Reduce, Reuse, Recycle: Repurposing Existing Measurements to Identify Stale Traceroutes

Vasileios Giotsas[†] Thomas Koch[‡] Elverton Fazzion[♮◇] Ítalo Cunha[◇] Matt Calder[♯‡]
Harsha V. Madhyastha[★] Ethan Katz-Bassett[‡]

[†]Lancaster University [‡]Columbia University [◇]Universidade Federal de Minas Gerais
[♮]Universidade Federal de São João del-Rei [♯]Microsoft [★]University of Michigan

ABSTRACT

Many systems rely on traceroutes to monitor or characterize the Internet. The quality of the systems' inferences depends on the completeness and freshness of the traceroutes, but the refreshing of traceroutes is constrained by limited resources at vantage points. Previous approaches predict which traceroutes are likely out-of-date in order to allocate measurements, or monitor BGP feeds for changes that overlap traceroutes. Both approaches miss many path changes for reasons including the difficulty in predicting changes and the coarse granularity of BGP paths.

This paper presents techniques to identify out-of-date traceroutes without issuing any measurements, even if a change is not visible at BGP granularity. We base our techniques on two observations. First, although BGP updates encode routes at AS granularity, routers issue updates when they change intra-domain routes or peering points within the same AS path. Second, route changes correlate across paths, and many publicly available traceroutes exist. Our techniques maintain an atlas of traceroutes by monitoring BGP updates and publicly available traceroutes for signals to mark overlapping atlas traceroutes as stale. We focus our analysis of traceroute path changes at the granularity of border router IPs which provides an abstraction finer than AS- or PoP-level but is not affected by the periodicity of intra-domain load balancers. Our evaluation indicates that 80% of the traceroutes that our techniques signal as stale have indeed changed, even though the AS hops remained the same. Our techniques combine to identify 79% of all border IP changes, without issuing a single online measurement.

CCS CONCEPTS

• **Networks** → **Network dynamics**; **Network monitoring**; *Public Internet*;

KEYWORDS

Internet topology, routing, traceroute, path changes, measurements.

IMC '20, October 27–29, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8138-3/20/10…$15.00
https://doi.org/10.1145/3419394.3423654

ACM Reference Format:
Vasileios Giotsas, Thomas Koch, Elverton Fazzion, Italo Cunha, Matt Calder, Harsha V. Madhyastha, and Ethan Katz-Bassett. 2020. Reduce, Reuse, Recycle: Repurposing Existing Measurements to Identify Stale Traceroutes. In *ACM Internet Measurement Conference (IMC '20), October 27–29, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3419394.3423654

1 INTRODUCTION

To monitor routing, predict performance, or make other inferences, many systems gather an atlas of traceroutes from a distributed set of vantage points [12, 54, 67]. Some systems monitor between all pairs of vantage points [5, 14, 16, 30, 72, 76]. Others measure from all vantage points to a set of targets [18, 44, 44, 45, 50, 76, 83]. Others choose custom targets per vantage point [7, 19, 22, 57]. The fidelity of these systems' observations or predictions depends on the completeness and accuracy of their views of Internet routing, which they must refresh to account for path changes.

Unfortunately, vantage points have limited probing budgets to restrict bandwidth utilization and other overheads in the networks hosting them. For example, CAIDA limits Ark monitors to 100 probe packets per second [12], and RIPE Atlas limits the probing rate of Probes to 6 Kbps [65]. Saturating these limits is also not advisable as interference between overlapping probes can negatively affect precision and synchronization of measurements [35].

These constraints mean that Internet-scale systems cannot frequently reissue traceroutes along all paths. To best cope with probing rate limits, a system must remeasure a path only when it is likely to have changed. But, how can a system make this determination? Path changes occur at arbitrary times and their frequency varies across paths [19], so remeasuring all paths periodically or in a random order wastes some measurements on unchanged routes, takes too long to detect some changes, and misses some changes altogether.

Prior work that attempted to detect route changes in a timely manner suffers from two significant limitations:

(1) *Detection via direct measurements has limited coverage.* DoubleTree [42], DTrack [19], and Sibyl [18] attempt to reduce the measurement cost to infer changes. However, techniques that require *any* measurements to test if a path has changed have two undesirable properties: (1) for a fixed measurement budget, the ability to keep traceroutes up-to-date is inversely proportional to the number of paths, and (2) many measurements will be "wasted" on paths that remain unchanged.
(2) *Detection via AS paths in BGP updates is coarse-grained.* Other approaches detect route changes by passively monitoring

BGP updates [18, 26, 45]. However, they only detect changes visible at the AS level; intradomain changes and changes at peering points go undetected.

To overcome these limitations of prior work, in this paper, we develop techniques which detect *fine-grained route changes* with *broad coverage* at *no measurement cost*. First, while BGP encodes routes at AS granularity, routers still issue updates when changing routes at finer granularities. We treat updates as signals that a route may have changed, even when the BGP AS-path remains unchanged. Second, we crawl publicly available traceroutes from measurement platforms to identify route changes. We treat changes as signals that overlapping routes might have changed.

Our techniques combine to detect 79% of the IP-level changes in border routers across two months of a daily RIPE Atlas traceroute campaign that generated 8 million traces per day, all without issuing any online measurements. Achieving high coverage requires recognizing routes that are impacted by a BGP update or changes observed in public traceroutes, and so our techniques target cases beyond easy ones that directly observe the path change. Our techniques are precise: 82% of the traceroutes that they indicate as out-of-date, in terms of border-level IP interconnections, have actually changed. Achieving high precision requires avoiding falsely associating an event with an unaffected traceroute, and so we develop techniques to scope impact.

Our techniques can be easily integrated into other systems that rely on up-to-date traceroutes. When our techniques flag a traceroute as likely to have changed, the system using the traceroute can take a system-specific action, such as reissuing a traceroute, discarding the route, or treating it as less trustworthy.

2 MOTIVATION

Many systems use corpuses of traceroutes. CDNs measure traceroutes to destinations around the Internet to aid in performance-aware traffic engineering [15, 33, 82]. Network tomography and troubleshooting systems use traceroute corpuses to identify links or networks responsible for a failure or performance problem [30, 45, 46, 57, 76]. Internet measurement and prediction systems rely on correlating paths in traceroute corpuses [18, 44, 50]. The inferences made by these systems degrade if they use out-of-date traceroutes that no longer reflect active routes.

Large corpuses cannot be refreshed frequently because measurement platforms have limited measurement budget. Achieving high Internet coverage requires topologically distributed vantage points [15, 16, 18]. The usual approach to achieving this goal is to develop collaborative platforms where networks hosting vantage points contribute resources to measurements. Given their collaborative nature and the requirement of not impacting network traffic and equipment, platforms such as Ark [12], RIPE Atlas [67], and SpeedChecker [69] operate under strict measurement budget constraints. The rate at which measurements can be issued on these platforms is further limited by recent observations that overlapping measurements are best avoided to minimize interference [35].

Existing approaches for updating traceroute corpuses are ineffective and inefficient. Ark [12] and iPlane [50] measure paths following a round-robin schedule, too infrequent for some paths and too frequent for others, since the stability of paths varies greatly [19, 81].

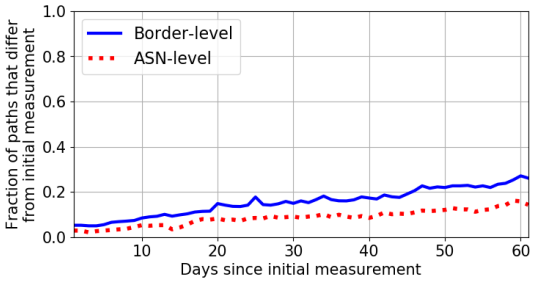


Figure 1: Fraction of paths with border-level and AS-level changes compared to initial traceroute over time. Most paths are unchanged even after two months.

DTrack observes previous changes to predict when a path is likely to change again [19] but ignores that changes (or lack thereof) in one path have implications for which other paths are likely to change (or remain stable). Donnet proposed a system that triggers traceroutes to a destination prefix in response to changes in the AS-path and communities observed in the prefix's BGP updates [21]. The approach, however, does not infer which traceroutes may be impacted by an observed AS-path or community change. Moreover, most ASes with traceroute vantage points do not provide public BGP feeds, so there may be no public BGP updates that indicate when the AS-path corresponding to a previously measured traceroute changes.

Repurposing of public traceroutes requires sacrificing coverage or freshness. RIPE Atlas and Ark collect and make publicly available a massive number of traceroutes each day, which make up a particularly large corpus gathered across many individual measurement campaigns. This corpus is attractive as a source of data for new use cases. However, each measurement campaign is subject to limited probing rates and individualized decisions of which paths to measure, and so the per-path inter-measurement interval varies and can be extremely long. So, relying on existing public traceroutes for a new use case necessitates either including very old traceroutes–some of which are certainly stale due to subsequent path changes, which can cause incorrect inferences–or only using recent traceroutes, severely limiting coverage.

Takeaways. In summary, there exist many systems whose efficacy depends on their ability to maintain up-to-date views of Internet routing, but they lack good techniques to cope with limitations on probing rates. If the systems had the ability to infer which paths have changed and which remain stable, they could smartly allocate probes to refresh only stale portions of the corpus, and they could use only those public traceroutes that remained unchanged. To illustrate this idea, we examined traceroutes issued from 897 sources to 497 destinations every 15 minutes for two months (details in §5.1.1). Our results in Figure 1 depict the fraction of paths that differ from their initial traceroute at different path granularities. Changes over time are not monotonic since a path may change then revert back to its initial measurement. After 30 days, 16% of paths have changed if we consider just the subset of routers at inter-AS borders. (We consider this granularity to eliminate most changes caused by ECMP load balancing [6]). For context, CAIDA's Ark IPv4 Routed /24 Topology Dataset uses a probing rate that would allow a vantage point to cycle through one traceroute to each /24

in 34-51 days, and so these traceroutes would be stale before the vantage point could cycle back. On the other hand, 72% of paths are unchanged at this granularity even after 2 months. So, efficient identification of changed paths could keep the corpus up-to-date with a low probing rate, focusing measurements where needed. Similarly, identification could enable, for example, safe use of the majority of public traceroutes issued over the previous two months (and judicious exclusion of those that were stale).

Hence, to fully capitalize on the potential to reuse traceroutes, we need the ability to determine when any previously measured traceroute path has changed, but without having to issue any measurements to make this determination. This paper describes a set of techniques which can do so in combination.

3 GOAL

Our goal is to create a system that keeps a *corpus of traceroutes* up-to-date by either refreshing or pruning a traceroute if the measured path changes. Changes at different granularities may matter to different use cases. At the finest granularity, a path is a sequence of IP addresses between a source and a destination. At the coarsest granularity we consider, a path is a sequence of AS hops. In between, we consider a path as a sequence of border routers, each of which has one or more IP addresses (aliases), abstracting away the intra-AS topology while still identifying multiple links between a pair of ASes. For our purposes, we consider a change to be an AS-level change if one or more of the ASes on the path changes, and we consider it to be a border-level change if one or more border routers change but it is *not* an AS-level change (so the AS-path remains the same, and the border change is at one or more interconnection points between the ASes).

In this paper, we focus on AS-level and inter-domain (border) IP-level changes, rather than intra-domain IP-level changes for two reasons. First, many use cases operate at these granularities, including topology discovery and mapping [24, 56], evaluation of the resilience and robustness of Internet connectivity [29, 43], measurement of inter-domain congestion [20, 48] and traffic engineering [55], and analysis of peering strategies [60]. Second, intra-AS IP-level changes can happen frequently at short time scales due to load balancing, load sharing, and tunneling, which rarely extend across AS boundaries [6, 28, 78]. As a result, intra-domain path dynamics exhibit a much higher degree of periodicity compared to inter-domain changes [38]. Extending to intra-AS IP-level changes is an interesting future direction.

4 METHODOLOGY

To achieve unprecedented coverage in our ability to detect when a path is likely to have changed, we developed two sets of complementary techniques. The techniques passively monitor existing data streams to detect *staleness prediction signals* that suggest that particular traceroutes in the *corpus* are out-of-date because of path changes. One set of techniques rely on BGP feeds (§4.1) and the other set leverages publicly available traceroute datasets (§4.2). Section 4.3 describes how our techniques can be applied to monitoring systems to help keep a corpus of traceroutes up-to-date. Figure 2 provides an overview of our technique and their sections and data sources. Appendix A describes the (existing) techniques we use

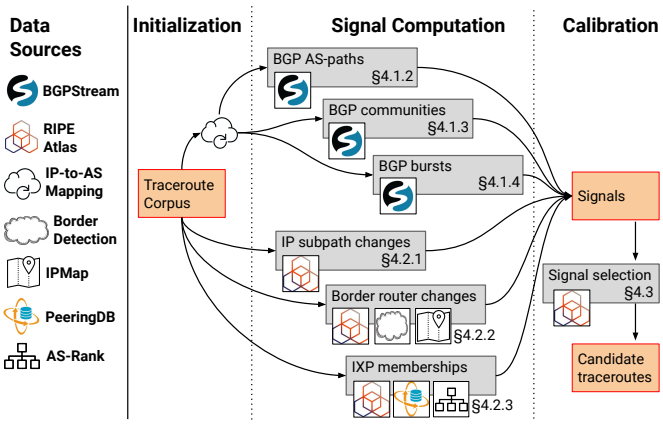


Figure 2: Methodology overview showing data sources (left), proposed techniques (gray boxes), and which data sources are used by each technique. IP-to-AS mapping and AS border detection use multiple data sources (Appendix A).

to map an IP-level traceroute to the coarser granularities and to geolocate IP addresses.

4.1 Detecting Changes from Public BGP Feeds

We seek to use BGP data not to build snapshots of the Internet's AS-level routing topology, but to detect when a traceroute in our corpus is likely to be out-of-date and incorrect. This problem is challenging because BGP routing activity is not necessarily reflected in traceroutes, and vice-versa. BGP provides an AS-path towards an IP prefix, a coarser granularity than the traceroute corpus's IP-level path to a (specific) IP destination. Moreover, vantage points that provide BGP data (e.g., RouteViews and RIPE RIS collectors) differ from vantage points from which traceroutes can be gathered (e.g., Ark, RIPE Atlas), both in terms of host ASes and geographic locations.

We overcome these limitations by developing techniques that monitor BGP feeds to infer IP-level border changes, where both types of changes need not result in any route changes at the AS level (i.e., not visible in BGP AS-paths). Our techniques consider BGP data obtained from BGP route collectors in a new light: an update from a BGP router signals that the router has changed its routing configuration, even if the update carries the same AS-path as the previous update. Our techniques extract signals of possible path changes by correlating BGP activity across time and across vantage points.

First, similar to earlier work [18], we identify traceroute *staleness prediction signals* by looking for changes in overlapping BGP AS-paths. Second, we monitor for changes in the BGP communities attribute, as such changes may indicate a change in the border IPs, even if the AS-path remains unchanged [21]. Third, we rely on the fact that many routers issue updates whenever they change a route, even though the transitive attributes may be identical to those of the previous update. Before describing any of these techniques, we discuss how we collect and preprocess BGP data.

4.1.1 Initializing BGP feeds to monitor. We use BGPStream [61] to stream updates from RouteViews and RIPE RIS collectors and

to maintain BGP table views for every route collector peer. We exclude prefixes more specific than /24, as such prefixes generally do not propagate far [9] and may indicate misconfigurations or blackholing [31]. We also strip from all AS-paths any AS used by Internet exchange points (IXPs) [63], so as to include AS links between IXP members and not the IXP itself. For every destination in the traceroute corpus, we find the most specific prefix advertised by each BGP vantage point (VP), a router peering with a BGP collector, and we monitor for updates in the VP's route to the prefix. Note that different ASes may use a different prefix to the same destination due to the presence of overlapping prefixes in the routing system [40].

4.1.2 Inference of AS-level path changes. Given a traceroute τ_d to a destination IP d measured at time t_0, traversing AS hops $\{a_1, a_2, \ldots, a_d\}$, we determine the times at which AS hops change in the BGP paths that intersect with τ_d.

In every fixed-duration time window $w_i = [t_i, t_{i+1}), t_i > t_0$,[1] we find the set of AS paths to d that overlap the traceroute (i.e., include some AS a_j from τ_d), denoting each by $\rho_{v,d,t'}$ for $t_i \leq t' < t_{i+1}$; ρ_{v,d,t_i} is the AS path used by VP v at the beginning of the window, and $\rho_{v,d,t'}$ for $t_i < t'$ indicates a BGP update from v at time t' changing its AS path. From these path segments, we compute the set of paths $P^{\text{intersect}}_{i,a_j,d}$ whose AS-paths first intersect τ_d at a_j, i.e., a_j is the AS farthest from the destination that is in both τ_d and $\rho_{v,d,t'}$. We also compute the subset of paths $P^{\text{match}}_{i,a_j,d} \subseteq P^{\text{intersect}}_{i,a_j,d}$ whose AS hops from a_j to the destination traverse the same ASes $a_j \ldots a_d$ as τ_d. For each time window w_i and AS $a_j \in \tau_d$, we compute the ratio of paths that match the current traceroute in the corpus, $P^{\text{ratio}}_{i,a_j,d} = |P^{\text{match}}_{i,a_j,d}| \div |P^{\text{intersect}}_{i,a_j,d}|$.

We monitor the resulting time series for $P^{\text{ratio}}_{i,a_j,d}$ across time windows and generate a *staleness prediction signal* when the Bitmap detection algorithm identifies an outlier in the time series [79]. We selected a statistical method for outlier detection instead of a machine learning approach because statistical methods can achieve better accuracy and execution time for univariate time series with no seasonality [8]. If $P^{\text{intersect}}_{i,a_j,d} = \emptyset$, we consider the value $P^{\text{ratio}}_{i,a_j,d}$ as missing and not as an outlier. Since $P^{\text{intersect}}$ and P^{match} count path updates rather than VPs, outliers in $P^{\text{ratio}}_{i,a_j,d}$ can capture both shifts of VPs away from the overlapping path and periods of routing instability, enabling the detection of IP-level changes even when an AS path reverts back to its original hops.

We need to avoid changes to the time series (and possible outliers) caused by variation in the underlying set of VPs over time. To achieve this, $P^{\text{ratio}}_{i,a_j,d}$ is computed over only the set of VPs that intersected τ_d at a_j when the traceroute was issued at t_0.

To prevent persistent path changes from introducing level-shifts in the time series for $P^{\text{ratio}}_{i,a_j,d}$ that obscure the detection of further outliers, we remove time windows flagged as outliers in order to preserve the stationarity of the time series [17, 75], so the persistent change will continue to register as an outlier suggesting the traceroute is stale.

[1]The time window duration is a function of the frequency at which public data is made available. In this paper we use a time period of 15 minutes in our analysis because it is the duration of a RouteViews BGP dump cycle. Since RIPE RIS dumps BGP messages every 5 minutes, a 15-minute window allow us to combine BGP messages from both projects in every time window.

```
TIME: 09/23/2020 10:00:12
TYPE: TABLE_DUMP_V2/IPV4 UNICAST
FROM: 195.66.224.175 AS13030
ASPATH: 13030 1299 2914 18747
COMMUNITY: 13030:2 13030:1299 13030:7214 13030:51701
ANNOUNCE: 200.61.128.0/19
```

```
TIME: 09/23/2020 12:00:12
TYPE: TABLE_DUMP_V2/IPV4 UNICAST
FROM: 195.66.224.175 AS13030
ASPATH: 13030 1299 2914 18747
COMMUNITY: 13030:2 13030:1299 13030:7173 13030:51203
ANNOUNCE: 200.61.128.0/19
```

Figure 3: Example change of BGP communities that indicate the interconnection location of AS13030 and AS1299 (from London (`13030:51701`) to Frankfurt (`13030:51203`)), while the AS path remains unchanged.

4.1.3 Tracking changes in BGP communities. BGP communities are often used to encode properties of a route, such as the geographic location at which an AS learned a route, traffic engineering policies associated with the route, or preferences for how the route is processed (e.g., whether it should be prepended or not exported). These encodings allow a router to communicate information to other routers in its own AS or other ASes.

Figure 3 provides an example. By convention, the top 16 bits of a community indicate the AS that defines it. The figure shows BGP updates from a route collector's peer 195.66.224.175 (in AS13030) to the destination prefix 200.61.128.0/19 at two points in time. The AS-path is the same; the communities, however, differ because border routers of AS13030 signal their locations by adding communities to routes they receive from external peers. Specifically, 195.66.224.175 switched from using a route learned from a router at the `Telehouse LON-1` point-of-presence (PoP) in London (`13030:51701`), to using a route from the `Interxion FRA-3` PoP in Frankfurt (`13030:51203`) [58]. While the AS-level path remains identical, the change of peering point signals a possible IP-level border change in any corpus traceroute to a destination in 200.61.128.0/19 that goes through AS13030.

To infer IP-level border changes based on BGP communities, we monitor for changes in the communities attached to the paths of BGP VPs that overlap an AS-level suffix of a traceroute τ_d. We only consider communities as relevant if they are defined by some AS a_j that intersects τ_d. If the path received from a VP has a change in communities associated with a_j (i.e., a community a_j:xxx is added and/or removed), we consider it a *staleness prediction signal* that τ_d may have changed, except in two cases which we explain next.

First, since communities are an optional transitive BGP attribute, the communities values may be stripped out by any AS along the path. Consequently, we may observe a community appearing or disappearing if there are changes in the AS hops between the intersecting AS a and the VP. For example, suppose a vantage point v's path changes from $\{v, x, a_j, ..., a_d\}$ to $\{v, y, a_j, ..., a_d\}$. If x strips out every community before propagating a route, while y preserves the

communities, then communities may appear in the BGP feed even though the set of communities added to the route never changed. To avoid such artificial changes, if the route changes from having to not having communities (or vice versa), we only consider it a *staleness prediction signal* if the AS-path remains the same. Additionally, if a new community appears on the path from v, but that same community was already on an overlapping AS path from another VP v', we do not consider it as a new signal of change.

Second, while some communities reflect attributes of border IPs (e.g., geolocation communities), other communities have no relation to the traversed IP hops (e.g., control of path prepending). The semantics of BGP communities are defined by each network, and documentation, if even publicly available, follows ad-hoc formats. Additionally, even community values that do relate to properties of border IPs may not relate to the traceroutes in the corpus. For example, the BGP path may traverse a different portion of the AS than the traceroute and may carry a geolocation community for an interconnection point that is not used by the traceroute. To prevent false signals due to these issues, our technique automatically learns over time which BGP communities correlate with changes, using a process we describe in Section 4.3.

4.1.4 Inferring changes from duplicate updates. To catch changes that do not manifest as AS path or community changes, we exploit the observation made in prior work that many seemingly duplicate updates correspond to changes in attributes that are not propagated across AS borders, some of which relate to route changes invisible at the AS granularity (e.g., multi-exit discriminator and IGP cost changes) [34, 62]. While we can associate changes in AS paths or BGP communities to particular ASes that appear on traceroutes, duplicate updates give no direct indication as to which AS triggered them.

To overcome this challenge, we check for contemporaneous duplicate/unchanged updates (i.e., updates without changes to AS-path or communities attributes) to the same destination from multiple BGP collector peers with overlapping AS-paths, suggesting that the change originated on the common subpath shared by the peers. In particular, for each AS-level suffix of the traceroute $\{a_j, \ldots, a_d\}$, we find the set of BGP VPs $\mathcal{V}_0^{j,d}$ that share that suffix at time t_0 when the traceroute τ_d is issued and track the subsets of VPs $\mathcal{V}_i^{j,d} \subseteq \mathcal{V}_0^{j,d}$ that propagate an unchanged update in window w_i. We monitor the time series tracking the number of such peers $U_i^{j,d} = |\mathcal{V}_i^{j,d}|$ for anomalous time intervals (§4.1.2).

A complication happens when some of the peers in $\mathcal{V}_i^{j,d}$ have a common subpath that extends beyond the portion that overlaps the traceroute. For example, consider a subset of BGP collector peers that all share a common AS subpath $\{a_{j-n}, \ldots, a_{j-1}, a_j, \ldots, a_d\}$, but only hops $\{a_j, \ldots, a_d\}$ are common with the corresponding traceroute. A later flurry of updates from these VPs could indicate that the traceroute is out-of-date, or it could be due to a change in $\{a_{j-n}, \ldots, a_{j-1}\}$, which does not overlap the traceroute.

To avoid this issue, we identify each AS a_k that is on the paths of at least 2 VPs in $\mathcal{V}_0^{j,d}$ and is not on the traceroute. For each, we find the set of VPs $\mathcal{V'}_i^{k,d}$ that traverse AS a_k on the way to d but not the entire subpath $\{a_j, \ldots, a_d\}$, and monitor the number of those VPs that propagate an unchanged BGP update $U'^{k,d}_i = |\mathcal{V'}_i^{k,d}|$.

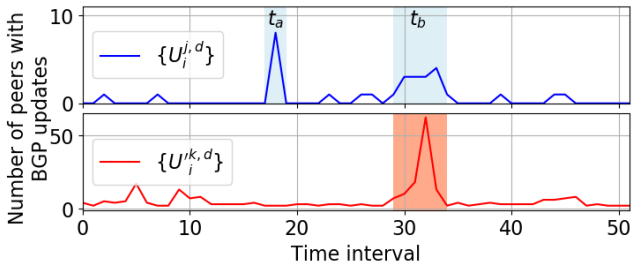


Figure 4: Example correlating bursts of BGP updates to infer potential changes of border-level IP interfaces. The shaded areas indicate time intervals with outliers. There are two time intervals with outliers (t_a and t_b) for $U_i^{j,d}$. The t_a outlier does not coincide with an outlier for $U'^{k,d}_i$, therefore we infer a potential IP-level border path change for τ_d at t_a. In contrast, during t_b both time series exhibit outliers, which indicates that a potential change happened outside the overlapping subpath between the BGP and traceroute paths.

When the set of VPs $\mathcal{V}_i^{j,d}$ have updates in window w_i that are detected as outliers, we check the corresponding series $U'^{k,d}_i$ for any and all ASes $a_k \notin \tau_d$ traversed by the VPs in $\mathcal{V}_i^{j,d}$. If at least one VP did not traverse any other AS a_k experiencing contemporaneous updates, then we generate a *staleness prediction signal* for the traceroute. Figure 4 shows an example of how we correlate the $U_i^{j,d}$ and $U'^{k,d}_i$ time series to infer IP-level border changes.

It is possible that $U_i^{j,d}$ and $U'^{k,d}_i$ may experience update bursts in the same time interval for different root causes. However, since usually at least some VPs in $\mathcal{V}_0^{j,d}$ do not share the same "extra" AS a_k and hence observe AS a_j independent from a_k, the technique can usually differentiate such contemporaneous but independent update bursts from bursts originating only from a_k. More sophisticated root cause detection techniques have been proposed in the past [10, 27, 41], but these works focus on (the simpler case of) bursts of BGP updates that include AS path changes.

4.2 Detecting Changes from Public Traceroutes

To identify changes that do not manifest in the visible BGP dynamics, we passively consume the massive, publicly-available traceroute datasets issued by monitoring platforms such as RIPE Atlas and Ark [12, 67, 68]. For instance, as of April 2020, RIPE Atlas consists of almost 11K active vantage points that collectively issue more than 10K measurements per second that are made publicly available [67].

While public datasets have lots of measurements overall, they have two key limitations due to limited probing budgets. First, most vantage points have recent traceroutes to only a small fraction of destinations, and so the IP-level overlap between the public dataset and the monitored corpus may be small. Second, many paths in public traceroutes are refreshed infrequently, so we cannot rely on directly observing a path change, unlike with BGP monitoring, where essentially every path change comes with an update.

To improve the overlap between public datasets and the monitored corpus to identify which corpus traceroutes have become out of date, our techniques loosen the definition of overlap. First, because the set of public traceroutes is large overall but contains

relatively infrequent traceroutes to most individual destinations, we maintain a sliding window of recent public traceroutes and consider those that overlap a subpath of a corpus traceroute τ regardless of their destination, loosening the destination-based monitoring of Section 4.1. The window size (duration) can be configured based on the use case, the size of the traceroute corpus being maintained relative to the budget (if any) available to refresh it, and the relative impact of removing a traceroute that is still valid versus waiting too long to prune a stale traceroute. Generally, a shorter window size allows for more timely staleness detection, while a longer window size enables staleness detection for more paths. Second, to further increase coverage without compromising accuracy, we use two techniques that offer different tradeoffs between the degree of overlap they require and the granularity of changes they can detect. One technique requires IP-level hop-by-hop overlap along a subpath and can detect IP-level border changes (§4.2.1). The other technique loosens the subpath overlap required, looking for public traceroutes that go from $\langle AS1, city1\rangle$ to $\langle AS2, city2\rangle$ via border router r (possibly with other hops in between), but can only detect router-level border changes (§4.2.2).

Given that our techniques "ignore" that BGP routing is destination-based when deciding which public traceroutes overlap a corpus traceroute, we use two approaches to avoid compromising accuracy. First, we restrict ourselves to subpaths that cross AS boundaries. Interdomain policies are more stable and provide a more reliable signal than intradomain routes, where traffic engineering can introduce dynamic and unpredictable routing reconfigurations. Second, rather than relying on an individual public traceroute as a signal, we look for significant shifts in the relative frequency at which public traceroutes uncover different routes between two points on a corpus traceroute. An individual public traceroute may differ from a corpus traceroute for reasons including load balancing or destination-based routing, whereas shifts in the frequency at which a subpath is observed across many traceroutes suggests meaningful routing changes.

Section 4.2.1 presents the concrete details of the technique for IP-level subpaths, and Section 4.2.2 describes how we adapt it for border-level paths between two cities. Additionally, Section 4.2.3 describes our approach to capture changes in IXP membership. Capturing IXP membership changes allows us to infer concurrent path changes that affect multiple sources and destinations.

4.2.1 Inference of IP-level subpath changes. For each corpus traceroute $\tau_d = \langle \iota_0, \ldots, d\rangle$, we process every subpath $\tau_{\langle m,n\rangle} = \langle \iota_m, \ldots, \iota_n\rangle$, $0 \le m < n \le d$ that traverses at least one inter-AS boundary as follows:

(1) Among *recent* public traceroutes (those within the sliding window), let $T^{\text{match}}_{i,m,n}$ be those that traverse $\tau_{\langle m,n\rangle}$ and $T^{\text{intersect}}_{i,m,n}$ be those that go through ι_m on the way to ι_n (but may follow a subpath other than $\tau_{\langle m,n\rangle}$).
(2) Let $T^{\text{ratio}}_{i,m,n} = |T^{\text{match}}_{i,m,n}| \div |T^{\text{intersect}}_{i,m,n}|$, the fraction of traceroutes within a sliding window w_i between ι_m and ι_n that match τ_d.
(3) We construct the time series of $T^{\text{ratio}}_{i,m,n}$ and generate a *staleness prediction signal* when we detect an outlier in the time-series using the modified z-score introduced in [37]. We use the modified z-score instead of the Bitmap algorithm we used for outlier detection in time series derived from BGP data (§4.1.2), because we found it to be more robust for the noisier traceroute data.

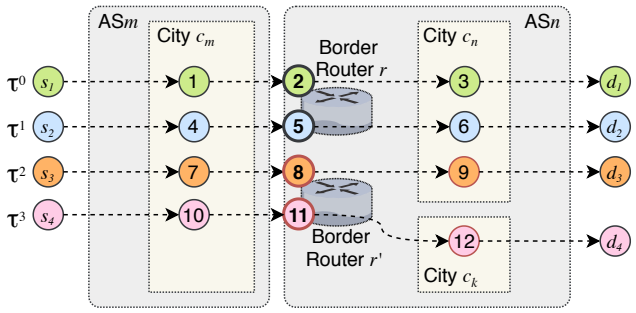


Figure 5: Example of monitoring a path in our traceroute corpus for router-level border changes. Traceroute τ^0 (from s_1 to d_1, green) traverses the subpath $\tau^0_{\langle c_m,c_n\rangle} = \{1,\ldots 2,\ldots 3\}$ between cities c_m and c_n, with hop 2 being on the border router r between two ASes AS_m and AS_n. Traceroute τ^1 (from s_2 to d_2, blue) traverses the same city-level hops through a different subpath $\tau^1_{\langle c_m,c_n\rangle} = \{4,\ldots 5,\ldots 6\}$, with hop 5 on the same border router r as hop 2. As such, both τ^0 and τ^1 are contained in $T^{\text{match}(r)}_{i,c_m,c_n} \subseteq T^{\text{intersect}}_{i,c_m,c_n}$. Traceroute τ^2 (from s_3 to d_3, orange) also traverses the city-level hops c_m and c_n but through a subpath $\tau^2_{\langle c_m,c_n\rangle} = \{7,\ldots 8,\ldots 9\}$, which crosses a different border router r'. Therefore, $\tau^2 \notin T^{\text{match}(r)}_{i,c_m,c_n}$, but $\tau^2 \in T^{\text{intersect}}_{i,c_m,c_n}$ still. Finally, traceroute τ^3 (from s_4 to d_4, pink) traverses hops 10, 11, and 12, and is not part of $T^{\text{intersect}}_{i,c_m,c_n}$ since it does not intersect any IP in city c_n.

Configuration of signal parameters. For each monitored subpath $\tau_{\langle m,n\rangle}$ we require that we have at least 20 consecutive windows, which is widely considered as the minimum recommended number of observations for robust outlier detection [53]. Accordingly, for each $\tau_{\langle m,n\rangle}$ we select the minimum window size that would allow us to collect 20 consecutive windows with data points. We use a minimum window duration of 15 minutes, the window size used in our BGP signals, and a maximum window size of 24 hours, to limit the amount of public traceroutes that we need to accumulate and process to 20 days to avoid resource-scaling and performance issues. If for a given $\tau_{\langle m,n\rangle}$ we have less than 20 consecutive windows with data points, we do not consider this subpath for staleness inferences.

4.2.2 Inference of router-level border changes. If public traceroutes do not reveal a stable distribution as to how frequently we see different paths between ι_m and ι_n, it is difficult to determine whether shifts indicate changes in which paths are in use or unrelated noise (i.e., the time series $T^{\text{ratio}}_{i,m,n}$ is not amenable to outlier detection). However, if public traceroutes indicate that two ASes AS_m and AS_n consistently use a specific border router r to exchange traffic between certain geographical locations (regardless of variations in the IP-level), then later indicate that the same ASes consistently use a different border router r' to transit traffic between the same

locations, it is likely that the ASes changed routing policy, as routing decisions such as early exit will generally be consistent across a PoP or city [70].

We adapt the approach in Section 4.2.1 as follows. Let $T^{\text{intersect}}_{i,c_m,c_n}$ be the set of recent public traceroutes that traverse (any) ι_m located in city c_m and AS_m, and (any) ι_n located in city c_n and AS_n, with $c_m \neq c_n$. Let $T^{\text{match}(r)}_{i,c_m,c_n} \subseteq T^{\text{intersect}}_{i,c_m,c_n}$ be the public traceroutes that traverse the border router r between $\langle \text{AS}_m, c_m \rangle$ and $\langle \text{AS}_n, c_n \rangle$. Figure 5 illustrates these sets. We compute the $T^{\text{ratio}(r)}_{i,c_m,c_n} = |T^{\text{match}(r)}_{i,c_m,c_n}| \div |T^{\text{intersect}}_{i,c_m,c_n}|$, for each time window w_i. The size of the time window is selected as in §4.2.1. When we detect outlier values in the $T^{\text{ratio}(r)}_{i,c_m,c_n}$ time series, we generate a *staleness prediction signal* indicating a border-level change between AS_m and AS_n.

4.2.3 Inference of IXP membership changes. We calculate an initial snapshot of IXP membership at the start of the staleness detection period based on PeeringDB, which we augment with the ASes that appear adjacent to IXP interfaces in traceroutes to account for missing PeeringDB information. We then continue to monitor the ASes that appear as near-end (left-adjacent) neighbors of IXP interfaces in our public traceroute feed. We ignore ASes that appear as far-end neighbors (right-adjacent) of IXPs, since most routers reply with the ingress interface to traceroute probes, which means that the next hop of an IXP interface may not correspond to the AS to which the IXP interface is assigned.

When we detect AS_i as a new IXP_x member, we search for (previously-collected) traceroutes that include AS_i and another AS_j that is already a member of IXP_x in our corpus, since the path may have changed to go directly from AS_i to AS_j via IXP_x. For each such traceroute, we check the relationship between AS_i and the next-hop AS_k used to reach AS_j using CAIDA's AS relationship database [49]. If AS_k is a provider of AS_i, then we generate a *staleness prediction signal*, as AS_i may prefer to send traffic to AS_j directly using the new, cheaper IXP interconnection. If AS_k is a public peer of AS_i (i.e., peering over a different IXP), we also generate a signal, since we assume shortest AS path routing when two neighbors have the same peering relationship (and BGP Local Preference). If AS_k is a private peer (i.e., the peering is not over an IXP interface) we do not generate a signal, since private peers are often assigned higher BGP Local Preference values than public peers [64]. We do generate a signal in the case of routing through a private peer if such changes were observed in public traceroute feeds, which allows us to infer that AS_i assigns equal BGP Local Preference values to public and private peers.

4.3 Using Staleness Prediction Signals

Depending on the goals and constraints of a system, the system may use *staleness prediction signals* to decide which traceroutes to refresh, to prune stale traceroutes, or to treat inferences made with stale traceroutes with lower confidence. This section discusses general approaches to using signals in real systems.

4.3.1 Refreshing Stale Traceroutes and Signal Calibration. Once stale traceroutes are detected in a corpus, in many scenarios it is desirable to issue new traceroutes to refresh them. However, the number of traceroutes signalled as stale can exceed the measurement budget, particularly in systems that require monitoring a large corpus (e.g., [18, 46, 50, 83]).

Each corpus traceroute crosses some set of borders between ASes. Each border may be monitored by zero or more of our techniques depending on the visibility provided by available vantage points. Some of the techniques monitor the use of the border(s) on the way to particular destinations (§4.1), whereas others monitor the use of the border along a subpath independent of destination (§4.2). We say a technique provides a *potential* staleness prediction signal for a border (and associated destination or subpath) that it monitors. We say a potential signal and a corpus traceroute are *related* if the potential signal monitors a border and destination/subpath of the traceroute. At any point in time, some of the potential signals related to a traceroute may indicate that it is stale (i.e., the technique generated a *staleness prediction signal* since the traceroute was issued). Any related potential signal for which a *staleness prediction signal* has not been generated implicitly indicates that the technique has not detected a change at that border.

To prioritize which traceroutes to reissue, we monitor the effectiveness of each potential signal over time and prioritize those that are effective. To capture effectiveness, every time we remeasure a traceroute, we evaluate the correctness of each potential signal related to the (old) traceroute. A signal that indicated a change in a portion of the old traceroute is considered a True Positive (TP) if that portion of the path has actually changed, or a False Positive (FP) if that portion of the path remains unchanged. A potential signal that (implicitly) indicated that a portion of a path had not changed is a True Negative (TN) if that portion remains in the new traceroute, or a False Negative (FN) if that portion is not in the new traceroute. For the vantage point v that issued the traceroute and each related potential signal s, we maintain running tallies $TP_{v,s}$, $FP_{v,s}$, $TN_{v,s}$, and $FN_{v,s}$ over a sliding window of the last $l = 30$ (by default) signal generation windows to allow for changes over time. We use these tallies to maintain the true positive rate $TPR_{v,s} = TP_{v,s}/(TP_{v,s}+FN_{v,s})$ and the true negative rate $TNR_{v,s} = TN_{v,s}/(TN_{v,s} + FP_{v,s})$. Before the initial sliding window is "full" of l windows, we consider $TPR_{v,s}$ and $TNR_{v,s}$ to be uninitialized.

We refresh a number of measurements at the end of each *staleness prediction signal* generation window w_i according to the probing budget available for refreshing the corpus. Let $\mathcal{S}_i$ be the set of staleness signals that predict a change at the end of window w_i, and $\bar{\mathcal{S}}_i$ be the set of potential signals that do not predict a change. Let $\mathcal{S}_i^v \subseteq \mathcal{S}_i$ and $\bar{\mathcal{S}}_i^v \subseteq \bar{\mathcal{S}}_i$ be the subsets related to a traceroute from vantage point v.

We decide which measurement to issue using the following steps:

(1) We first choose the traceroute VP v with the highest relative TPR across all VPs v' with signals in S_i. More precisely, we choose $v = \text{argmax}_v \sum_{s \in S_i^v} TPR_{s,v} / \left(\sum_{v'} \sum_{s \in S_i^{v'}} TPR_{s,v'} \right)$. This selection prioritizes VPs whose measurements more often detect changes, increasing efficacy of the refreshing process.
(2) For simplicity, we calculate a single probability for the selected VP v to refresh each of its traceroutes that a related

staleness prediction signal indicates is stale. The potential signals may not agree on which traceroutes have changed. We combine their "opinions" to decide a probability of refreshing a traceroute, as follows: $P^{\text{refresh}}_{v,i} = \frac{\sum_{s \in S^v_i} TPR_{s,v}}{\sum_{s \in S^v_i} TPR_{s,v} + \sum_{s \in \bar{S}^v_i} TNR_{s,v}}$. This calculation potentially considers *TPR* and *TNR* inferred across multiple traceroutes, multiple borders per traceroute, and multiple potential signals per border. The potential signals may "disagree" on whether or not their monitored portions of traceroutes need to be refreshed. The *TPR* of signals that indicate staleness will drive up the likelihood of refreshing a traceroute, and the *TNR* of potential signals that do not indicate staleness will drive down the likelihood.

(3) For every signal in S^v_i we iterate over all related corpus traceroutes from v–that is, the set that the signal monitors and hence now suggests are stale–and, if measurement budget remains, we issue a remapping traceroute with probability $P^{\text{refresh}}_{v,i}$.
(4) If after executing step 3 there is still measurement budget available, we remove v from the set of VPs that can be selected and we repeat the process from step 1.
(5) While budget remains after the following process, which in particular happens during bootstrapping while $TPR_{v,s}$ and $TNR_{v,s}$ remain uninitialized for many vantage points and signals, we order the signals according to the attributes in Table 1, ordered by their priority from highest to lowest. The first 5 attributes compare the overlap of the traceroutes inferred as stale and the public traceroutes or BGP feed that triggered the inference. When two signals are tied for one attribute, before moving to the next attribute we use the number of VPs as a tie-breaker for BGP-based signals or the deviation from the staleness detection z-score for traceroute-based signals. We use this technique instead of random signal selection to bootstrap our TPR calculations using the best possible signals for each vantage point, so that we avoid building low scores for signals that can be potentially useful for a vantage point but may not be selected due to a bad start.

4.3.2 Revoking stale signals. Paths often change from a preferred prevalent route to a less-preferred route temporarily during disruptions, before changing back to the preferred prevalent route after the disruption is solved [19]. Some of our techniques provide not just a signal of when a corpus traceroute has gone stale but also if it later reverts to its original route. When all AS path (§4.1.2), community (§4.1.3), IP-level subpath (§4.2.1), and inter-city border router (§4.2.2) *staleness prediction signals* associated with a particular corpus traceroute revert to the value they had when the corpus traceroute was issued, we discard the *staleness prediction signals* and consider the corpus traceroute fresh again without reissuing a traceroute.

5 EVALUATION OF PRECISION AND COVERAGE

This section evaluates our techniques in the following scenarios:

(1) Section 5.1 presents a *retrospective evaluation* to evaluate the coverage and precision of our *staleness prediction signals*. We compare traceroutes across consecutive rounds of periodic measurements and assess the relationship between changes and signals that occur between the measurement rounds.
(2) Section 5.2 presents a *live evaluation*, in which we use our techniques to maintain a traceroute corpus for two months. We compare the efficacy of issuing refresh traceroutes using our techniques and random choices.
(3) Section 5.3 *compares our techniques with earlier approaches.*

Table 1: Ordered list of signal attributes used to sort signals by priority when choosing measurements to refresh stale traceroutes during the bootstrap period.

Priority	Signal Attribute
1	Longest IP-level path overlap
2	Longest AS-level path overlap
3	VPs in the same AS and city
4	VPs in the same AS
5	VPs in the same city
6	AS-level change
7	Border-level or IXP change

Metrics. We evaluate the precision and coverage of our techniques in detecting path changes. We define *precision* as the fraction of signals that identify a path change in our dataset, and *coverage* as the fraction of path changes for which our techniques generate signals. Precision is the ratio between the number of correct signals (true positives) and the number of signals (positives). Coverage is the ratio between the number of correct signals and the number of path changes (true positives plus false negatives).[2]

Public BGP, traceroute feeds, and traceroute processing. We collect all the available RouteViews and RIPE RIS data from BGPStream to compute signals using our BGP-based techniques, starting two days prior to the initialization of the corpus of traceroutes. During our measurement period, RouteViews and RIS offered 710 IPv4 VPs in 485 ASes, 84% of which advertised their full BGP table to the collectors. The public traceroutes we use for each scenario are explained in their respective subsections. Appendix A describes standard approaches we use to process traceroutes.

5.1 Retrospective Evaluation

5.1.1 Traceroute corpus. We use the RIPE Atlas anchoring measurements, which issue two types of traceroutes every 900 seconds:

(1) A traceroute to every Atlas Anchor (a device with more CPU, memory, and network bandwidth than regular Probes) from approximately 400 Atlas Probes. The set of Probes can differ across Anchors but is kept stable across rounds for each particular Anchor. If a Probe becomes inactive, it is replaced.
(2) A mesh of traceroutes between all Anchors.

We start collecting anchoring measurements to 497 Anchors on t_0 = 2019-02-15. Every 900-second round, the anchoring measurements produce about 446K traceroutes, 199K traceroutes between Probes and Anchors, and 247K traceroutes between Anchors.

Figure 1 shows the fraction of border-level and AS-level paths that are different from their initial t_0 traceroute over a period of

[2]We use the term *coverage* rather than *recall* (calculated the same way) because false negatives (undetected changes) are mainly caused by a lack of vantage points in the proper locations.

Table 2: Precision and coverage for each path staleness prediction technique for the retrospective evaluation. Each technique has high precision, and combining all techniques is necessary to achieve high coverage.

			Coverage					
			AS or border changes		AS-level changes		Border-level changes	
Technique	#Signals	Precision	Individual	Unique	Individual	Unique	Individual	Unique
BGP AS-paths	377,067	0.82	0.13	0.07	0.28	0.16	0.05	0.02
BGP communities	267,571	0.80	0.09	0.05	0.03	0.01	0.12	0.07
BGP update bursts	363,368	0.72	0.11	0.03	0.04	0.01	0.14	0.04
BGP Total	1,008,006	0.74	0.27		0.29		0.24	
Colocation changes	305,909	0.85	0.13	0.08	0.12	0.06	0.13	0.10
Traceroute subpaths	1,244,558	0.81	0.51	0.35	0.42	0.23	0.56	0.41
Traceroute borders	261,965	0.83	0.11	0.07	0.19	0.09	0.1	0.05
Traceroute total	1,812,432	0.82	0.69		0.70		0.67	
All techniques	**2,820,438**	**0.80**	**0.81**		**0.86**		**0.79**	

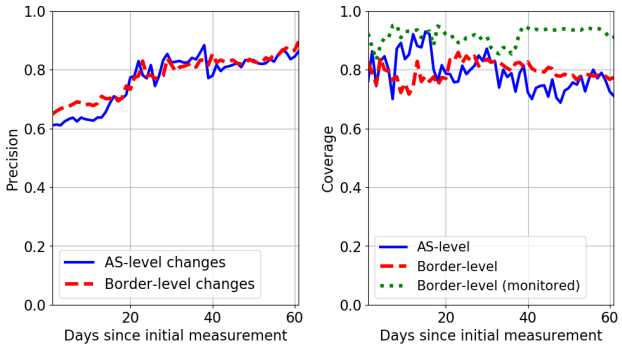
two months. Path changes accumulate over time, with about 15% of the AS-level and 28% of the border-level paths changed at the end of the 60-day measurement period. On the one hand, these fractions are high enough that inferences made using this corpus would be significantly hampered if traceroutes are not refreshed over time, which is not possible for most large-scale traceroute campaigns due to rate limits that prevent refreshing all paths frequently enough. On the other hand, the fraction of "fresh" traceroutes remains high, showing that one can indeed cope with stringent rate limits by reusing archived traceroutes if one is able to detect and filter-out (or selectively refresh) stale traceroutes.

We divide the available VPs, i.e., Atlas Probes and Anchors, into two randomly selected subsets of equal size, P_{public} and P_{corpus}. Our goal is to maintain up-to-date traceroutes from the VPs in P_{corpus} to the Anchors, a corpus of 223K (source, destination) pairs.

5.1.2 Detecting changes. For our traceroute-based techniques, we use the publicly available traceroutes from 4,372 RIPE Atlas VPs in P_{public}, excluding traceroutes toward the targets of the anchoring measurements. By excluding all traceroutes to destinations of the anchoring measurements and public traceroutes from VPs in P_{corpus}, we avoid biasing our results by deriving change signals from traceroutes that closely reflect the corpus we are trying to maintain. This setting is appropriate for evaluating our techniques because it mirrors an intended use case of relying on public RIPE Atlas traceroutes to aid in maintaining a researcher's own RIPE Atlas traceroutes, since it is the most widely used large-scale traceroute platform today, with the most stringent rate limits.

5.1.3 Results. Table 2 presents the number, precision, and coverage of our signals across the 60-day period. All techniques have high precision and contribute unique inferences (i.e., inferences not made by any other technique), so all techniques are useful and necessary to achieve the combined coverage of 79% for border-level changes.

By monitoring for changes in BGP AS-paths and BGP communities as well as for bursts of BGP updates, we are able to identify 27% of the changes overall and 24% of the border-level changes. Further, when our BGP communities technique indicates that a traceroute is stale, the traceroute has actually changed 80% of the time. The technique that monitors for BGP update bursts is not as precise, at 72% precision. Our three traceroute-based techniques identify 69% of the changes, including 67% of border-level changes.



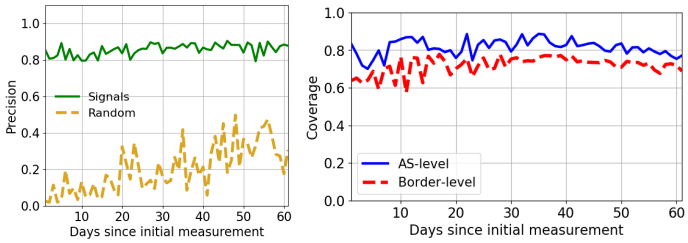
(a) Precision **(b) Coverage**

Figure 6: Precision and coverage of signals across all techniques and VPs for the retrospective evaluation.

Combined they have a precision of 82%, and each of the techniques individually have precision above 81%. Combined with the BGP techniques, we identify 79% of all border-level path changes and 86% of AS-level changes, without issuing any online measurements.

As a comparison point, monitoring of BGP feeds for changes in overlapping AS paths, a technique used in other works [18], only captures 13% of the changes in the traceroute corpus. The poor coverage is partly attributable to the technique only monitoring for changes in AS-level paths, but it only captures 28% of changes even if we limit ourselves to considering AS-level changes in the traceroutes. Since our full set of techniques capture 86% of AS-level changes, they offer significant utility over earlier approaches even for the large number of prior systems (e.g., [41, 83]) which only use traceroutes that have been converted to AS-level paths.

Figure 6a shows the precision across all signals and VPs combined for each day in the evaluation period. At the start of the evaluation period about 60% of the inferences are true. Our techniques for generating signals iteratively improve over time leading to more than 80% correct inferences after the mid-point of the evaluation period and almost 90% at the end, for both AS-level and border-level changes. This high precision implies that a system can effectively use our signals to refresh traceroutes or to identify stale routes. The calibration process is especially beneficial for determining which BGP communities correlate with border-level path changes and which of the VPs in public feeds of routing data



(a) Border-level precision of traceroutes issued using signals versus at random.

(b) Fraction of path changes captured by random traceroutes also captured by staleness inference signals.

Figure 7: Results of live evaluation.

correlate with the traceroute sources in the traceroute corpus (i.e., calibration learns the TPR achieved from different VPs). Appendix B demonstrates how effective tuning of community-based *staleness prediction signals* is at pruning at false positives over time.

Figure 6b shows that coverage is stable over time, and usually above 80%. For the subset of changes that we even have a chance of detecting using one of our techniques (i.e., changes which have some overlap with the data we use to generate signals), coverage is even higher at over 90%. This high coverage implies that a real system can detect most path changes using our signals, particularly on paths where we monitor changes. Moreover, a system using our techniques knows the set of paths that our techniques are capable of monitoring and can treat paths where they lack visibility differently, if necessary. Appendix C examines reasons for the high coverage.

5.2 Live Evaluation

Next, we evaluate the performance of our staleness prediction techniques on a live monitoring system restricted to issue 10 thousand traceroutes per day, RIPE Atlas's rate limit per non-privileged user.

5.2.1 Traceroute corpus. To expand our evaluation to a larger set of destination IPs, we need to sidestep the need for repeated measurements along every path. Therefore, in our live experiment, we evaluate our signals based on the traceroutes we issue to refresh the traceroutes signaled as stale, allowing us to use a larger dataset as our initial traceroute corpus, with more destinations.

Our live evaluation uses the built-in #5051 RIPE Atlas measurement as the initial traceroute corpus. The built-in #5051 measurement aims to map the Internet topology by probing the .1 address in each /24 prefix visible in the RouteViews and RIPE RIS public BGP feeds [66]. A #5051 measurement round is performed every 900 seconds, but not all Probes participate in every round. Since the set of destinations is very large, it is not possible to measure each prefix from every Probe. RIPE Atlas randomly allocates destinations to Probes in every round. We use one day of traceroutes from the #5051 measurement as our initial corpus of 993,948 traceroutes.

5.2.2 Detecting changes. We executed the live evaluation for two months after the initial measurement, issuing 10K "refresh" traceroutes per day at random and 10K using signals generated by our techniques. We use the #5051 measurements on the remaining days to generate signals using our traceroute-based techniques. When the number of signals exceeds the probing quota, we choose traceroutes to refresh based on signal performance for each VP (§4.3).

5.2.3 Results. Figure 7a compares the precision of the traceroutes issued to refresh the corpus (i.e., the fraction that revealed a path change), when they are issued at random or chosen based on *staleness prediction signals*. Our results show that chosen signals have precision generally above 80% across the 2 month evaluation period, while random selection exhibits much lower precision, wasting measurement budget. The figure shows precision for border-level changes, and results for AS-level changes were similar (not shown). Random traceroutes work better over time because more paths change at least once as time progresses.

Figure 7b shows the fraction of traceroute changes captured by the random traceroutes that were flagged by *staleness prediction signals*. We expect the random traceroutes to be an unbiased sample of the (unknown) set of all changes in the monitored paths, so coverage of changes detected by our random traceroutes should be representative of the overall coverage across all changes. For AS-level changes coverage is typically above 80%, while for border-level changes coverage fluctuates around 70–75% after 20 days.

5.3 Comparison with DTRACK and Sibyl

We compare the efficiency of signals for maintaining an up-to-date traceroute corpus (by issuing traceroutes to refresh paths with *staleness prediction signals*) with three other approaches. The first alternative we consider is periodic round-robin route traceroutes, similar to measurement campaigns on CAIDA's Ark and RIPE Atlas.

Second, we consider Sibyl's approach to patch a corpus of existing traceroutes with new traceroutes [18]. Whenever Sibyl remeasures the route to a destination, it compares the new route with the previous route. When Sibyl finds a path change from subpath s to another subpath s', it patches all other traceroutes in the corpus that traverse s to traverse s' instead. We apply Sibyl's patching and pruning on top of periodic traceroutes: any route change detected in periodic traceroutes is used to patch other traceroutes.

Third, we consider DTRACK [19], which shares the same goal of our techniques: reduce measurement cost to more accurately track path changes. DTRACK measures Internet paths once at startup to build the traceroute corpus, then starts a *change detection* phase where it sends single-packet TTL-limited probes to varying hops on measured paths to detect changes. During the detection phase, DTRACK probes each path at a rate proportional to the path's estimated probability of change. Whenever a change is detected, DTRACK runs traceroute to remap the change and update the corpus.

To evaluate the four techniques (ours plus the three alternate approaches) when configured at various rate limits, we gather a dataset at a much higher rate limit to use as a pseudo-ground-truth[3]. We emulate the four approaches by having them decide which measurements to remeasure at what times based on their individual criteria, using the pseudo-ground-truth to determine the result of those measurements, since it has frequent measurements of all paths. As the pseudo-ground-truth, we collect a dataset of path changes between 1–13 April 2019 from a PlanetLab node, using DTRACK to maximize the number of changes detected. Over the period, DTRACK monitored 5500 paths from the PlanetLab node, and the measurements traversed 2819 ASes. including 91% of those

[3]We say *pseudo*-ground-truth because it may still miss short-lived path changes.

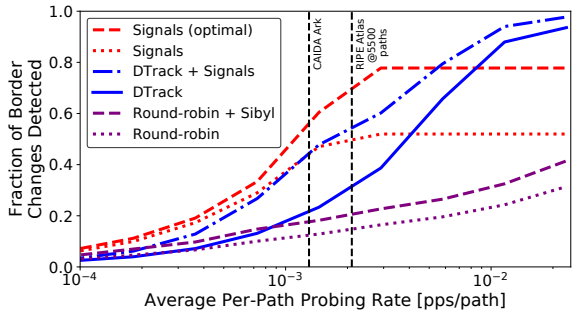


Figure 8: Fraction of changes detected for signals, DTRACK, Sibyl, and round-robin traceroutes as a function of probing budget. Sibyl and round-robin traceroutes miss many changes. Signals outperform DTRACK for low probing budgets, but are limited by their coverage. DTRACK+SIGNALS demonstrates the benefits of combining both.

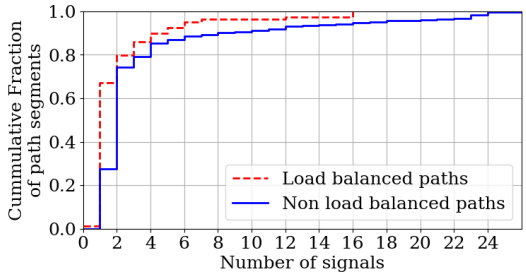


Figure 9: Cumulative distribution of *staleness prediction signals* per load-balanced and non-load-balanced segment.

with more than 500 ASes in their customer cones [49]. This pseudo-ground-truth DTRACK was configured with a detection probing rate of 30 pps and unlimited budget for remapping; the total average probing budget is 67 pps. We *emulate* round-robin traceroutes, Sibyl's patching, DTRACK, and signals on lower probing rates, which would be necessary for a corpus to larger sets of destinations or from constrained vantage points such as RIPE Atlas or Ark.

When emulating Sibyl's patching, we consider an optimistic scenario where we do not patch a traceroute if it would introduce an error relative to ground truth and we do not penalize Sibyl when it occurs. In a real system, a traceroute would be triggered when a signal is generated, but for our trace-driven emulations, we need to match signals with independent, unsynchronized measurements. To accommodate DTRACK, RIPE Atlas, and BGP collectors observing a routing event at different times, e.g., because of BGP propagation delay, batching at BGP collectors, and unsynchronized traceroutes, we allow a signal to match a route change within 30 minutes of the signal's generation window.[4] If there are multiple possible matchings, we choose the maximum matching that minimizes the difference between the start of each signal's window and the change the signal is matched to. If a signal is matched, it triggers a traceroute that remeasures the changed path. If a signal is not matched to any change in the DTRACK dataset, it could be that DTRACK missed a path change, but we conservatively choose to consider the signal a false positive that triggers a traceroute and wastes probing budget since there is no change.

As an additional comparison, we consider an "optimal" mapping of signals that ignores false positives to capture the impact of future improvements to signal accuracy, and matches signals to *all* changes within 30 minutes of their windows to capture potential coverage if public data allowed signal inference at shorter time scales.

Figure 8 shows the fraction of border-level changes detected by each approach at different average per-path probing rates. As probing rate increases, more changes are detected. Signals make efficient use of limited probing budgets, where it outperforms DTRACK, but are limited by their coverage for high probing budgets. Compared to our evaluation on RIPE Atlas data in §5.1, signal coverage is lower in the DTRACK dataset (52%), possibly due to DTRACK's detection of shorter-lived border changes that Atlas traceroutes are not frequent enough to observe. Sibyl's patching improves over periodic traceroutes but is inferior to DTRACK and signals. The figure shows vertical lines indicating the average per-path probing rate for CAIDA's Ark [12] and RIPE Atlas when configured to monitor 5500 paths, as in our dataset. The results show that signals would provide significant benefit for these systems, even if they ran DTRACK instead of periodic traceroutes. Optimal signals outperform all approaches until the probing budget is enough to remap all signals, at which point optimal signals flatlines due to limited coverage. Although the optimal line shows there is room for improvement, our proposed signal-generation techniques and public data already capture most of the changes in the DTRACK dataset.

[4] 30 minutes is a compromise between synchronicity, the signal-generation window duration, and DTRACK's detection probing rate. We expect 30 minutes to cover DTRACK's detection delay, in case signals are generated before DTRACK's detection.

5.4 Impact of Load Balancing

Paths with load balancing can cause traceroutes to change over time, even without an underlying path change, potentially introducing false positive *staleness prediction signals*. Recent work found that 81% of load balanced paths diverged and reconverged within the same AS [77]. These instances of intradomain load balancing do not impact our *staleness prediction signal* techniques, since we focus on detecting path changes that occur at AS borders. So, we evaluate how our techniques act in the uncommon case of interdomain load balancing [3, 77], where the load balanced paths diverge in one AS and reconverge in another AS. We obtained the divergence and convergence IP addresses of interdomain load balanced paths detected by Diamond Miner [77]. The set of load balanced alternatives between a divergence and convergence point is called a *diamond*.

The authors of Diamond Miner provided us with 81,581 interdomain diamonds collected in August 2019. The diamonds include 3,627 divergence IP addresses and 58,647 convergence IP addresses between 5,567 AS pairs (644 divergence ASes and 3,197 convergence ASes). We collected public RIPE Atlas traceroutes between 2019/08/1-14 that traverse the same pairs of divergence and convergence addresses. In total we found 3,181,062 traceroutes that traverse 1,711 diamonds that involve 635 divergence and 1,583 convergence IPs. We computed *staleness prediction signals* as described in Sections 4.2.1 and 4.2.2 (using a 12-hour window). Our techniques did not detect any *staleness prediction signals* (false or otherwise) overlapping 91.2% of the 1,711 diamonds.

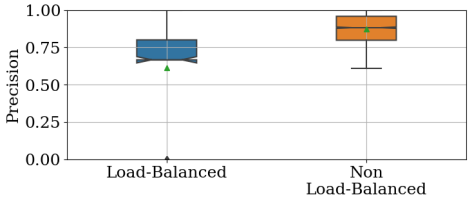


Figure 10: Precision of *staleness prediction signals* for load-balanced and non-load-balanced paths.

To give context to the performance of our techniques on the 168 diamonds (9.8%) where we detect overlapping *staleness prediction signals*, we collected the set of RIPE Atlas traceroutes that passed through the same AS pair as one of the 1,711 diamonds but did *not* include a divergence IP address (i.e., did not include a *known* load balancer, although for simplicity we will refer to them as *non-load-balanced paths*). In total, our techniques detected *staleness prediction signals* for 7.1% of the non-load-balanced segments for the same time period, which we compare to the *staleness prediction signals* detected for the segments overlapping the 168 diamonds. Figure 9 compares the distribution of *staleness prediction signals* (i.e., number of changes predicted) per interdomain segment that are load-balanced (diamonds) versus segments that are not load-balanced. The graph shows that our techniques detect a similar number of *staleness prediction signals* for the two types of segments, with non-load-balanced paths predicted to change slightly more often. This result suggests that our techniques are able to account for the varying paths observed by traceroutes across a load balancer without falsely inferring many path changes. Figure 10 shows the distribution of the precision of these signals. Whereas signals for non-load-balanced segments have a median precision of 84%, signals for path segments with load balancers exhibit a median precision of 68%, suggesting that load balancers sometimes trick our techniques into falsely inferring a path has changed.

The precision of our traceroute-based techniques is 82% overall (Table 2), and this analysis suggests that one source of errors is interdomain load balancing, which is only known to occur in a small fraction of ASes (representing 19% of diamonds according to Diamond Miner's lenient definition of interdomain [77] that includes some diamonds that end at but do not cross a border, or 4.7% according to a more conservative definition [3]). Even for paths traversing load balancers, our precision remains high. In the future, we can investigate how to incorporate Diamond Miner results into our techniques, perhaps by not considering as an outlier changes that may just represent shifts across branches of a diamond.

6 EVALUATION OF USE CASES

Our techniques (i) improve a state-of-the-art measurement tool (§6.1); (ii) enable efficient (re)use of rate-limited traceroutes (§6.2); and (iii) improve a system that relies on a traceroute corpus (Appendix D).

6.1 Integration with DTRACK

We integrate our techniques into an extended version of DTRACK, DTRACK+SIGNALS, that checks for path changes whenever our techniques generate a signal. We match signals and route changes as described in §5.3. For traceroute-based signals, DTRACK+SIGNALS sends a detection probe to one of the hops inferred as changed by the signal; for BGP-based signals that identify the AS-border where the change occurred, DTRACK+SIGNALS sends a detection probe to the far end of the border link that triggered the signal; for BGP-based signals that identify a set of ASes that might have triggered the change, DTRACK+SIGNALS sends a detection probe to the last hop in each AS. If a probe confirms the change, a remap is triggered. These single-probe checks for path changes integrate seamlessly into DTRACK's change detection, allowing DTRACK+SIGNALS to spend one or a few probes to identify and ignore incorrect signals (false positives). Our signals help DTRACK+SIGNALS target paths that have likely changed, increasing probe utility and detecting more changes. Our signals require DTRACK+SIGNALS to invest probes into verifying changes; however, signal verification takes one or a few probes, and signal precision is high (Table 2), resulting in improved detection.

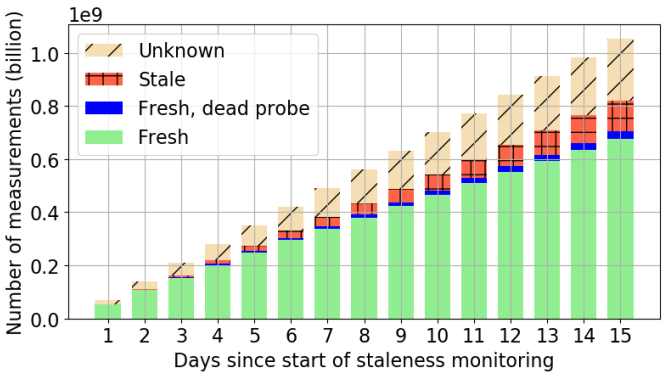


Figure 11: Number of *fresh* and *stale* RIPE Atlas traceroutes accumulated from 01–15 May 2020. Signals identify which traceroutes remain *fresh* and safe to use.

Figure 8 shows that DTRACK+SIGNALS performs similarly to signals alone when the probing budget is low, but it is not limited by coverage as the budget increases as the excess budget can be used by "vanilla" DTRACK's detection probes allocated according to its predictions of which paths are least stable. For example, at Ark's probing budget, DTRACK+SIGNALS detects 24% more border changes than DTRACK. As the probing budget increases, the relative benefit of signals and the difference between DTRACK and DTRACK+SIGNALS decreases. However, signals still provide gains even at high probing budgets as signals help DTRACK+SIGNALS detect when paths become unstable after long stability periods, a transition that is particularly hard to predict [19]. With a probing budget of 0.003 pps/path, vanilla DTRACK misses changes after 25% of true positive *staleness prediction signals* (not shown).

6.2 Reusability of Archival Traceroutes

In addition to keeping an individual system's corpus fresh, our staleness detection techniques can be applied to the huge corpus of publicly accessible traceroutes, enabling two related applications. First, by identifying traceroutes collected over a long time window that are still valid, these traceroutes can be used to generate a set of usable traceroutes vastly larger than the probing budget allocated to individual experiments. Second, they can be applied to requests for new traceroutes to identify those that can be safely served by an existing traceroute, giving requesters an option to reduce system load and preserve their budgets for other measurements.

This section evaluates those two use cases on the 1.15 billion public RIPE Atlas traceroutes issued in the first two weeks of May 2020. We classify a traceroute as *"stale"* if our techniques detect at least one *staleness prediction signal* after it is issued. If we detect no signals of change for a traceroute, we characterize it as *"fresh"* if our techniques and vantage points suffice to monitor every border-level interconnection, or *"unknown"* if they can monitor only a subset (or none). We use all public RIPE traceroutes for our techniques, to understand the full potential of reusing archived traceroutes.

Figure 11 shows that, at the end of our measurement period, RIPE Atlas has executed over 690 million traceroutes (60%) that are still fresh in terms of border-level IP hops and therefore reusable. As a point of comparison, RIPE Atlas limits a user to 1M credits per day at a cost of 10-30 credits per traceroute, so a user can issue up to 1.4 million traceroutes in two weeks. The set of available RIPE Atlas Probes changes over time, and 4% of the reusable traceroutes cannot be remeasured because the source Probe stopped being available (*"fresh, dead Probe"*). While proportionally small, the number of "fresh" traceroutes that can be safely used but not measured again is over 27 million. While stale traceroutes also increase over time, RIPE Atlas accumulates new traceroutes at a faster rate.

We next investigate the benefit that this high degree of reusability can have on preserving measurement budget and reducing system load. Of the 1.15 billion traceroutes, 985 million are user-defined measurements (UDMs) rather than RIPE's built-in measurements.[5] We consider a use case of needing the border-level path from (any Probe in) a particular ⟨AS, city⟩ to a particular destination prefix. We find that 90.3% of the UDMs (a median of 66 million traceroutes per day) can be satisfied by an already measured traceroute. However, such a drastic reduction of the number of public traceroutes will also affect our ability to detect stale traceroutes. To estimate the actual fraction of UDMs that can be avoided, we assume that a UDM traceroute that can be satisfied by an existing traceroute is not measured and exclude it from the traceroutes that our techniques use to generate *staleness prediction signals*. The reduction in available traceroutes also reduces coverage and requires more UDMs. Under these conditions we find that 68.6% of the UDMs can be avoided (median of 48.2 million measurements per day).

7 RELATED WORK

Section 2 discussed research which is most closely related to and motivates ours. This section summarizes other work.

Reducing number and cost of traceroute measurements. Given restricted budget constraints in measurement platforms, researchers have proposed techniques to reduce the number of traceroute measurement and the cost of individual measurements. Approaches include avoiding redundant TTL-limited probes to routers close to measurement sources and destinations [19, 22], pruning measurement sources and destinations without sacrificing network coverage [7, 71], or reducing the cost of route measurements [4, 78]. Our techniques are complementary to these approaches.

IGP/iBGP/eBGP dynamics. Teixeira et al. [73, 74] studied the impact of hot-potato changes on BGP dynamics, and found that up to 5% of the externally-visible BGP updates of a backbone AS were triggered by IGP events. These results have been confirmed by follow-up work on the same network by Wu et al. [80]. Similarly, Agarwal et al. [2] showed that local IGP engineering caused up to 25% of traffic with neighboring ASes to shift egress points. Park et al. [62] investigated the root causes of BGP update duplicates, and proposed a methodology to correlate duplicate updates with iBGP dynamics. In particular, they observed that 96% of duplicate eBGP updates were caused by an intradomain routing change which updated a non-transitive BGP attribute, such as Cluster-ID and MED values. Our work builds on these insights to detect IP-level changes that generate BGP activity without AS-path changes.

Correlating changes across measurements. Fazzion et al. [25] proposed techniques to verify if a change observed in a traceroute also impact traceroutes from that source to other destinations. Their techniques require active measurements to verify inferences.

Sibyl [18] is most similar to this paper. In addition to traceroute patching described and evaluated in §5.3, Sibyl also converts each traceroute to AS granularity, then monitors BGP route collectors for overlapping paths to the destination. If the BGP feed later reveals a change in the overlapping portion, but the first AS in the overlap remains in the new AS path, Sibyl infers that the traceroute is out of date. We build on this AS path monitoring (§4.1), and our other techniques significantly extend Sibyl's, allow detection of more changes (§5.3), and require no active measurements.

8 CONCLUSION

In recent years, the number of available traceroute vantage points has greatly increased, thanks to platforms such as RIPE Atlas. This increase enables expanded coverage of the Internet, but the vantage points are severely resource constrained and rate limited. These rate limits make it challenging for systems and studies that use large sets of traceroutes to reason about the Internet: accumulating measurements over a longer time period increases coverage at the expense of increased staleness, with path changes during the measurement interval affecting the results.

Our work overcomes this tradeoff by inferring which traceroutes in a corpus have gone stale due to path changes, allowing other traceroutes to be safely used for long periods of time and avoiding wasting measurements on unchanged paths. Our techniques use patterns in BGP updates as signals for changes not explicitly visible in the updates, and they monitor publicly available traceroutes to look for changes that overlap the corpus. Combined, they detect 81% of path changes. By *recycling* publicly available data, our techniques enable the safe *reuse* of traceroutes known to be unchanged and *reduce* the measurement budget needed to keep a corpus fresh.

Acknowledgements. This paper has been partially funded by a RIPE NCC Community Projects Fund, NSF grant 1836872, CAPES, CNPq, and FAPEMIG. We would like to thank the anonymous reviewers and our shepherd Cristel Pelsser for their insightful comments, and Kevin Vermeulen for aiding in the evaluation of the impact of interdomain load balancers on our techniques.

[5] We consider a measurement a UDM if it has ID greater than 1,000,000 and is not an Anchoring measurement [66].

A PROCESSING TRACEROUTES

IP-to-ASN mapping. We map IP-level hops to ASNs using longest prefix matching on BGP prefix advertisements received by the RouteViews and RIPE RIS collectors. If a prefix is missing in global routing tables, or in the case of Multi-Origin AS (MOAS) prefixes, we use prefix assignment and delegation data from RIR databases. We use traIXroute to correctly map IXP interfaces to ASNs [59]. When mapping traceroute IPs to ASes, we merge consecutive identical AS hops into one; if two hops that map to the same AS are separated by one or more unmapped hops, we also merge them together. We ignore all traceroutes whose AS mapping contains AS loops.

Mapping AS boundaries. To infer the AS boundaries in traceroutes, we use bdrmapIT if the source of the traceroute data is compatible with it [51]. Otherwise we use MAP-IT [52]. MAP-IT may not be able to infer AS interconnections that do not appear in an adequate number of traceroutes. If for an interconnection $\langle \text{IP}_{\text{AS}_x}, \text{IP}_{\text{AS}_y} \rangle$ MAP-IT did not pinpoint the border IP and bdrmapIT did not apply, we assume that both IPs are part of the border unless IP_{AS_y} belongs to an IXP, which we then consider as the border. We identify IP addresses that belong to the same router by performing alias resolution using MIDAR [47]. By combining the output of border mapping and alias resolution, we represent each traceroute at the granularity of a border router path.

Handling unresponsive traceroute hops. When comparing traceroutes, we want to avoid identifying measurement errors as route changes. For each unresponsive hop (⋆) with responsive hops on both sides, we check whether we only ever observe a single responsive hop matching that triple. If so, we patch the unresponsive hop. We treat any remaining unresponsive hops as wildcards that cannot indicate a change.

IP geolocation approach. We use three techniques to geolocate IP addresses. First, we use IPMap [13], which combines the results of multiple geolocation approaches. IPMap has been found to be 99.5% accurate for country-level geolocation [39] and 80.3% accurate at the city-level, better than commercial databases [23]. When IPMap is not able to geolocate an IP interface, we run a custom shortest-ping measurement, which we describe next. If that does not work, we rely on a city-level implementation of the CFS algorithm [32]. If an IP address cannot be geolocated with one of these techniques, we do not consider path segments starting or ending with that IP address as part of PoP-level staleness inference signals (§4.2.2).

Our implementation of shortest-ping geolocation follows an approach similar to IPMap. To derive candidate locations for a target IP ι, we first map ι to an AS, then look up the complete list of (facility, city) locations for that AS, if it exists in PeeringDB. If we can decode a location hint from a reverse DNS lookup of ι, we filter the list to only candidates that match the hint. We extract geolocation hints encoded in DNS names using CAIDA DDec [11], which combines DNS decoding rules from undns [71] and DRoP [36]. For each candidate (facility, city), we identify any available RIPE Atlas vantage point or Looking Glass server that is within 40km of that city and in an AS that either has a presence in that facility or is in the customer cone [49] of an AS that is in the facility. ASes in the same facility as ι's AS may interconnect there with ι, yielding the vantage point a short path to that AS. We then sort vantage points according to the following preferences. Vantage points in ASes with presence in the facilities are preferred over ASes that are only in the customer cones. Of those at the facilities, vantage points in ASes with known relationships are preferred over those in ASes without a known connection to AS_ι. The ones with known relationships are ordered according to their inter-AS relationship with AS_ι, following the same ordering as `Local Preference` values. Vantage points in ASes Vantage point ASes that have AS_ι as a customer are the most preferred, while vantage point ASes that use AS_ι as transit provider are the least preferred. We use this preference order to increase the chances of getting a direct path between the vantage point and ι. If multiple vantage points are equally preferred, we pick one at random. Starting from the most preferred vantage point, we issue three pings to ι. If the measured round-trip latency is at most 1ms (maximum distance of 100km based on the speed of light in fiber), we declare ι to be in the vantage point's city. Otherwise we repeat the same process from the next most preferred vantage point.

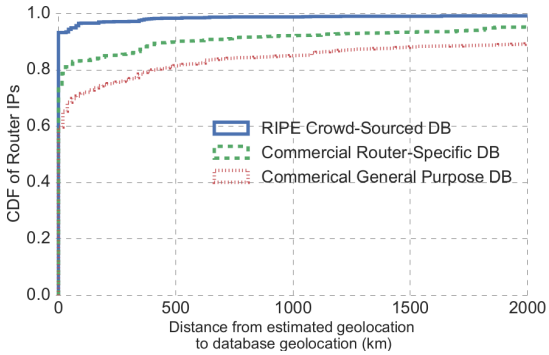


Figure 12: Validation of our geolocation technique by comparing to three databases.

IP geolocation validation. We validated our ping-based technique in 2017, on a set of traceroutes issued daily in May 2016 from 30 Ark monitors to the $x.1$ address in 360K /24 prefixes.[6] We discarded traceroutes that either contained IP-level cycles or did not reach the destination. For our analysis, we use the 72% of ⟨source, destination⟩ pairs for which we are left with a traceroute for every day of our measurement period.

Our ping-based technique was able to locate 82% of the border router IP addresses seen in these traceroutes, in each case finding a vantage point with a ping time of 1ms or less to the IP address. We could not locate 10% of IP addresses because they did not respond to pings, and 8% were responsive but we were unable to find a vantage point within 1ms. Our technique probed each IP address from an average of 8.3 vantage points, using 3 ping packets per vantage point, indicating that our PeeringDB-driven search enabled efficient identification of probing vantage points to realize a reasonable probing overhead.

To validate our ping-based technique, we compared its geolocation estimates of 14,720 IP addresses to three geolocation databases. The first is a crowd-sourced router geolocation dataset containing

[6] We used this dataset when we were developing our *staleness prediction signal* techniques but only use it as a source of IP addresses to validate our geolocation technique in this paper. For the rest of the paper, we run the technique we validate here on IP addresses from the new traceroute data described in the relevant sections.

approximately 35K submissions from network operators and collected by the RIPE OpenIP Map project [1]. We use a snapshot taken on 2017-06-05 and restrict our analysis to the 1319 IP addresses that contain latitude and longitude data and overlap with our data. Because this data is submitted by operators about networks they maintain, it should be highly accurate. To provide broader coverage, we also use two commercial geolocation databases, one a router-specific dataset from a well-known Internet performance firm and the other a popular, general purpose, dataset from a different vendor. Our data set had an overlap of 5,872 IP addresses with the router-specific database and had all IP addresses covered by the general purpose database.

Figure 12 depicts the difference in our geolocations and those of the databases for the set of common IP addresses. Our geolocation performs very well according to the crowd-sourced dataset, with 93% of our geolocations exactly matching, 96% < 100km and 98% < 500km. Compared with the locations from the router-specific database, our technique also performed well, with 75% of our geolocations exactly matching and 90% < 500km. The general purpose database shows the highest level of geolocation error, with only 60% matching exactly and 82% < 500km. In cases where our technique produces a different location than a database, it is hard to know which is correct. By the speed of light, a responsive address must be within 100km of a vantage point with a 1ms RTT, and so our technique can only have error greater than 100km if we have the wrong geolocation for a vantage point or if a different device responds to a ping on behalf of the border IP address (perhaps a middlebox or if the address is reused in different locations). Given the high similarity between our locations and those of the crowd-sourced database, which we consider to be the most definitive, we do not investigate the cause of this discrepancy further in this paper.

Use of active measurements. The services we use for IP alias identification and IP geolocation use active measurements. However, the measurement cost is low relative to the benefit we get by identifying corpus traceroutes that are unlikely to have changed and hence need not be reissued. First, geolocation and alias information changes on a much slower timescale than routes, and so the measurements do not need to be refreshed frequently. Second, we use existing services that provide the data and may already have issued the measurements. Third, the alias identification uses Ark, which is much less resource-constrained than platforms such as RIPE Atlas. Fourth, the measurements require only ping measurements, which incur significantly less probing overhead than traceroutes.

B EVALUATION OF TUNING PRECISION OF BGP COMMUNITY SIGNALS

While all our techniques are subject to calibration (§4.3.1), for most techniques, the purpose of calibration is to discover which corpus vantage points correlate with which public data sources. With communities, there is an additional challenge of also needing to learn which communities indicate where the route goes versus being unrelated to path changes entirely (not just unrelated to path changes in the corpus). This section demonstrates that our calibration can prune out unrelated communities, allowing a long-running system employing our methodology to achieve higher precision as time goes on. Figure 6a shows that precision of change identification starts at a reasonable 60%, but improves significantly over time as each technique gets calibrated. Here we characterize the automated learning of BGP communities related to path changes.

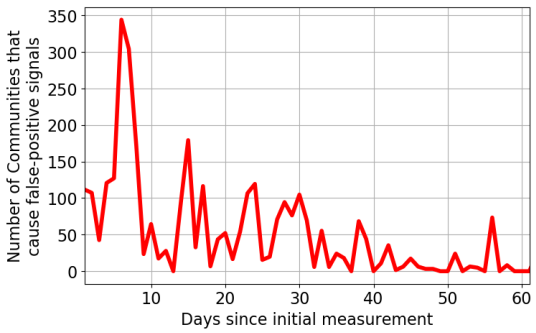


Figure 13: The number of BGP communities that generate false-positive signals decreases over time as our techniques identify and prune communities that are not related to path changes.

Figure 13 shows the number of communities that generate false-positive signals each day, i.e., the number of BGP communities that generated signals and had overlapping traceroutes that allowed us to determine that the community change did not indicate a path change. We see that over time the number of communities that generate false-positive signals decreases, as our techniques automatically prune communities that are not related to location or routing changes. We note that this approach may also be used to infer which BGP communities are used to signal geographical or topological information about routes.

C EXPLAINING HIGH COVERAGE

Figure 6b shows the coverage of our techniques remain high throughout the 60-day retrospective evaluation, with coverage being above 90% for paths public data can monitor. The high coverage is partially explained by overlapping paths between data sources and the traceroute corpus, as well as routing events impacting multiple paths and being visible on multiple traceroutes.

We find that 266 BGP VPs are located in 164 ASes that also host Atlas Probes, covering 720 (16%) of the Probes in P_{corpus} (§5.1.1). This large overlap follows from RIPE Atlas's large footprint and possibly from research-friendly networks hosting RIPE Atlas Probes and peering with BGP route collectors. This overlap provides good alignment between BGP feeds and public traceroutes (in our evaluation and in real systems/studies using RIPE Atlas). Moreover, due to the large number and distribution of RIPE Atlas Probes, the sets P_{public} and P_{corpus} of VPs have extensive overlap in terms of ASes and geographical locations. In particular, 2,931 (67%) of the Probes in P_{public} and 2,976 (67%) of the Probes in P_{corpus} are colocated in 658 ASes, while 1,925 Probes are located within 50km of another Probe from the other set. This setting is appropriate for evaluation because we intend our techniques to enable more efficient use (and re-use) of RIPE Atlas traceroutes, so the overlap in practice will be even higher (because we partition by source Probes and by destinations).

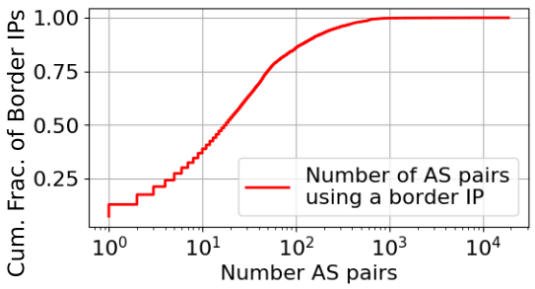


Figure 14: Distribution across border router IPs of the number of AS pairs that use the same border IP. This overlap allows us to correlate changes observed in one public traceroute to other traceroutes in the corpus.

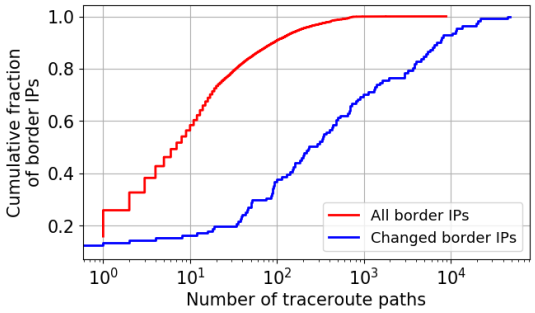


Figure 15: Distribution of the number of paths in our public traceroute feed for each border IP. Only 40% of all the border IPs are visible in 10 or more paths, but over 80% of the border IPs that are part of path changes are covered by at least 10 paths.

Finally, there is also significant overlap across public traceroutes and the traceroute corpus. Figure 14 shows the distribution, across all IP addresses at the border between two adjacent ASes in the traceroute corpus (*border IPs*), of the number of adjacent AS pairs that use that border IP. We observe border IPs are used by a large number of AS pairs; about 60% of the border IPs are used by more than 10 AS pairs, while 40% are used by more than 30 AS pairs. This is likely the case for border IPs assigned to routers at IXPs and colocation facilities. This diversity implies that our techniques can correlate measurements from different sources to identify changes. Figure 15 shows that border IPs involved in path changes tend to appear in more paths than border IPs that are not involved in any change. This property makes changes easier to observe, as routers with changes are covered in more paths, and helps achieve high coverage, as changes in these high-centrality routers impact (and trigger signals of changes for) a higher number of paths.

This result bodes well for real use of our approaches, as RIPE Atlas is both the largest set of public traceroutes to crawl for signals and an ideal use case for our techniques, as studies using it are severally constrained in their measurement budgets.

D INTEGRATION WITH IPLANE

We evaluate the impact of our techniques on iPlane [50], a service that predicts unmeasured routes by splicing segments of already measured traceroutes and that also serves as a prediction service for Sibyl [18]. To briefly explain how iPlane works, suppose iPlane needs to predict the path between a source s and a destination d. iPlane will search its corpus of measured traceroutes to find two traceroutes (s, d') and (s', d) that intersect at an intermediate PoP p, and will assume that the path (s, p, d) approximates the actual path between s and d. iPlane can benefit from accumulating a large number of archived traceroutes, since it is more likely to find traceroutes that can be spliced. On the other hand, staleness can lead to incorrect predictions, as paths that intersect in the corpus may no longer intersect in practice.

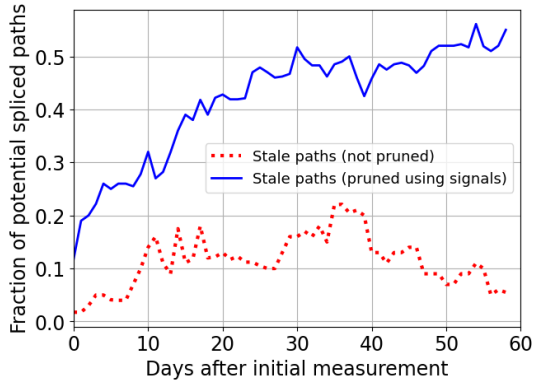
Traceroute corpus. To implement iPlane's methodology, we first construct the initial corpus by collecting one day (2019-03-01) of traceroutes from RIPE Atlas's anchoring measurements (as in §5.1) which results in 497,076 different paths. We use the anchoring measurements because they are repeated every 900 seconds between the same set of Atlas Probes and destinations (Atlas Anchors), allowing us to compute the staleness of the traceroute corpus over time.

For the initial corpus of traceroutes, we first group the IP addresses to PoPs, by mapping each address to an ⟨AS, city⟩ tuple using RIPE's IPMap for geolocation [23]. If we cannot geolocate an IP address, we consider it as its own PoP. In total, we were able to geolocate 31% of the 144,220 public IPs in our dataset. We then maintain the paths that go through each PoP, and identify spliced paths of the form (s, p, d) where a traceroute from s to some destination intersects at p with a traceroute from some source to d.

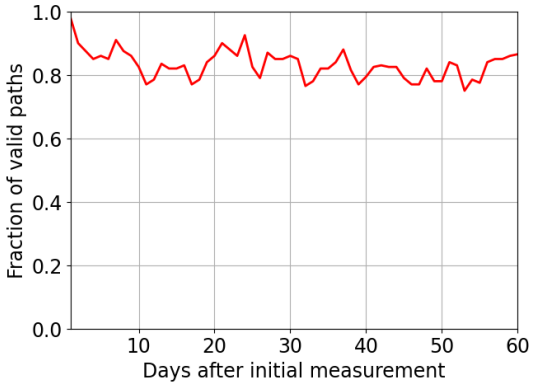
Tracking changes. To understand the impact of staleness in iPlane's ability to correctly splice traceroutes, we maintain two parallel versions of iPlane and its traceroute corpus. One retains the initial set of anchoring measurements. For the other, we remove traceroutes that our staleness signals suggest are out-of-date from iPlane's corpus, and we re-add them if our signals later suggest they are valid again (i.e., changed back to the original route) (§4.3.2).

Results. For each day in the evaluation period, we apply iPlane's methodology to generate spliced paths from each RIPE Atlas Probe to every Anchor that the Probe did not issue traceroutes to in the anchoring measurements. For each day, Figure 16a depicts what fraction of spliced paths are invalid because one or more of the paths has changed, causing the paths to no longer intersect. If iPlane simply relies on its initial corpus, over half of its spliced paths are invalid by the end of the measurement period. With our techniques, the fraction stale rarely rises above 20%, and it is below 10% at the end of the two month period. Importantly, Figure 16b shows that we are still able to retain the vast majority of valid spliced paths when pruning traceroutes flagged by our signals.

This result has two implications. First, iPlane can rely on our techniques to help prune out stale traceroutes. Second, our techniques let iPlane rely even on resource constrained vantage points such as RIPE Atlas, because our techniques lend confidence as to which traceroutes can be kept around for long periods of time. While our evaluation is over a small set of destinations, this ability



(a) Out of iPlane**'s initial spliced paths, fraction that are incorrect (stale) without using staleness prediction (not pruned) and when our staleness inference techniques are used (pruned).**



(b) Fraction of valid spliced paths iPlane **retained (not rejected as stale) when using our signals.**

Figure 16: Evaluation of benefit of using our *staleness prediction signal* techniques to prune stale traceroute from iPlane **corpus. Our techniques limit the fraction of staled paths over time (Figure 16a) without pruning out a significant fraction of valid paths (Figure 16b).**

to identify traceroutes that remain stable is very helpful for a system like iPlane that wants to consider the full set of hundreds of thousands of routable Internet prefixes.

REFERENCES

[1] Emile Aben. 2014. OpenIPMap Update. (2014).

[2] Sharad Agarwal, Antonio Nucci, and Supratik Bhattacharyya. 2005. Measuring the shared fate of IGP engineering and interdomain traffic. In *IEEE International Conference on Network Protocols*. IEEE, Boston, MA, USA.

[3] Rafael L. C. Almeida. 2019. *Classification of Load Balancing in the Internet*. Master's thesis. UFMG.

[4] Rafael L. C. Almeida, Italo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. 2020. Classification of Load Balancing in the Internet. In *INFOCOM*. IEEE.

[5] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. 2001. Resilient Overlay Networks. In *Proceedings on the Symposium on Operating System Principles*. ACM, Banff, Alberta, Canada.

[6] Brice Augustin, Timur Friedman, and Renata Teixeira. 2011. Measuring multipath routing in the Internet. *IEEE/ACM Transactions on Networking* 19 (2011).

[7] Robert Beverly, Arthur Berger, and Geoffrey G. Xie. 2010. Primitives for Active Internet Topology Mapping: Toward High-frequency Characterization. In *Proceedings of the 2010 Internet Measurement Conference*. ACM, Melbourne, Australia.

[8] Mohammad Braei and Sebastian Wagner. 2020. Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. *arXiv preprint arXiv:2004.00433* (2020).

[9] Randy Bush, Olaf Maennel, Matthew Roughan, and Steve Uhlig. 2009. Internet Optometry: Assessing the Broken Glasses in Internet Reachability. In *Proceedings of the 2009 Internet Measurement Conference*. ACM, Chicago, Illinois, USA.

[10] Matthew Caesar, L Subramanian, and Randy H Katz. 2003. Root cause analysis of BGP dynamics. In *Proceedings of the 2003 Internet Measurement Conference*. ACM, Miami, FL, USA.

[11] CAIDA. 2014. DNS Decoding daatabase. https://ddec.caida.org. (2014).

[12] CAIDA. 2019. Archipelago (Ark) Measurement Infrastructure. http://www.caida.org/projects/ark/. (2019).

[13] Massimo Candela. 2019. RIPE IPmap - What's Under the Hood? https://labs.ripe.net/Members/massimo_candela/ripe-ipmap-whats-under-the-hood. (January 2019).

[14] Jin Cao, Drew Davis, Scott Vander Wiel, and Bin Yu. 2000. Time-Varying Network Tomography: Router Link Data. *J. Amer. Statist. Assoc.* 95 (2000).

[15] Fangfei Chen, Ramesh K Sitaraman, and Marcelo Torres. 2015. End-user mapping: Next generation request routing for content delivery. In *Proceedings of the 2015 ACM SIGCOMM Conference*. ACM, London, UK.

[16] K. Chen, D. R. Choffnes, R. Potharaju, Y. Chen, F. E. Bustamante, D. Pei, and Y. Zhao. 2014. Where the Sidewalk Ends: Extending the Internet AS Graph Using Traceroutes from P2P Users. *IEEE/ACM Transactions on Computers* 63 (2014).

[17] Kokyo Choy. 2001. Outlier detection for stationary time series. *Journal of Statistical Planning and Inference* 99 (2001).

[18] Ítalo Cunha, Pietro Marchetta, Matt Calder, Yi-Ching Chiu, Bruno VA Machado, Antonio Pescapè, Vasileios Giotsas, Harsha V Madhyastha, and Ethan Katz-Bassett. 2016. Sibyl: a practical Internet route oracle. In *Symposium on Networked Systems Design and Implementation*. USENIX, Boston, MA, USA.

[19] Italo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. 2011. Predicting and Tracking Internet Path Changes. In *Proceedings of the 2011 ACM SIGCOMM Conference*. ACM, Toronto, Ontario, Canada.

[20] Amogh Dhamdhere, David D Clark, Alexander Gamero-Garrido, Matthew Luckie, Ricky KP Mok, Gautam Akiwate, Kabir Gogia, Vaibhav Bajpai, Alex C Snoeren, and KC Claffy. 2018. Inferring persistent interdomain congestion. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, New York, NY, USA.

[21] Benoit Donnet. 2009. Incentives for BGP Guided IP-Level Topology Discovery. In *International Workshop on Traffic Monitoring and Analysis*. TMA, Zurich, Switzerland.

[22] Benoit Donnet, Philippe Raoult, Timur Friedman, and Mark Crovella. 2005. Efficient Algorithms for Large-scale Topology Discovery. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. ACM, Banff, Alberta, Canada.

[23] Ben Du, Massimo Candela, Bradley Huffaker, Alex C. Snoeren, and KC Claffy. 2020. RIPE IPmap active geoaddress: mechanism and performance evaluation. *ACM SIGCOMM Computer Communication Review* 50 (2020).

[24] Rodérick Fanou, Pierre Francois, and Emile Aben. 2015. On the Diversity of Interdomain Routing in Africa. In *International Conference on Passive and Active Network Measurement*. PAM, New York, NY, USA.

[25] Elverton Fazzion, Ítalo Cunha, Dorgival Guedes, Wagner Meira Jr., Renata Teixeira, Darryl Veitch, and Christophe Diot. 2016. Efficient Remapping of Internet Routing Events. In *ACM SIGCOMM Posters and Demos*. ACM, Florianopolis, Brazil.

[26] Nick Feamster, David G Andersen, Hari Balakrishnan, and M Frans Kaashoek. 2003. Measuring the Effects of Internet Path Faults on Reactive Routing. *ACM SIGMETRICS Performance Evaluation Review* 31 (2003).

[27] Anja Feldmann, Olaf Maennel, Z Morley Mao, Arthur Berger, and Bruce Maggs. 2004. Locating Internet Routing Instabilities. In *Proceedings of the 2004 ACM SIGCOMM Conference*. ACM, Austin, TX, USA.

[28] T. Flach, E. Katz-Bassett, and R. Govindan. 2012. Quantifying Violations of Destination-based Forwarding on the Internet. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, Boston, MA, USA.

[29] Romain Fontugne, Cristel Pelsser, Emile Aben, and Randy Bush. 2017. Pinpointing Delay and Forwarding Anomalies using Large-Scale Traceroute Measurements. In *Proceedings of the 2017 Internet Measurement Conference*. ACM, London, United Kingdom.

[30] D. Ghita, C. Karakus, K. Argyraki, and P. Thiran. 2011. Shifting Network Tomography Toward a Practical Goal. In *CoNEXT*. ACM, Tokyo, Japan.

[31] Vasileios Giotsas, Georgios Smaragdakis, Christoph Dietzel, Philipp Richter, Anja Feldmann, and Arthur Berger. 2017. Inferring BGP Blackholing Activity in the Internet. In *Proceedings of the 2017 Internet Measurement Conference*. ACM, London, United Kingdom.

[32] Vasileios Giotsas, Georgios Smaragdakis, Bradley Huffaker, Matthew Luckie, and KC Claffy. 2015. Mapping peering interconnections to a facility. In *CoNEXT*. ACM, Heidelberg, Germany.

[33] G. Gürsun, N. Ruchansky, E. Terzi, and M. Crovella. 2012. Routing State Distance: A Path-based Metric for Network Analysis. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, Boston, MA, USA.

[34] David Hauweele, Bruno Quoitin, Cristel Pelsser, and Randy Bush. 2018. What do parrots and BGP routers have in common?. In *Proceedings of the 2018 ACM SIGCOMM Conference*. ACM, Budapest, Hungary.

[35] Thomas Holterbach, Cristel Pelsser, Randy Bush, and Laurent Vanbever. 2015. Quantifying interference between measurements on the RIPE Atlas platform. In *Proceedings of the 2015 Internet Measurement Conference*. ACM, Tokyo, Japan.

[36] Bradley Huffaker, Marina Fomenkov, and KC Claffy. 2014. DRoP: DNS-based router positioning. *ACM SIGCOMM Computer Communication Review* 44 (2014).

[37] B Iglewicz and D Hoaglin. 1993. The ASQC basic references in quality control: statistical techniques. *How to detect and handle outliers* 16 (1993).

[38] Mattia Iodice, Massimo Candela, and Giuseppe Di Battista. 2019. Periodic Path Changes in RIPE Atlas. *IEEE Access* 7 (2019).

[39] Costas Iordanou, Georgios Smaragdakis, Ingmar Poese, and Nikolaos Laoutaris. 2018. Tracing cross border web tracking. In *Proceedings of the 2018 Internet Measurement Conference*. ACM, Boston, MA, USA.

[40] Quentin Jacquemart, Guillaume Urvoy-Keller, and Ernst Biersack. 2016. Behind IP prefix overlaps in the BGP routing table. In *International Conference on Passive and Active Network Measurement*. PAM, Heraklion, Crete, Greece.

[41] Umar Javed, Italo Cunha, David Choffnes, Ethan Katz-Bassett, Thomas Anderson, and Arvind Krishnamurthy. 2013. PoiRoot: Investigating the root cause of interdomain path changes. In *Proceedings of the 2013 ACM SIGCOMM Conference*. ACM, Hong Kong.

[42] Xing Jin, Wanqing Tu, and S-HG Chan. 2008. Traceroute-based topology inference without network coordinate estimation. In *International Conference on Communications*. IEEE, Beijing, China.

[43] Min Suk Kang and Virgil D Gligor. 2014. Routing bottlenecks in the Internet: Causes, exploits, and countermeasures. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Scottsdale, AZ, USA.

[44] Ethan Katz-Bassett, Harsha V Madhyastha, Vijay Kumar Adhikari, Colin Scott, Justine Sherry, Peter Van Wesep, Thomas E Anderson, and Arvind Krishnamurthy. 2010. Reverse traceroute. In *Symposium on Networked Systems Design and Implementation*. USENIX, Boston, MA, USA.

[45] Ethan Katz-Bassett, Harsha V Madhyastha, John P John, Arvind Krishnamurthy, David Wetherall, and Thomas E Anderson. 2008. Studying Black Holes in the Internet with Hubble. In *Symposium on Networked Systems Design and Implementation*. USENIX, San Francisco, CA, USA.

[46] E. Katz-Bassett, C. Scott, D. R. Choffnes, Í. Cunha, V. Valancius, N. Feamster, H. V. Madhyastha, T. Anderson, and A. Krishnamurthy. 2012. LIFEGUARD: Practical Repair of Persistent Route Failures. In *Proceedings of the 2012 ACM SIGCOMM Conference*. ACM, Helsinki, Finland.

[47] K. Keys, Y. Hyun, M. Luckie, and K. Claffy. 2013. Internet-scale IPv4 Alias Resolution with MIDAR. *IEEE/ACM Transactions on Networking* 21 (2013).

[48] Leiwen Deng and A. Kuzmanovic. 2008. Monitoring persistently congested Internet links. In *2008 IEEE International Conference on Network Protocols*. IEEE, Orlando, FL, USA.

[49] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and others. 2013. AS relationships, customer cones, and validation. In *Proceedings of the 2013 Internet Measurement Conference*. ACM, Barcelona, Spain.

[50] 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 *Proceedings of the Symposium on Operating Systems Design and Implementation*. USENIX, Berkely, CA, USA.

[51] Alexander Marder, Matthew Luckie, Amogh Dhamdhere, Bradley Huffaker, KC Claffy, and Jonathan M. Smith. 2018. Pushing the Boundaries with bdrmapIT: Mapping Router Ownership at Internet Scale. In *Proceedings of the 2018 Internet Measurement Conference*. ACM, Boston, MA, USA.

[52] Alexander Marder and Jonathan M Smith. 2016. MAP-IT: Multipass accurate passive inferences from traceroute. In *Proceedings of the 2016 Internet Measurement Conference*. ACM, Santa Monica, CA, USA.

[53] Richard McCleary, Richard A Hay, Erroll E Meidinger, and David McDowall. 1980. *Applied time series analysis for the social sciences*. Sage Publications, Beverly Hills, CA, USA.

[54] Measurement-Lab. 2019. Paris traceroute dataset. https://www.measurementlab.net/tests/paris_traceroute/. (2019).

[55] Ricky KP Mok, Vaibhav Bajpai, Amogh Dhamdhere, and KC Claffy. 2018. Revealing the Load-Balancing Behavior of YouTube Traffic on Interdomain Links. In *International Conference on Passive and Active Network Measurement*. PAM, Berlin, Germany.

[56] R. Motamedi, B. Yeganeh, B. Chandrasekaran, R. Rejaie, B. M. Maggs, and W. Willinger. 2019. On Mapping the Interconnections in Today's Internet. *IEEE/ACM Transactions on Networking* 27 (2019).

[57] H. Nguyen, R. Teixeira, P. Thiran, and C. Diot. 2009. Minimizing Probing Cost for Detecting Interface Failures: Algorithms and Scalability Analysis. In *INFOCOM*. IEEE, Rio de Janeiro, Brazil.

[58] Init7 NOC. 2019. BGP Communities for Init7 Customers. https://as13030.net/static/pdf/as13030_bgp_communities.pdf. (2019).

[59] George Nomikos and Xenofontas Dimitropoulos. 2016. traIXroute: Detecting IXPs in traceroute paths. In *International Conference on Passive and Active Network Measurement*. PAM, Heraklion, Crete, Greece.

[60] George Nomikos, Vasileios Kotronis, Pavlos Sermpezis, Petros Gigis, Lefteris Manassakis, Christoph Dietzel, Stavros Konstantaras, Xenofontas Dimitropoulos, and Vasileios Giotsas. 2018. O Peer, Where Art Thou? Uncovering Remote Peering Interconnections at IXPs. In *Proceedings of the 2018 Internet Measurement Conference*. ACM, Boston, MA, USA.

[61] Chiara Orsini, Alistair King, Danilo Giordano, Vasileios Giotsas, and Alberto Dainotti. 2016. BGPStream: a software framework for live and historical BGP data analysis. In *Proceedings of the 2016 Internet Measurement Conference*. ACM, Santa Monica, CA, USA.

[62] Jong Han Park, Dan Jen, Mohit Lad, Shane Amante, Danny McPherson, and Lixia Zhang. 2010. Investigating occurrence of duplicate updates in BGP announcements. In *International Conference on Passive and Active Network Measurement*. PAM, Zurich, Switzerland.

[63] PeeringDB. 2019. Route Server ASNs. https://www.peeringdb.com/advanced_search?info_type__in=Route+Server&reftag=net. (2019).

[64] Philipp Richter, Georgios Smaragdakis, Anja Feldmann, Nikolaos Chatzis, Jan Boettger, and Walter Willinger. 2014. Peering at peerings: On the role of IXP route servers. In *Proceedings of the 2014 Internet Measurement Conference*. ACM, Vancouver, BC, Canada.

[65] RIPE NCC. 2019. How much bandwidth will the probe consume? https://atlas.ripe.net/about/faq/. (2019).

[66] RIPE NCC. 2019. ID Space Layout for Built-In Measurements. https://atlas.ripe.net/docs/built-in/. (2019).

[67] RIPE NCC. 2019. RIPE Atlas. https://atlas.ripe.net. (2019).

[68] Rachee Singh and Phillipa Gill. 2016. PathCache: A Path Prediction Toolkit. In *ACM SIGCOMM Posters and Demos*. ACM, Florianopolis, Brazil.

[69] SpeedChecker Ltd. 2019. SpeedChecker. https://www.speedchecker.com/. (2019).

[70] Neil Spring, Ratul Mahajan, and Thomas Anderson. 2003. The causes of path inflation. In *Proceedings of the 2003 ACM SIGCOMM Conference*. ACM, Karlsruhe, Germany.

[71] Neil Spring, Ratul Mahajan, and David Wetherall. 2002. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions On Networking* 12 (2002).

[72] Claudia Tebaldi and Mike West. 1998. Bayesian Inference on Network Traffic Using Link Count Data. *J. Amer. Statist. Assoc.* 93 (1998).

[73] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. 2004. Dynamics of hot-potato routing in IP networks. *ACM SIGMETRICS Performance Evaluation Review* 32 (2004).

[74] Renata Teixeira, Aman Shaikh, Timothy G Griffin, and Jennifer Rexford. 2008. Impact of hot-potato routing changes in IP networks. *IEEE/ACM Transactions On Networking* 16 (2008).

[75] Ruey S Tsay. 1988. Outliers, level shifts, and variance changes in time series. *Journal of forecasting* 7 (1988).

[76] Yehuda Vardi. 2004. Metrics useful in network tomography studies. *IEEE Signal Processing Letters* 11 (2004).

[77] Kevin Vermeulen, Justin P Rohrer, Robert Beverly, Olivier Fourmaux, and Timur Friedman. 2020. Diamond-Miner: Comprehensive Discovery of the Internet's Topology Diamonds. In *Symposium on Networked Systems Design and Implementation*. USENIX, Santa Clara, CA, USA.

[78] K. Vermeulen, S. D. Strowes, O. Fourmaux, and T. Friedman. 2018. Multilevel MDA-Lite Paris Traceroute. In *Proceedings of the 2018 Internet Measurement Conference*. ACM, Boston, MA, USA.

[79] Li Wei, Nitin Kumar, Venkata Nishanth Lolla, Eamonn J Keogh, Stefano Lonardi, and Chotirat (Ann) Ratanamahatana. 2005. Assumption-Free Anomaly Detection in Time Series.. In *SSDBM*. SSDBM, Santa Barbara, CA, USA.

[80] Jian Wu, Zhuoqing Morley Mao, Jennifer Rexford, and Jia Wang. 2005. Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network. In *Symposium on Networked Systems Design and Implementation*. USENIX, Berkeley, CA, USA.

[81] Y Zhang, N Duffield, V Paxson, and S Shenker. 2001. On the consistency of Internet path properties. In *ACM SIGCOMM IMW*. ACM, San Fransisco, CA, USA.

[82] Zheng Zhang, Ming Zhang, Albert Greenberg, Y. Charlie Hu, Ratul Mahajan, and Blaine Christian. 2010. Optimizing Cost and Performance in Online Service Provider Networks. In *Symposium on Networked Systems Design and Implementation*. USENIX, San Jose, CA, USA.

[83] Zheng Zhang, Ying Zhang, Y Charlie Hu, Z Morley Mao, and Randy Bush. 2008. iSPY: detecting IP prefix hijacking on my own. In *Proceedings of the 2008 ACM SIGCOMM Conference*, Vol. 38. ACM, Seattle, WA, USA.