pprev(\cdot)$. When the size of each $S_\mathcal{A}$ reaches a local maximum at time t (line 9), the algorithm reports a candidate event. This corresponds to seeking for the time instant at which the highest number of sd-pairs have seen IP address $\mathcal{A}$ (dis)appear in their traceroute paths. The interval $[prev(t_\mathcal{A}), t_\mathcal{A}]$ of validity of the local maximum (line 10) turns out to be a narrow time window within which a physical event occurred (see Theorem 1). The middle part of Fig. 4 shows a sample output of this phase, where each segment represents a candidate event: for each extended address appearing in the changed sets of τ_a, τ_b, and τ_c, the corresponding sets of sd-pairs $S_{1^{\text{pre}}}$, $S_{2^{\text{post}}}$, and $S_{3^{\text{post}}}$ that involved that address are constructed and updated. In particular, set $S_{1^{\text{pre}}}$ reaches its maximum size at time t_3, when extended address 1^{pre} is in the changed set of τ_a, τ_b, and τ_c, namely IP address 1 has disappeared for sd-pairs a, b, and c. The reported candidate event ends at t_4, when the size of $S_{1^{\text{pre}}}$ is again reduced: it is therefore $(t_3, t_4, \{a, b, c\}, 1^{\text{pre}})$. Similar considerations apply for the construction of the other two candidate events $(t_3, t_5, \{a, b\}, 2^{\text{post}})$ and $(t_2, t_4, \{b, c\}, 3^{\text{post}})$.

Phase 3 – Event inference: in this phase, detailed in Fig. 6, candidate events are sieved to build a set of inferred events, each consisting of a time window, a scope, a set of involved IP addresses (which contains the hub of the corresponding physical event), and a type (up/down/unknown). As a first clean-up step, all candidate events whose set of sd-pairs is properly contained in the set of sd-pairs of another candidate event that overlaps in time are discarded (lines 2-6). In this way, only events with maximal impact are reported. Afterwards, the algorithm considers groups $CEvents(S, t_1, t_2)$ of candidate events spanning the same time interval $[t_1, t_2]$ and having S as set of sd-pairs (line 7), and constructs an inferred event for every set S. The inferred event has the following structure (lines 9-13): the time interval is $[t_1, t_2]$; the scope is S; the involved IP addresses are the union of the addresses of candidate events in $CEvents(S, t_1, t_2)$; and the type is inferred based on the labels of the extended addresses of candidate events in $CEvents(S, t_1, t_2)$ (lines 11-13). A sample result of the application of this phase is in the lower part of Fig. 4: candidate events $(t_3, t_5, \{a, b\}, 2^{\text{post}})$ and $(t_2, t_4, \{b, c\}, 3^{\text{post}})$ (segments in the second and third row of phase 2, respectively) are discarded because their sets of sd-pairs are contained in the one of the overlapping candidate event $(t_3, t_4, \{a, b, c\}, 1^{\text{pre}})$. At this point, there is only one set of sd-pairs left, $\{a, b, c\}$: the only candidate event having such set is reported as an event, which affected IP address 1 (that is also the hub of the event) and whose type is down because of the label of 1^{pre}.

Our algorithm is correct and complete, as stated by the following theorems that link the inference results with physical events (defined in Section III). The theorems are proven under the hypothesis (*non-concurrency*) that each transition is induced by one physical event and that distinct transitions induced by distinct physical events do not overlap in time (this hypothesis is discussed in Section V.4). We also recall that in our model we assume that each transition always involves at least one edge of the physical event that caused it (see Section III).

Theorem 1 (Correctness): Each event inferred by the inference algorithm corresponds to a physical event whose hub is in the set of involved addresses reported in the inferred event

```
Input: a set $CEvents$ of candidate events produced in phase 2 (see Fig. 5)
Output: a set $Events$ of tuples $(t_1, t_2, S, \Pi, type)$, each representing an
    inferred event occurred between $t_1$ and $t_2$, whose scope is $S$, which
    involved the IP addresses in $\Pi$, and whose type is $type$.
 1: $Events$ = $\emptyset$
 2: for every pair $e = (t_1, t_2, S, \mathcal{A})$, $\tilde{e} = (\tilde{t}_1, \tilde{t}_2, \tilde{S}, \tilde{\mathcal{A}})$ in $CEvents$ do
 3:     if $e$ and $\tilde{e}$ overlap in time and $\tilde{S} \subset S$ then
 4:         remove $\tilde{e}$ from $CEvents$
 5:     end if
 6: end for
 7: Group candidate events $(t_1, t_2, S, \mathcal{A})$ in $CEvents$ by $S$, $t_1$, and $t_2$.
 8: for every computed group $CEvents(S, t_1, t_2)$ do
 9:     $eaddr$ = $\bigcup_{(t_1,t_2,S,\mathcal{A}) \in CEvents(S)} \mathcal{A}$
10:     $\Pi$ = $eaddr$ (without labels)
11:     $type$ = unknown
12:     $type$ = down if all addresses in $eaddr$ are tagged as pre
13:     $type$ = up if all addresses in $eaddr$ are tagged as post
14:     add $(t_1, t_2, S, \Pi, type)$ to $Events$
15: end for
16: return $Events$
```

Fig. 6: Inference algorithm, Phase 3.

and the instant when the physical event occurred is within the reported interval.

Proof: Let $(t_1, t_2, S, \Pi, type)$ be an inferred event. Consider π in Π. By construction, π has (dis)appeared during $[t_1, t_2]$ in all transitions τ_i for every $i \in S$. Since, all τ_i overlap in time they necessarily are caused by the same physical event (non-concurrency). This physical event must have occurred in the active interval of each transition, by definition of transition. The intersection of the active intervals of all transitions is the reported interval $[t_1, t_2]$ by construction. Since, each transition τ_i always involves at least one edge of the physical event that caused it, $\Delta(\tau_i)$ contains the hub h of the physical event. Since $\Pi = \cap_i \Delta(\tau_i)$, then $h \in \Pi$. ∎

Theorem 2 (Completeness): For every visible physical event, an inferred event is reported by the algorithm whose set of involved addresses contains the hub of the physical event and the instant when the physical event occurred is within the reported interval.

Proof: Suppose a physical event $E^{\downarrow}$ with hub h occurs at time $\bar{t}$. Since the physical event is visible, there exists at least one sd-pair that changed its traceroute after $\bar{t}$. The scope $S(E^{\downarrow})$ is the set of such sd-pairs. For each $i \in S(E^{\downarrow})$, Phase 1 of the algorithm constructs transitions τ_i whose intervals contain $\bar{t}$. All τ_i intersect at a common interval $[t_1, t_2]$ comprising $\bar{t}$ and have $h^{\text{pre}} \in \Delta(\tau_i)$. The set $S_{h^{\text{pre}}}$ becomes equal to $S(E^{\downarrow})$ between t_1 and t_2 in phase 2 of the algorithm. Since the physical event is unique (non-concurrency) and we cannot have transitions without a physical event, the candidate event $e = (t_1, t_2, S(E^{\downarrow}), h^{\text{pre}})$ has the largest possible set of sd-pairs. Therefore e is not filtered out by Phase 3 and an event $(t_1, t_2, S(E^{\downarrow}), \Pi, type)$ with $h \in \Pi$ is reported. Analogous arguments can be applied to the cases of $E^{\uparrow}$ and $E^{\uparrow\downarrow}$. ∎

The computational complexity of our inference algorithm is $O(|T| + |CEevents|^2 \cdot I)$, where T is the set of transitions and I is the maximum impact. In fact, Phase 1 takes $O(|T|)$. Phase 2 also takes $O(|T|)$, since each transition is processed only twice and the size of the changed set of every transition is bounded by the maximum length of the traceroute paths (the fraction of traceroutes with more than 30 hops is negligible [34]).

Phase 3 takes $O(|CEvents|^2 \cdot I)$ because of the overlap check at lines 2-6 (the following event construction can be performed efficiently by scanning candidate events).

V. Applicability Considerations

In this section, we discuss several aspects concerning the application of our approach to real-world traceroute data.

1) Time synchronization: In our description, we assume that the internal clocks of each probe is properly synchronized. This is usually performed by NTP, whose precision is in the order of a few milliseconds [35], while traceroutes are performed on a timescale of minutes (or seconds when performed at high rates). In our opinion, this is not a relevant issue.

2) Probing frequency: The probing frequency should be properly dimensioned. It affects the extent T of the time interval of the inferred events and in certain cases the ability to detect physical events. The extent T can be regarded as the tolerance with which the algorithm positions events in time. Note that, T depends not only on the probing frequency, but also on the number of transitions induced by that event (i.e. its impact), which in turn depends on the topological position of the physical event with respect to sources and destinations of measurements. If probes perform measurements at frequency f with random independent phase, the expected value of T is $1/(f \cdot |S|)$, where $|S|$ is the impact of the event, since frequencies sum up. Regarding the ability to detect physical events, consider a pair of consecutive opposite physical events (e.g., shutdown and boot of a router) e_1 and e_2 occurring at times t_1 and t_2 respectively, with $\Delta t = t2 - t1$. The network state before t_1 and after t_2 is the same. Let us suppose that the positions of sources and destinations are so that the set $\overline{S}$ of sd-pairs that can see e_1 and e_2 is non-empty. We denote by f the probing frequency. We have two transitions for each sd-pair in $\overline{S}$ if $\Delta t > 1/f$, otherwise there are chances to “miss” the event depending on the probing phase. Namely, each sd-pair detects the pair of events with probability $f \Delta t$. In this case, if measurements in $\overline{S}$ have random independent phase, the expected impact of the inferred event is $|\overline{S}| f \Delta t$.

3) Aliasing: A single network device equipped with multiple network interfaces (e.g., a router) may reply with different IP addresses to different traceroutes, a phenomenon known as *aliasing* [36]. In this case, distinct sd-pairs may recognize one router as several ones. As a consequence, detection of some empathies may fail, causing our algorithm to infer multiple small events rather than a single larger one, in the worst case. For example, consider the case in which our algorithm, in the absence of aliasing, would detect a routing event with exactly one involved router r. Now suppose that r is affected by aliasing and replies with two distinct IP addresses, r_A and r_B, to distinct set of sd-pairs A and B. Our algorithm would infer two routing events pinpointing both r_A and r_B impacting A and B, respectively. This might cause one or both the inferred events to have impact below the threshold introduced in Section V.9 and hence pass undetected. Aliasing that affects routers not involved in the event do not affect the inference capability of our approach, even if those routers appear in changed sets of some transitions. Further, the adoption of databases created by means of IP-alias resolution techniques may be considered (see for example [37]).

4) Simultaneous (concurrent) events: Consider two physical events e_1 and e_2. Each of these is associated with a set of transitions. Due to the fact that traceroute sampling period is of the order of minutes, even non-strictly simultaneous events can result in possibly a part of their transitions to overlap in time. This occurs independently from the (un)synchronization of sampling across different sd-pairs. Suppose that e_1 and e_2 are close enough, both in topology and in time. In this case, "secondary" extended addresses that are present in changed sets of transitions for both events might get a cumulative effect and turn out to be erroneously recognized as associated with a non-existent third event.

The probability of observing such a situation in real data is hard to estimate, since it depends on the topology of the network, the positions of sources and destinations of traceroutes, and the probability distribution of the physical events in the network and over time. While the topic deserves a deeper analysis, we can make some simple preliminary considerations on a couple of extreme cases. If physical events are independent and their frequency is much lower than the measurement frequency, it is unlikely to have time-overlapping transitions for distinct events. In cases where two physical events are induced by a single root-cause that is physically/logically close to the involved appliances, there is a non-negligible probability to pinpoint wrong addresses that however tend to be close to the appliances that caused the route change.

5) Non-instantaneous routing dissemination: It is well known that certain routing events take some time to propagate. Link state routing protocols, like OSPF, are quite fast, with propagation delays well below the sampling granularity we are considering. On the other hand, BGP advertisements might be delayed by the MRAI timer, which is set to 30 seconds according to the BGP standard. Hence, when BGP is involved, at a certain time some routers may see (and propagate) a stale version of the routing. In this case, it might occur that not all transitions that are caused by a certain routing event overlap on a common interval. In this situation, our algorithm infers multiple events, usually with largely overlapping scopes. This phenomenon is particularly evident in our induced event experiment (see Section VI.1), where we manually analyzed the recorded traceroutes to understand the source of the multiplicity of dots for each event that can be seen in Figure 7.

We note that this problem could be mitigated by a higher probing frequency, but this approach reduces the time-accuracy of the inference for all events. In practice, a possible approach is to merge events with same involved addresses that are close in time and have similar scopes (see also Section VI.4).

6) Load balancers: One aspect that may indeed taint the output of our algorithm is that the vast majority of Internet paths traverse load balancers [38]. They are the cause of a high number of apparent routing changes which may be improperly reported as physical events. Compensating this issue requires knowledge of the load balancers, which is realistic for an Internet Service Provider that wants to apply our methodology, and can otherwise be constructed by applying discovery techniques such as Paris Traceroute [39]. Unfortunately, this technique was not yet available in the measurement platforms we considered, at the time of the experiments described in Section VI. For our experiments, we preprocessed traceroutes by using a simple heuristic that cleaned up most of the noise introduced by load balancers: we analyzed all the input traceroutes in their time order and, for each destination, we tracked the evolution over time of the routing (actually their next hop for that destination) of every node along the traceroute paths. Nodes with unstable routing, i.e., that change routing over $X\%$ of the samples, are considered to be load balancers and their next hops are replaced by a single arbitrarily chosen representative IP address. Choosing a threshold X that catches most of the load balancers, while keeping real physical events intact, is a hard task. For this reason, this technique should be regarded as an ad-hoc one, and techniques like those described in [39] should be preferred when available. For the experiments discussed in Section VI, we empirically chose to consider load balancers nodes that change their next hop in more than 20% of the samples, which reduces the number of transitions by 57%, 78%, and 66% in Experiments 1, 2 and 3 respectively.

7) Traceroutes with cycles and private addresses: Real traceroutes may show "anomalies" like cycles or private addresses. While it is possible to provide correct configurations that show these behavior, in our experiments, we observed only a very limited number of them and we completely discarded these kind of traceroutes. However, other approaches might be adopted, for example, preserving the part of a traceroute that does not contain such anomalies but can still provide valuable information about routing changes.

8) Probe positions and event impact: The capability of inferring a certain routing event from traceroutes depends on the coverage of the part of the network where the event occurred, by traceroutes given as input to our algorithm. This strongly depends on the choice of the positions of the probes and of traceroute targets. The problem of minimizing the number of probes while keeping a given event detection capability was studied in [40], [41]. The problem was proven to be NP-hard and heuristic approaches were proposed. Assuming that the main use of our methodology is within an ISP, to obtain a positioning that provides meaningful results in terms of detected events and reported impact value, probes should be positioned to mimic user positioning and they have to target commonly accessed services. In this way, the more critical parts of the network (e.g., the backbone) tend to be traversed by a much larger number of traceroutes with respect to marginal regions and weighted according to their relevance for typical use. This implies that the impact of the inferred events is roughly proportional to the quantity of relevant routing paths (that connect users with services) that the real routing event has changed. The more probe-positioning and traceroute-targeting represent users and their way of using the network, the more the impact of the inferred events will be significant. Note that this approach is independent from the real network topology and the current routing, which can change or be unknown. A more formal analysis of the relation between probe positioning and validity of impacts of inferred events is left as future work.

9) Impact threshold (noise filtering): As we will see in Section VI, our algorithm usually detects a large number of events. The vast majority of them have very low impact and are not interesting, while some are clearly outstanding in term of impact and might deserve the attention of an operator. Our guess is that noise occurs in the periphery of the network, which is less covered by the traceroutes. This interpretation was confirmed by looking at samples we get from Experiment 2 of Section VI: the changes in low impact events turned out to be almost always close to the beginning of the traceroutes, that is, close to the probes, which we know are located in the periphery of the network, for that experiment. One might regard this large number of low-impact events as physiological and hence as *noise* to be ignored. We now describe a possible approach to fix a threshold to help an operator to focus only on significant inferences, which is also the one adopted in Section VI. We assume noise events and interesting events coming from two distinct random processes. Namely, (i) the noise process has high average occurrence frequency and the distribution of its impact I is greater than zero for $0 < I \leq I_{\max}$ and zero elsewhere, and (ii) the interesting events process has very low average occurrence frequency but impact higher than $I_{\max}$. Ideally, we would like to fix the threshold right above $I_{\max}$. In practice, to fix the threshold, we analyze the density distribution of impact values. In a stationary network, the density related with the noise should be stable over time, hence the following procedure can be performed once or with large periodicity, e.g., once a week. First, we compute the density of impact values for a set of inferred events of interest. Then we analyze this density ascending, starting from impact 1. We look for the first value for which the impact density reaches zero. We set the threshold to that value of impact. The intuition that motivates this approach is that during a period of time in which only noise is present, the density estimates the distribution of the values of the impact of noise-related physical events. Hence, we set the threshold where the probability of having a certain value of impact suddenly drop. In production, this approach may lead to false positives due to statistical fluctuations not occurred in the data used for deciding the threshold. Our approach can be improved for example by multiplying the threshold for a constant factor slightly larger than one. Also, in a setting in which the threshold is periodically recomputed, the applied threshold might be the result of a low pass filter of the sequence of the thresholds computed with our simple approach, to smooth statistical fluctuations. In principle, a theoretical model predicting the distribution of impact of noise-related events may help the operator to fix this threshold in a more informed way. However, this model should take into account many aspects, comprising network topology, sources and destinations of the measurements, reboots, faults, congestion, etc. The development of such a model is left as a future work.

VI. Experimental Results

To evaluate the applicability of our methodology, we executed our algorithm on several sets of traceroute paths collected by currently active measurement platforms with the intent to assess the effectiveness of our approach. We discuss it in Section VI.5. All events we considered occurred in the Internet. We first considered a sequence of routing changes that was purposely injected in the network with a known schedule (using them as ground truth). We then used our algorithm to detect spontaneous events. Among the many we detected, we provide a detailed analysis of two of them for which we had a solid ground truth. One is related to a big European operator and one to an important European Internet eXchange Point (IXP). Measurements used in all the following experiments are produced by probes that were independently deployed. Configuration of the measurements (including probing frequency) was not tailored for the use of our methodology or to discover these specific events. For Experiments 2 and 3, the configuration was not performed by us. For Experiment 1, measurements were programmed by us in the past, before the work of this paper started, for a completely different purpose without considering the methodology under evaluation. The software we used is avaliable on the web along with scripts to download and start computations for Experiments 1 and 3[1].

1) First Experiment (Induced Events): For this experiment, we re-used measurements performed in the context of an investigation about the efficacy of IXPs [42]. In that context, we partnered with an Italian ISP that has BGP peerings with three main upstream providers and with a number of ASes at three IXPs, i.e. MIX, NaMeX (the main IXPs in Italy), and AMS-IX. The three IXPs had no connections relevant for our experiment[2]. An IP subnet reserved for the experiment was announced via BGP to different subsets of peers, according to the schedule in Table I. During the experiment, 89 RIPE Atlas probes located in Italy were instructed to perform traceroutes every 10 minutes (between 2014-05-02 13:00 UTC and 2014-05-03 15:00 UTC) targeting a host inside the reserved subnet. After applying the load balancers cleanup heuristic described in Section IV, we fed our algorithm with the collected traceroutes.

The produced output, which took only a few seconds to compute, is plotted in Figure 7: each inferred event $(t_1, t_2, S, \Pi, type)$ is represented by a point whose coordinates are the center of interval $[t_1, t_2]$ (X axis) and the event's impact $|S|$ (Y axis), and whose color identifies a specific set Π of involved IP addresses. We computed an impact threshold according to the procedure described in Section V. The histogram analyzed by the procedure is shown in Figure 8 and the threshold turns out to be 11. Out of all the inferred events, 23 exceeded this impact threshold, which is also highlighted with a dashed horizontal line in Figure 7. The threshold clearly separates outstanding events from noise. It is evident that these 23 events tend to concentrate (red boxes) around the time instants of BGP announcements (vertical gray

[1]https://github.com/empadig/empadig

[2]NaMeX and AMS-IX were connected by a link. However, that link was not used by any means in our specific setting.

TABLE I: Schedule of BGP announcements for the controlled experiment in Section VI.

	Time	Upstreams	MIX	NaMeX	AMS-IX
	May 02, before 14:22	✓	✓	✓	✓
#1	May 02, 14:22	✓			
#2	May 02, 18:22		✓	✓	
#3	May 02, 22:22			✓	
#4	May 03, 02:22		✓		
#5	May 03, 06:22				✓
#6	May 03, 10:22	✓	✓	✓	✓

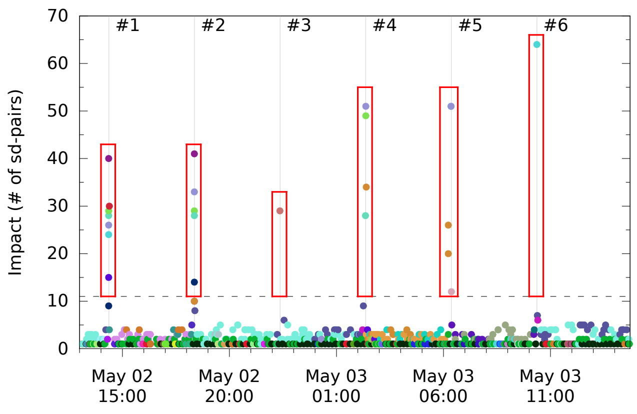


Fig. 7: Impacts of the events inferred for the first experiment (induced events).

lines numbered according to the rows of Table I), and indeed the center of the time interval of each event falls within seconds from the corresponding announcement. In addition, the maximum extension of each interval $[t_1, t_2]$ was 2 minutes, confirming that our methodology can detect an event very quickly after the instant in which it actually happened. Set Π consisted of a single IP address for 87% of the events and of at most 4 IP addresses for 2 events, demonstrating a high precision in pointing out possible event causes. In all cases, reported addresses were clearly associated with our partner ISP, its upstream providers or its IXPs (we performed this check using registries). Further, their positions in traceroutes were consistent with the fact that a change occurred at the border of the ISP, that is where BGP reconfiguration was performed.

For at least one announcement change (#3) the detection was optimal, namely we inferred a single event where all the 29 involved sd-pairs switched from MIX to NaMeX. Multiple events were instead inferred in other cases, probably due to interference between routing changes occurring close in time to each other or to non-instantaneous routing propagation.

Note that, announcement change #6 is a very extreme case. In our model, it corresponds to three concurrent physical up events in which routes all cease to pass through the same router located at AMS-IX. Our algorithm wrongly infers only one down event with the address of that router.

2) Second Experiment (Spontaneous Event at a large ISP): For the second experiment, we considered traceroute paths collected approximately every 8 hours by 3320 probes distributed within the network of a European operator, called

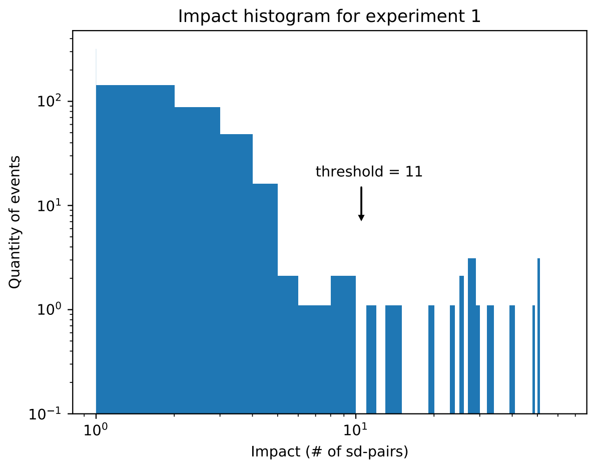


Fig. 8: Histogram of the impacts for the first experiment in doubly logarithmic scale. The value of the threshold, computed as described in Section V.9, is highlighted by an arrow.

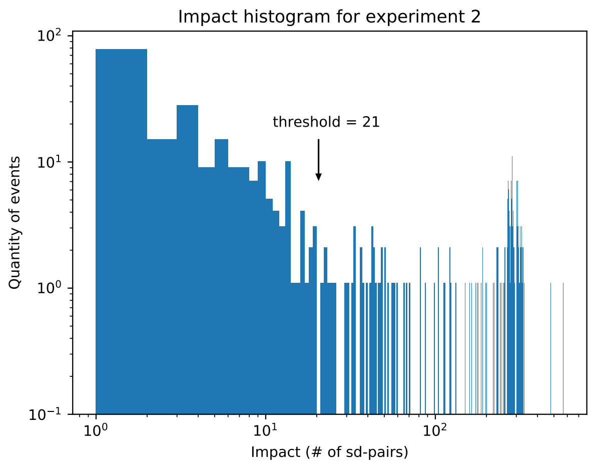


Fig. 9: Histogram of the impacts for the second experiment in doubly logarithmic scale. The computed threshold is highlighted by an arrow.

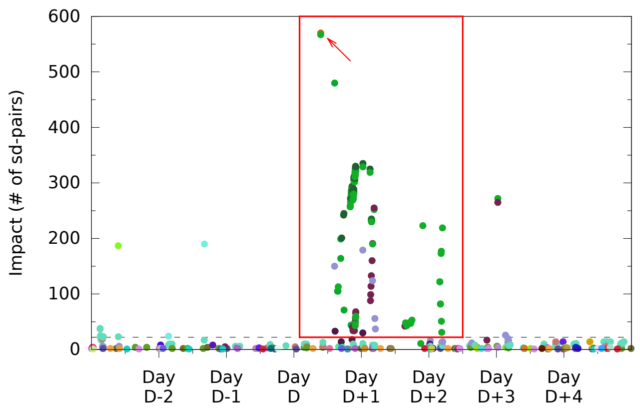


Fig. 10: Impacts of the events inferred for the second experiment (spontaneous events).

EOp in the following for privacy reasons. Traceroutes were performed towards destinations located both inside and outside EOp's network. In a private communication EOp informed us about a "routing failure" in one of its ASes occurred on day D, therefore we focused on traceroutes collected in a 9-days time window comprising this day. We applied our load balancers heuristic on a slightly richer data set consisting of almost 260.000 traceroutes collected over 15 days. Our algorithm then took about 3 minutes to completely process the cleaned set of traceroutes, computing almost 60.000 transitions in phase 1. We separately ascertained that the load balancers heuristic reduced the number of transitions by about a factor of 3 and the number of inferred events by about a factor of 20.

Any operator is primarily interested in the events occurring in its network. Hence, we filtered out inferred events that did not involve (in Π) any IP addresses within EOp's network. The result is shown in Figure 10.

We computed an impact threshold adopting the procedure described in Section V. The histogram analyzed by the procedure is shown in Figure 9 and the threshold turns out to be 22. Figure 10 shows that events exceeding this threshold are mainly concentrated within a time window whose center falls within 24 hours from Day D, and are followed by some less impactful events occurring up to 2 days later (red box in the figure), totaling 199 events involving 838 unique sd-pairs. Our algorithm also singled out their candidate causes pretty accurately, given that the union of all sets Π consisted of only 7 IP addresses. Considering the frequency of traceroutes, these events were also quite precisely located in time: the length of their time intervals ranged from about 10 hours to as low as 1 second (due to traceroutes not being synchronized), with a standard deviation of 30 minutes.

As it can be seen from the figure, reported events are rather fragmented despite affecting the same set of IP addresses (points with the same colors in the figure): this is due to the fact that routing propagation delays caused many non-overlapping transitions to be constructed in phase 1. Interestingly, the two events with impact higher than 550 (indicated by an arrow in the figure) were of type down and up, indicating that all the traceroute paths of the involved sd-pairs switched to alternate routes sharing some common IP addresses (all within EOp's network). Events whose time window is centered between $D+1$ and $D+2$ are likely due to configuration changes undertaken to restore a working routing.

EOp provided us with information about the evolution of an "average download speed" metric over time in the form of a graphical chart, with one sample per our. Even if they do not specify how they measured them, they declared those measurements were taken during the incident and subsequent recovery actions. We manually scaled and aligned the chart of our inferred events against the chart provided by EOp and visually observed that all major down events we detected were close, with tolerance of ± 1 hour, to visible speed drops and the up events to the restoration of the normal speed. We also asked operators of EOp to say if the set of 7 ip addresses that our algorithm inferred as involved in the event were correct. They replied they were.

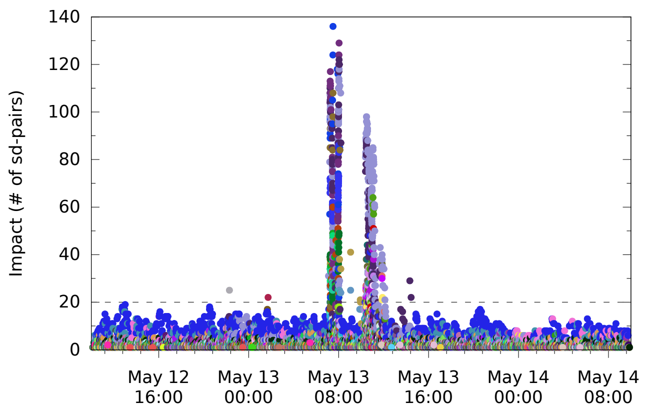


Fig. 11: Impacts of the events inferred for the third experiment (spontaneous events).

3) Third Experiment (Spontaneous Event at an IXP): For the third experiment, we considered traceroute paths collected during a significant misbehavior that occurred at AMS-IX, on May 13th 2015 at around 10:00 UTC. The outage was observed by RIPE Atlas probes [43] and was reported on many Network Operator Groups (NOGs). According to an atnog.at e-mail by the NOC manager of AMS-IX [44], a hardware loop on an access switch was mistakenly introduced during maintenance. The problem caused most of the traffic passing through the IXP to be re-routed elsewhere.

For this experiment, we consider traceroutes performed by RIPE Atlas probes toward *anchors*, namely default targets for built-in measurements. Every probe regularly runs built-in measurements against a selected set of anchors. For this case study, we analyzed the traceroutes performed by 1.424 probes distributed all over the world towards 6 anchors located in Amsterdam. We analyzed the traceroutes towards these anchors in the 48 hours comprising the event. Each probe runs a traceroute every 15 minutes towards a subset of the 6 anchors. On average, every anchor is targeted by 400 probes.

Fig. 11 shows the result of the experiment. The total number of traceroutes is 413.925. Our algorithm took 1 minute and 38 seconds to completely process the input paths, extracting 44.583 transitions. In this case, the threshold computation is based on a period not comprising the event (from 10:00 to 23:59 of May 12th), whose histogram is shown in Figure 12.

Outstanding events are mainly concentrated around two instants: 08:00 UTC and 10:00 UTC. Our algorithm singled out causes pretty accurately. In fact, we manually analyzed the 13 most relevant causes in the peak around 10:00 UTC and discovered that each of them contains a single IP address which either belongs to AMS-IX or to members of AMS-IX, which is coherent with our ground truth. Concerning the peak around 08:00 UTC, unfortunately, we do not have a ground truth. However, analyzing the inferred events, we found that they are all related to traceroutes towards the same anchor within one of the AMS-IX members. All inferred events show routes leaving AMS-IX and starting to pass through upstream providers. We guess that a member of AMS-IX lost connection to the IXP, maybe due to a hardware failure, and this may have

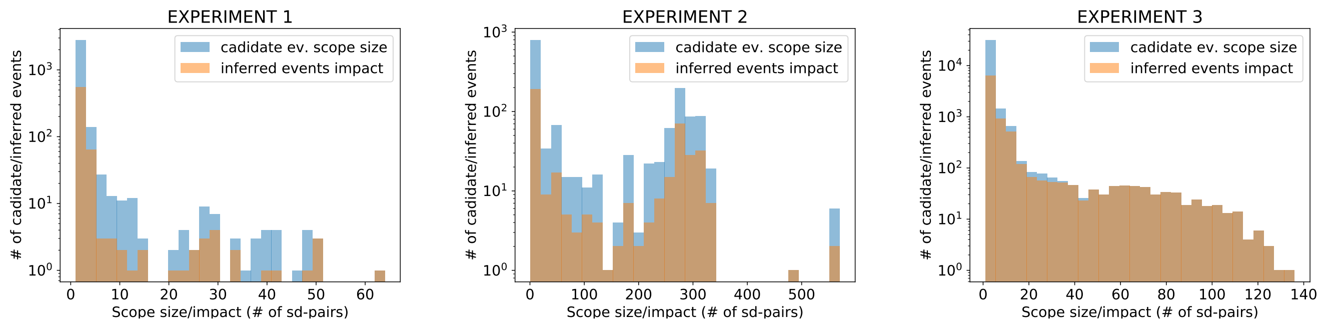


Fig. 13: Densities of the scope size of inferred and candidate events for all three experiments (y-axis log scale).

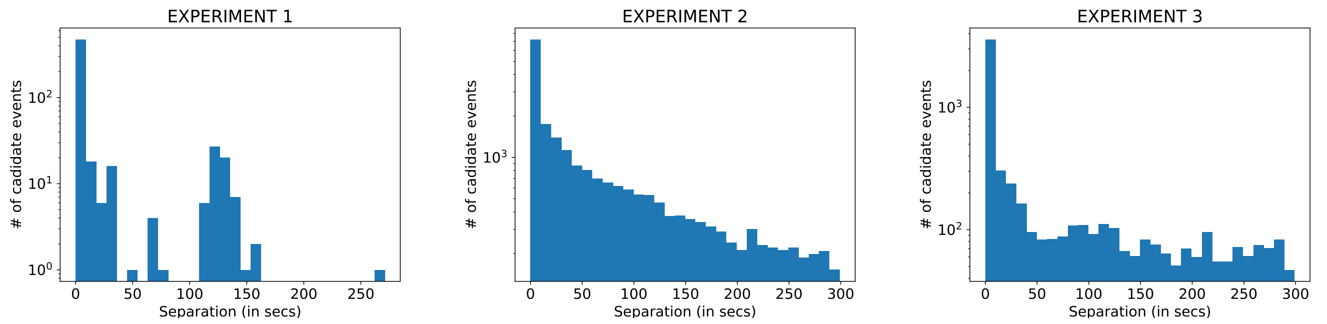


Fig. 14: Density of *separation* between candidate events with the same extended addresses (y-axis log scale).

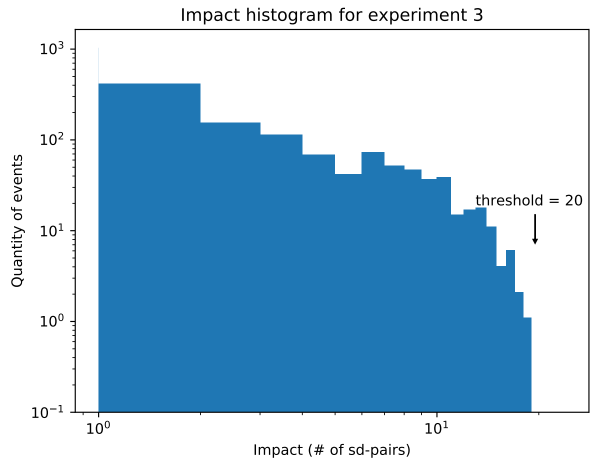


Fig. 12: Histogram of the impacts for the third experiment in doubly logarithmic scale. The computed threshold is highlighted by an arrow.

required the maintenance action occurred at 10:00 UTC.

4) Further Analyses: Now, we further investigate certain details of the behaviour of our approach.

We measured how often an address, appearing in δ^{pre} of a transition τ ending at t_τ, appears in any traceroute collected in the time interval $[t_\tau, t_\tau + 1/f]$, where f is the probing frequency. In Experiments 1 and 3, this occurs about half of the times. In Experiment 2, this occurs in about 98% of the cases. These results show that considering only one transition is not very informative and our approach, which considers many transitions together, provides a clear added value.

Figure 13 shows three charts produced as follows. We recorded all candidate events (CE) with their scope size, i.e., the number of associated sd-pairs. For each experiment, the density of the scope size of CEs is plotted together with the density of the scope size of inferred events, i.e. their impact. Charts clearly show that Phase 3 of our algorithm does have a role, since not all CEs are promoted to inferred events. This is less evident for Experiment 3. We suspect that this is caused by the fact that the outage analyzed in that experiment occurred in The Netherlands, where most of the probes were located, reducing the diversity of the routing paths recorded. In fact, high path diversity gives rise to a large number of CEs with small scope size, while low path diversity tends to create CEs with scope size close to that of the final inferred event.

As described in Section V.5, our approach may produce multiple events when routing propagation suffers some delay due to the fact that multiple skewed similar CEs may be produced. We analysed the *separation* between CEs defined as the distance in time between the start time of a CE and the end time of the last CE with the same extended address. We recorded all the separations according to the above definition. The density of these values, for separations that are less than 5 minutes, are plotted in Figure 14. These charts show that CEs with small separation are present but their amount strongly decay with the increase of separation. This suggests that the practical clean up approach described at the end of Section V.5, based on the closeness in time of inferred events, is viable. According to it, given a certain separation value θ to use as threshold, all events whose separation values appear in the charts at the left of θ should be subject to clean up.

5) Discussion: From our experiments, we can observe that the precision in time of the inferred events is quite high. Further, the reported addresses were always coherent with the ground truth we have. Our approach for noise filtering (Section V.9) turned out to be quite effective in our experiments and may be considered for practical adoption. We observed the effect of routing propagation delay leading to duplicated inferred events. In Section V.5, we propose to filter out duplicates on the basis of similarity of the events and closeness in time. In Section VI.4, we performed a first validation of this approach showing that the vast majority of pairs of events reporting the same addresses are very often close in time. We also investigated if all the steps of the proposed approach are really needed in practice. In Section VI.4, we deeply analyze the data showing that considering only each transaction or considering only each candidate event lead to poor results in a large number of cases.

One may ask if our approach can be adapted to support the timely detection of physical events, quickly after their occurrence. We notice how it is not strictly needed for our approach to know all the transitions in advance. In fact, transitions could be acquired incrementally and, correspondingly, an operator may see the impact of an event to rise over time. The more transitions are acquired the more the reported impact is close to the real one. With this approach, a higher threshold results in postponing when the inferred event is reported. On the other hand, the slope with which the impact of an inferred event increases can be used to estimate its final impact value to anticipate when it is reported.

VII. Conclusions and Future Work

We have presented a model and a methodology to identify and analyze network events based on the notion of empathic traceroute measurements. We have translated our theoretical approach into an algorithm and applied it to real-world data, proving its effectiveness. We think that the availability of this methodology may encourage operators in adopting probe-based traceroute measurements for monitoring and troubleshooting their networks.

Our main future directions of investigation are targeted to obtain (1) an on-line distributed version that is able to scale to an arbitrarily large number of measurements and to timely reports inferred events, (2) a variation of our approach to be applied to the publicly available BGP routing data, (3) a variation of our approach that allows a user to distinguish infrastructure-related problems from other kinds of problems (e.g., congestion-related ones), and (4) an assessment of the added value of correlating network events with other log sources in a production environment.

References

[1] M. Di Bartolomeo, V. Di Donato, M. Pizzonia, C. Squarcella, and M. Rimondini, “Discovering high-impact routing events using traceroutes,” in *2015 IEEE Symposium on Computers and Communication (ISCC)*, July 2015, pp. 295–300.

[2] V. Bajpai and J. Schönwälder, “A survey on internet performance measurement platforms and related standardization efforts,” *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1313–1341, thirdquarter 2015.

[3] M. Bagnulo, P. Eardley, T. Burbridge, B. Trammell, and R. Winter, “Standardizing large-scale measurement platforms,” *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 2, pp. 58–63, 2013.

[4] M. Linsner, P. Eardley, T. Burbridge, and F. Sorensen, “Large-Scale Broadband Measurement Use Cases,” RFC 7536 (Informational), Internet Engineering Task Force, May 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7536.txt

[5] L. Fang, A. Atlas, F. Chiussi, K. Kompella, and G. Swallow, “Ldp failure detection and recovery,” *IEEE Communications magazine*, vol. 42, no. 10, pp. 117–123, 2004.

[6] N. Duffield, “Network tomography of binary network performance characteristics,” *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5373–5388, 2006.

[7] “SamKnows,” 2017. [Online]. Available: https://www.samknows.com

[8] “RIPE Atlas,” 2017. [Online]. Available: http://atlas.ripe.net

[9] “PlanetLab,” 2017. [Online]. Available: www.planet-lab.org

[10] “Ark,” 2017. [Online]. Available: http://www.caida.org/projects/ark

[11] M. Bagnulo, P. Eardley, T. Burbridge, and J. Schönwälder, “Information model for large-scale measurement platforms (LMAPs),” Internet Requests for Comments, RFC Editor, RFC 8193, Aug 2017. [Online]. Available: http://www.rfc-editor.org/rfc/rfc8193.txt

[12] J. Schönwälder and V. Bajpai, “A YANG data model for LMAP measurement agents,” Internet Requests for Comments, RFC Editor, RFC 8194, Aug 2017. [Online]. Available: http://www.rfc-editor.org/rfc/rfc8194.txt

[13] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. E. Anderson, “Studying black holes in the Internet with Hubble.” in *Proc. NSDI*, 2008.

[14] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang, “Planetseer: Internet path failure monitoring and characterization in wide-area services,” in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, ser. OSDI’04. Berkeley, CA, USA: USENIX Association, 2004, pp. 12–12. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251254.1251266

[15] E. Katz-Bassett, C. Scott, D. R. Choffnes, Í. Cunha, V. Valancius, N. Feamster, H. V. Madhyastha, T. Anderson, and A. Krishnamurthy, “Lifeguard: Practical repair of persistent route failures,” *ACM SIGCOMM Comput. Commun. Review*, vol. 42, no. 4, pp. 395–406, 2012.

[16] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, “Netdiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data,” in *Proc. CoNEXT*, 2007.

[17] M. Caesar, L. Subramanian, and R. H. Katz, “Towards localizing root causes of bgp dynamics,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-03-1292, 2003. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2003/6364.html

[18] “Routing information service (RIS),” RIPE-NCC. [Online]. Available: http://ris.ripe.net

[19] “Routeviews,” University of Oregon. [Online]. Available: http://www.routeviews.org/

[20] D.-F. Chang, R. Govindan, and J. Heidemann, “The temporal and topological characteristics of bgp path changes,” in *11th IEEE International Conference on Network Protocols, 2003. Proceedings.*, Nov 2003, pp. 190–199.

[21] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, “Locating Internet routing instabilities,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 205–218, Aug. 2004.

[22] R. Teixeira and J. Rexford, “A measurement framework for pin-pointing routing changes,” in *Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality*, ser. NetT ’04. New York, NY, USA: ACM, 2004, pp. 313–318. [Online]. Available: http://doi.acm.org/10.1145/1016687.1016704

[23] U. Javed, I. Cunha, D. Choffnes, E. Katz-Bassett, T. Anderson, and A. Krishnamurthy, “Poiroot: Investigating the root cause of interdomain path changes,” *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 183–194, Aug. 2013.

[24] B. Al-Musawi, P. Branch, and G. Armitage, “Bgp anomaly detection techniques: A survey,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 377–396, Firstquarter 2017.

[25] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, “Ip fault localization via risk modeling,” in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 57–70.

[26] ——, “Detection and localization of network black holes,” in *Proceedings of the IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. Washington, DC, USA:

IEEE Computer Society, 2007, pp. 2180–2188. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2007.252

[27] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 13–24, Aug. 2007. [Online]. Available: http://doi.acm.org/10.1145/1282427.1282383

[28] R. N. Mysore, R. Mahajan, A. Vahdat, and G. Varghese, "Gestalt: Fast, unified fault localization for networked systems," in *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIX ATC'14. Berkeley, CA, USA: USENIX Association, 2014, pp. 255–268. [Online]. Available: http://dl.acm.org/citation.cfm?id=2643634.2643662

[29] R. R. Kompella, J. Yates, A. G. Greenberg, and A. C. Snoeren, "Detection and localization of network black holes." in *INFOCOM '07*.

[30] Í. Cunha, R. Teixeira, N. Feamster, and C. Diot, "Measurement methods for fast and accurate blackhole identification with binary tomography," in *Proc. IMC*, 2009.

[31] Y. Huang, N. Feamster, and R. Teixeira, "Practical issues with using network tomography for fault diagnosis," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 5, pp. 53–58, 2008.

[32] L. Ma, T. He, A. Swami, D. Towsley, K. K. Leung, and J. Lowe, "Node failure localization via network tomography," in *Proc. IMC*, 2014.

[33] N. Brownlee, "On searching for patterns in traceroute responses," in *Passive and Active Measurement: 15th International Conference, PAM 2014, Los Angeles, CA, USA, March 10-11, 2014, Proceedings*, 2014, pp. 67–76.

[34] M. Candela, M. Di Bartolomeo, G. Di Battista, and C. Squarcella, "Radian: Visual exploration of traceroutes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 7, pp. 2194–2208, Jul 2018.

[35] C. D. Murta, P. R. Torres Jr, and P. Mohapatra, "Qrpp1-4: Characterizing quality of time and topology in a time synchronization network," in *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*. IEEE, 2006, pp. 1–5.

[36] P. Marchetta, V. Persico, and A. Pescapè, "Pythia: yet another active probing technique for alias resolution." in *Proc. CoNEXT*, 2013.

[37] K. Keys, Y. Hyun, M. Luckie, and K. Claffy, "Internet-scale ipv4 alias resolution with MIDAR," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 2, pp. 383–399, 2013.

[38] B. Augustin, T. Friedman, and R. Teixeira, "Measuring load-balanced paths in the Internet," in *Proc. IMC*, 2007.

[39] 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. IMC*, 2006.

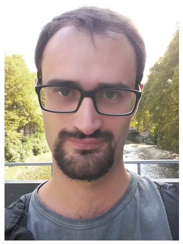
[40] H. X. Nguyen and P. Thiran, "Active measurement for multiple link failures diagnosis in ip networks," in *International Workshop on Passive and Active Network Measurement*. Springer, 2004, pp. 185–194.

[41] C. Chaudet, E. Fleury, I. G. Lassous, H. Rivano, and M.-E. Voge, "Optimal positioning of active and passive monitoring devices," in *Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology*, ser. CoNEXT '05. New York, NY, USA: ACM, 2005, pp. 71–82. [Online]. Available: http://doi.acm.org/10.1145/1095921.1095932

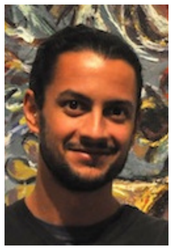
[42] M. Di Bartolomeo, G. Di Battista, R. di Lallo, and C. Squarcella, "Is it really worth to peer at ixps? a comparative study," in *Computers and Communication (ISCC), 2015 IEEE Symposium on*. IEEE, 2015, pp. 421–426.

[43] R. Kisteleki, "The AMS-IX outage as seen with RIPE atlas," https://labs.ripe.net/Members/kistel/the-ams-ix-outage-as-seen-with-ripe-atlas, 2015.

[44] K. Koutalis, "Outage post-mortem: 13 May 2015, 100GE loop on AMS-IX ISP Peering LAN," https://atnog.at/pipermail/atnog/2015-May/000039.html, 2015.



Marco Di Bartolomeo has a Ph.D. in Computer Science and Engineering from Roma Tre University, Italy, where he worked in the Graph Drawing and Network Visualization group. His research interests include graph drawing, network visualization, graph algorithms, and temporal data visualization. Since 2016 he has worked as a software engineer at Google, Inc.



Valentino Di Donato has a Ph.D. in Computer Science and Engineering from Roma Tre University, Italy, where he worked in the Computer Networks and Data Visualization group. His research interests include network data analysis, graph algorithms, and data visualization. Since 2017 he has worked as a Data Visualization Expert at CGnal, Ltd.

Maurizio Pizzonia is Assistant Professor at Roma Tre University. His research interests are about algorithms and software systems for Internet analysis, data and cloud security, blockchain, and information visualization. He had management roles in international research projects about networking and security. In 2011, he founded a company about cloud storage integrity. The most successful and lasting projects he initiated are BGPlay, a tool to visually explore open inter-domain routing data also adopted by RIPE NCC, and Netkit, a teaching tool for realistic emulation of computer networks.

Massimo Rimondini got his PhD in Computer Science and Automation in 2007 at the Roma Tre University. His thesis, entitled "Interdomain Routing Policies in the Internet: Inference and Analysis", marked the start of 9 years of research in the field of Internet routing, especially focused on the control plane comprising modeling and improving the behavior of routing protocols, analyzing their performance and inferring routing events and their causes. His contributions appeared in top-ranked international journals published by IEEE and ACM and were presented at several international conferences. In 2016 he moved to an Internet Service Provider company, where he currently leads the Network Design & Engineering group and is in charge of expanding the backbone and handling relationships with other service providers.

Claudio Squarcella has a Ph.D. in Computer Science and Engineering from Roma Tre University, Italy, where he worked in the Graph Drawing and Network Visualization group. His research interests include network visualization, graph drawing, and network data analysis. He worked as a software engineer at ThousandEyes, Inc., and since 2016 he has been at Sysdig.