Pushing the Boundaries with bdrmapIT: Mapping Router Ownership at Internet Scale

Alexander Marder
University of Pennsylvania
amarder@seas.upenn.edu

Matthew Luckie
University of Waikato
mjl@wand.net.nz

Amogh Dhamdhere
CAIDA / UC San Diego
amogh@caida.org

Bradley Huffaker
CAIDA / UC San Diego
bradley@caida.org

kc claffy
CAIDA / UC San Diego
kc@caida.org

Jonathan M. Smith
University of Pennsylvania
jms@seas.upenn.edu

ABSTRACT

Two complementary approaches to mapping network boundaries from traceroute paths recently emerged [27,31]. Both approaches apply heuristics to inform inferences extracted from traceroute measurement campaigns. *bdrmap* [27] used targeted traceroutes from a specific network, alias resolution probing techniques, and AS relationship inferences, to infer the boundaries of that specific network and the other networks attached at each boundary. *MAPIT* [31] tackled the ambitious challenge of inferring *all AS-level network boundaries* in a massive archived collection of traceroutes launched from many different networks. Both were substantial contributions to the state-of-the-art, and inspired a collaboration to explore the potential to combine the approaches. We present and evaluate `bdrmapIT`, the result of that exploration, which yielded a more complete, accurate, and general solution to this persistent and central challenge of Internet topology research. `bdrmapIT` achieves 91.8%-98.8% accuracy when mapping AS boundaries in two Internet-wide traceroute datasets, vastly improving on `MAP-IT`'s coverage without sacrificing `bdrmap`'s ability to map a single network. The `bdrmapIT` source code is available at https://git.io/fAsI0.

ACM Reference Format:
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 *2018 Internet Measurement Conference (IMC '18), October 31-November 2, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3278532.3278538

1 INTRODUCTION

A long-standing challenge of Internet topology research is router-level topology discovery and ownership inference, which relies on IP-level measurements that trick routers into revealing network structure (e.g., traceroute), and heuristics to interpret such measurements. The challenge is most daunting in between autonomously managed networks. The task of mapping the borders between networks at the router level is equivalent to the task of identifying routers in traceroute measurements, and then inferring who owns those routers.

There are fundamental architectural constraints that hinder router ownership inference: the TCP/IP architecture has no notion of interdomain boundaries at the network layer, nor any notion of boundaries around a single router. That is, at the IP layer, there is no unique router identifier. Router software diversity further complicates inference: a given router may respond to traceroute with a source address obtained from the interface it received the probe packet on, the interface it sends the reply on, or some other interface. Due to network addressing practices, the IP addresses on these interfaces may not even belong to the owner of the router. When constructing an Internet-scale topology, superimposing measurements from multiple vantage points (VPs) can mitigate some of these risks, but results in topologies where links farther from VPs are less likely to be observed [24], and those that are observed have fewer constraints to use in ownership inference. Finally, validating router-level inferences against ground truth requires tedious cooperation from operators who have limited incentive or time.

Techniques to accurately map network borders were elusive until 2016, when two independently conceived approaches [27, 31] achieved accuracy significantly superior to the then state-of-the-art. Both `MAP-IT` and `bdrmap` use heuristics to minimize well-known errors in interpreting traceroute data, but `bdrmap` developed specialized heuristics to analyze a router-level graph it inferred from one vantage point, while `MAP-IT` used an iterative graph-refinement process on an interface-level graph previously gathered from many vantage points. The different goals, design choices, and assumptions inspired us to ask: can we leverage the lessons from both efforts to create a more general-purpose solution? We report the results of that effort here.

A general solution to the border mapping and router ownership inference problem, operating at Internet scale, will accelerate progress in a number of research and operational pursuits. For example, CAIDA has used `bdrmap` for three years to study interdomain congestion, but has restricted itself to links involving the measurement VP's network; a generalized border mapping tool could amplify visibility to a much broader set of interdomain links. Other congestion inference [37] and resilience assessment [14,25,33,36] research could be extended to identify networks and links experiencing congestion. A new border mapping tool could address well-known pitfalls with less rigorous approaches to identifying interdomain links [32,38]. `MAP-IT` was already instrumental in uncovering bugs in traceroute implementations [34]—investigation of

IMC '18, October 31-November 2, 2018, Boston, MA, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5619-0/18/10.
https://doi.org/10.1145/3278532.3278538

anomalous MAP-IT inferences revealed that the M-lab traceroutes used as input were corrupted.

We make the following three contributions:

(1) We developed and implemented bdrmapIT, which uses the sophisticated bdrmap heuristics as additional input to the MAP-IT boundary location algorithm. In §2 we discuss how we leverage the strengths of bdrmap and MAP-IT, §3 gives an overview of the synthesized approach, and §4, §5, and §6 detail the algorithm.

(2) We demonstrated the superior accuracy and coverage of bdrmapIT over either previous approach. We validated bdrmapIT against ground truth from a tier 1, a large access, and two R&E networks, achieving 91.8%-98.8% accuracy, despite not using traceroute VPs in any of the validating networks. We also demonstrated that bdrmapIT's accuracy is independent of the number of vantage points: the performance is equivalent when we decrease the number of VPs from 80 to 20 (§7).

(3) We release our implementation and source code to promote reproducibility, and so that others can use our tool for their own analyses. We have incorporated bdrmapIT into CAIDA's ITDK [6] generation process.

§	Step	bdrmap	MAP-IT
4	Phase 1: Construct the Graph		
4.1	Label AS-level Metadata	X	X
4.2	Assign Link Confidence Labels	X	
4.3	Create Origin AS Sets		
4.4	Identify Destination ASes	X	
5	Phase 2: Annotate Last Hops		
5.1	When Dest. AS set is empty	X	
5.2	When Dest. AS set is not empty	X	
6	Phase 3: Graph Refinement		
6.1	Annotate IRs	X	X
6.1.1	Apply Link Vote Heuristics	X	
6.1.2	Correct Reallocated Prefixes		
6.1.3	Check for Exceptions		
6.1.4	Apply IR Vote Heuristics		X
6.1.5	Check for a Hidden AS	X	
6.2	Annotate Interfaces		X
6.3	Refine the Graph		X

Table 1: bdrmapIT heuristics adapted from bdrmap and MAP-IT.

2 RELATED WORK

The canonical approach to convert IP-level traceroute output to an AS-level path uses the origin AS announcing the longest matching prefix into the global BGP routing system. The risk of this approach is that some routers respond to traceroute probes with a source IP address belonging to a different network. In 2010, Zhang reported that between 16% and 47% of AS adjacencies inferred using the canonical longest prefix match approach were likely false [40]. In 2003, Mao's "AS traceroute" [30] used correlated BGP and traceroute views from the same VP, DNS names, and WHOIS data to perform IP-AS mappings, later improving them further using dynamic programming, although only for a /24 address granularity [29]. Generally, interdomain links use /30 or /31 prefixes to use address space efficiently, and co-located BGP and traceroute views are rare. In 2009, Chen *et al.* proposed a set of heuristics to distill some missing AS-level links from traceroute data [16]. In 2010, Huffaker *et al.* developed and validated four different router ownership heuristics using IPv4 alias resolution, inferred AS relationships, and degree [18] separately from each other; their best-performing heuristic was correct 71% of the time.

In 2016, two distinct approaches towards inferring router ownership were proposed independently: bdrmap [27], and MAP-IT [31]. bdrmap focuses on identifying all interdomain links observable by a single VP in a hosting AS, and consists of data collection and router ownership inference components. The data collection component conducts traceroutes from the VP towards every prefix routed in the Internet. The data collection phase is reactive, using alias resolution to infer which interfaces returned by traceroute belong to the same routers, and additional traceroutes to different addresses within a single prefix if a prior traceroute might have found an off-path interface within the target AS.

The inference component of bdrmap uses the collected data to infer router ownership within the hosting AS and adjacent ASes. Starting at the VP, bdrmap performs a breadth-first search based on hop-count from the VP of the traceroute responses to identify routers internal to the VP network, defined as all routers that appear prior to an interface address announced by the VP network in the traceroutes. Subsequent routers are either operated by the VP network, or by a directly connected neighbor. bdrmap uses heuristics to infer ownership of subsequent routers until all routers immediately subsequent to the network boundary have been mapped to other ASes.

bdrmap heuristics correctly infer router ownership when an edge network operator drops traceroute probes at their border router, when routers reply using unrouted IP addresses, when routers respond with an off-path address announced by a third party in BGP, and uses AS relationships to reason about ownership when traceroute returns IP paths that are not congruent with BGP policy.

MAP-IT consists only of an inference component, to identify interdomain links (between ISP networks) in the Internet core. In contrast to bdrmap, MAP-IT aggregates all available traceroute data collected by many VPs in many ASes, but does not use any alias resolution to infer routers. Instead, MAP-IT employs localized reasoning on an interface-level graph, drawing inferences from each interface and its neighbors in isolation. MAP-IT iterates over the set of interfaces several times, in each iteration identifying the interfaces used for interdomain links. The primary inference method is to find an interface with an address originated by one AS, where a plurality of either its subsequent or preceding interfaces map to another AS, indicating a link between the two networks. After each iteration, MAP-IT refines the graph, enabling more accurate interdomain inferences in subsequent iterations until an iteration yields no changes.

Our new technique, implemented in the bdrmapIT tool, leverages the strengths of bdrmap and MAP-IT for use in a general-purpose solution. Specifically: (1) bdrmap infers AS owners only for routers at the first AS boundary and requires a VP in each network of interest, and (2) MAP-IT lacks heuristics for edge networks and low-visibility links, such as routers without subsequent hops in

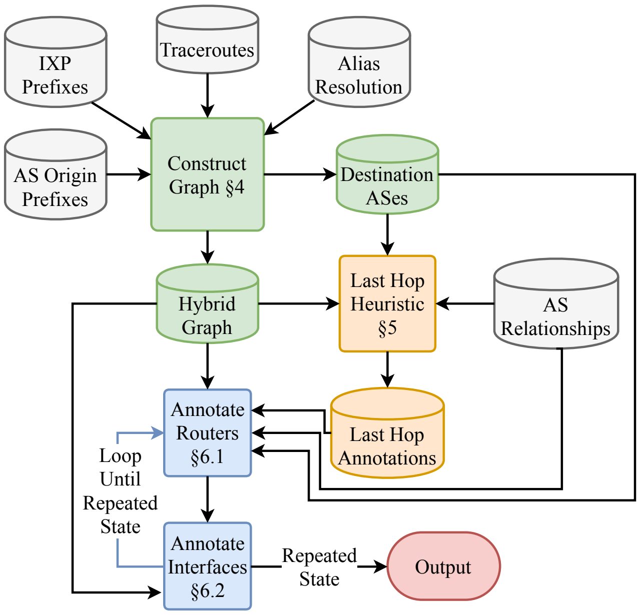


Figure 1: bdrmapIT's three phases: Constructing the Graph, Annotating Last Hops, and Annotating IRs and Interfaces.

traceroute due to firewalls, and does not use router alias inferences. To create bdrmapIT, we adapted bdrmap heuristics to the MAP-IT graph refinement framework with localized reasoning, yielding a technique capable of inferring the owner of routers visible from any number of VPs, in any number of networks. When restricted to the input for bdrmap or MAP-IT, bdrmapIT is at least as accurate as the prior techniques. Table 1 gives an overview of bdrmapIT and the genesis of each heuristic.

We faced technical challenges adapting heuristics into bdrmapIT. Adapting bdrmap heuristics required removing assumptions made when mapping a single VP network's routers to accommodate MAP-IT's local reasoning. First, router ownership inferences become harder with more AS-level diversity around the router, meaning that mapping router IP interface addresses to the AS who owns the router requires more sophisticated heuristics. To overcome this difficulty, bdrmapIT constructs sets of candidate origin ASes (§4.3) and iteratively narrows the set. Similarly, bdrmap detects third-party addresses when it sees an unexpected AS in between the VP AS and an adjacent network. Other traceroute datasets may expose third party addresses several AS hops removed from the VP. Our adapted third-party heuristic uses origin AS sets, destination AS sets (§4.4), and router operator inferences (§6.1); this technique to identify third-party addresses improves with the graph refinement (§6) process adapted from MAP-IT. Finally, adapting MAP-IT's graph refinement heuristics to router graphs instead of interface graphs was a significant challenge. Accommodating alias resolution input data motivated our strategy laid out in §4.2 and §6.1, where we infer router operators at the router granularity, without first selecting an AS for each alias.

Term	Definition
– Graph –	
IR:	Inferred Router
interface:	interface used on the IR
subsequent interface:	interface that follows an IR interface in a traceroute
link:	inferred connection from IR to subsequent interface
– Origin AS –	
interface origin AS:	origin AS of interface's IP address
IR origin AS set:	union of IR interfaces' origin AS sets
$L_{(IR_i,j)}$ **(link origin AS set):**	set of origin ASes observed immediately previous to the link in a traceroute
D_{IR} **(destination set):**	set of origin ASes of destination IPs of paths crossing IR
AS annotation:	AS of inferred operator of IR or interface

Table 2: Glossary of terms

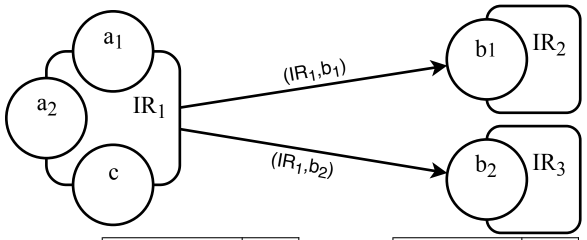
3 OVERVIEW

bdrmapIT has three phases, illustrated in Fig. 1. The first phase builds a directed graph from the traceroutes and alias resolution (§4). The second phase infers the operators of routers that appear only at the end of traceroutes (§5). These mappings are not subject to refinement, and provide topological context for mappings in the final phase. The final phase maps routers observed in the middle of at least one traceroute path to ASes (§6). The last phase is iterative, visiting routers and interfaces multiple times to make accurate inferences.

3.1 Constructing Interface Graph (§4)

bdrmapIT creates an inferred router (IR) graph by combining previously collected traceroutes with inferred IP router aliases data. Many datasets have few (or no) aliases resolved, and bdrmapIT will map AS borders without it, but aliases can improve mapping accuracy by providing additional router operator constraints and ensuring a consistent inference for interfaces used on the same router. *Links* connect IRs to interfaces seen subsequently in a traceroute (Figure 2). bdrmapIT works with existing datasets, which dictate the graph, without the opportunity for additional probing. To aid our analysis, we store significant graph metadata, e.g., for each interface we store its origin AS, which is the AS announcing the longest matching prefix for the IP address of that interface. We label links according to our confidence in their ability to inform accurate router ownership inference (§4). We also store the origin AS set for each link, which contains the origin ASes of all IR interfaces seen prior to the connected interface. We label IRs with their destination AS set (§5.2), which contains the destination ASes of the traceroutes in which any IR interface appears.

In addition to static metadata labels, we include dynamic *annotations* for every IR and interface, which bdrmapIT continually refines throughout the algorithm. bdrmapIT assigns annotations when inferring the operators of last hop IRs (§5), as well as during the graph refinement loop (§6). IR annotations indicate the AS operating the IR, while interface annotations represent the AS connected to the interface, i.e., the other side of the link (Fig. 3). When annotating IRs and interfaces we make two assumptions that are generally but not always true: that routers (IRs) are operated by a single AS; and



	hops	dst		hops	dst
path 1:	... a1 b1	b3	path 3:	... c b2	e
path 2:	... a2 b2	d			

Figure 2: Using paths 1-3, we build $IR_{1,2,3}$ (boxes) and the links between IRs and interfaces (circles).

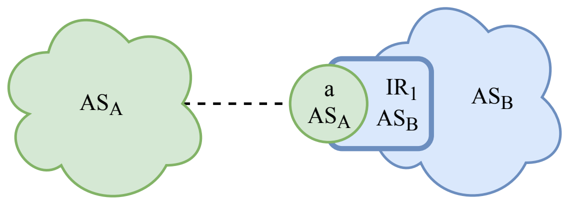


Figure 3: IR_1 (box) is annotated with AS_B representing its inferred operator, while its interface (circle), with IP address a, is annotated with AS_A representing that it is inferred to be connect to a router operated by AS_A.

that interdomain links are point-to-point, except public peering links at IXPs.

3.2 Annotating Last Hops with Ownership (§5)

When the last interface in a traceroute is not the destination, it could be on the border router of the network containing the destination. This occurs when an AS configures its firewall to prevent traceroute responses to probes from other networks. In this phase we infer the operator of an IR with no outgoing links to be the destination AS of the paths on which the IR's interfaces were observed.

This technique is surprisingly effective, and enables us to accurately infer links to networks that do not respond to traceroute probes from internal routers. These ASes are especially tricky, since we often do not see any addresses from their address space in the traceroutes. Some of the AS-links inferred in this step do not appear in our BGP paths, thus complementing our BGP-observed AS connectivity. `bdrmapIT` also relies on these ownership mappings in the graph refinement step (§6.1).

3.3 Annotating IRs and Interfaces (§6)

In order to deal with traceroute, routing, and IP space artifacts, the final phase is an iterative process that first annotates the IRs (§6.1), then annotates the interfaces (§6.2), and then repeats this process until completion, indicated by a repeated state. Annotations assigned in each iteration help refine the graph, enabling more accurate annotations in subsequent iterations. Each iteration reconsiders every annotation assigned based on the current annotations of neighboring IRs and interfaces. We do not revise the annotations assigned in the second phase, since those annotations are based entirely on static metadata.

4 PHASE 1: CONSTRUCT THE GRAPH

The first step is to construct an annotated IR graph from the traceroutes, alias resolution data, and external data sources. All subsequent refinement on the IR occurs locally, using only the static metadata labels (from the first phase) or the annotations (which may change during iterations) of its immediate neighbors. We do not directly consider remote IRs, but their annotations propagate across the graph in each iteration. We also extract static graph metadata — interface origin ASes and IXP prefixes (§4.1), link confidence labels (§4.2), origin AS sets (§4.3), and IR destination AS sets (§4.4).

4.1 Label AS-level Metadata

This phase labels the initial graph with additional metadata to enable subsequent inferences: origin ASes, IXP prefixes, and AS relationships.

Determining Origin ASes: We assume that one AS among the origin ASes for interface IP addresses on an IR is more likely than the others to be the operator of the IR. We derive interface origin ASes using BGP announcements collected by Routeviews [13] and RIPE RIS [12]. For each prefix we determine the origin AS as the last AS in the AS path. To determine the origin AS for an interface, we use the longest matching prefix from the route announcements. We then initialize the graph by annotating each interface with the origin AS of the corresponding IP address and create the IR origin AS set as the union of the IR's interface AS mappings.

Not every prefix is visible in BGP announcements, so we supplement this data with RIR delegations [2–4,7,11], using the AS identifiers in the extended delegation files to match IP prefixes with ASes. RIR delegations can be stale, since ASes can reassign prefixes, so we only use the prefixes from RIR delegations not already covered by a BGP prefix. Of the addresses seen in our experiments, 99.95% have a matching prefix in either BGP announcements, RIR delegations, or IXP prefixes.

Collecting IXP Addresses: `bdrmapIT` considers IXP prefixes specially, since some ASes originate IXP prefixes in BGP, which could cause unrelated ASes to be included in an origin AS set for an IR. We therefore compile a list of IXP prefixes using data volunteered by ISPs and IXPs to PeeringDB [10], Packet Clearing House (PCH) [9], and EuroIX [5], and do not consider BGP origin ASes for addresses covered by these prefixes when building origin AS sets.

Inferring AS Relationships: AS relationships constrain the set of possible paths, so we use them to constrain the set of ASes used for IR labeling. We rely on Luckie *et al.*'s technique [28] to determine whether two adjacent ASes in BGP paths are in a transit relationship. This technique also infers the customer cone for an AS, i.e., ASes reachable by customer links [28].

4.2 Assign Link Confidence Labels

The likelihood that an interface-to-interface link is a point-to-point link depends on the type of ICMP response used to infer the link. We

Label	Priority	Description
$IR_i \overset{N}{\longleftrightarrow} j$	1	Same origin AS or hop-distance of 1, where j does not respond with ICMP Echo Reply.
$IR_i \overset{E}{\longleftrightarrow} j$	2	Hop-distance of 1. j responds with ICMP Echo Reply.
$IR_i \overset{M}{\longleftrightarrow} j$	3	Hop-distance > 1. i and j have different origin ASes.

Table 3: Link type confidence labels: Nexthop (N), Echo (E), and Multihop (M).

use this dependence to label links with indicators of confidence in their existence, as follows. For each traceroute, we create a link from each IR to the first interface seen subsequently in that traceroute. When we encounter adjacent hops i and j, possibly separated by private addresses or unresponsive hops, we create a link between i's IR (IR_i) and j. We label the link with one of three labels in Tab. 3, determined by the distance between the hops, and the ICMP type of j's reply. If a link receives multiple labels, bdrmapIT uses the highest confidence one.

The highest confidence links, $IR_i \overset{N}{\longleftrightarrow} j$, provide the most reliable information; they also account for 96.4% of links seen in our traceroute datasets. A link receives this label in two cases. Both require that j responds with ICMP Time Exceeded or Destination Unreachable, which typically indicates that the traceroute probe arrived at interface j on the responding router, or that the router responded using j. In the first case, i and j have the same origin AS. We are not concerned with the hop-distance between them since that same origin AS likely operates them. In the second case, they have different origin ASes, but i and j have a hop-distance of one. We consider links derived from adjacent traceroute hops as reliable as hops with the same origin AS, since AS_j indicates that j's router is operated by AS_j, IR_i is operated by AS_j, or both.

When j responds with ICMP Echo Reply, we label the link $IR_i \overset{E}{\longleftrightarrow} j$. Unlike other response types, Echo Replies do not indicate that j was the ingress or egress interface, but rather that j was an interface on the responding router. As long as i and j are adjacent, we indicate this distinction with the label $IR_i \overset{E}{\longleftrightarrow} j$. Of the IRs in our datasets with links to at least one subsequent interface, 2.8% have $\overset{E}{\longleftrightarrow}$ links but no $\overset{N}{\longleftrightarrow}$ links.

Lastly, we use label $i \overset{M}{\longleftrightarrow} j$ when i and j are separated by unresponsive hops or private addresses, and $AS_i \neq AS_j$. In such cases we cannot assume that i and j are on routers operated by the same AS, and there could be one or more AS-hops between R_i and R_j. We use $\overset{M}{\longleftrightarrow}$ links only when no other link types are available.

Fig. 4 illustrates the process of assigning labels to links. We label the first link $IR_1 \overset{N}{\longleftrightarrow} b$ since the hops are adjacent and IR_2 responds with Time Exceeded. The next two links are labeled $IR_2 \overset{M}{\longleftrightarrow} c_1$ and $IR_4 \overset{N}{\longleftrightarrow} c_2$ respectively. In the third link c_1 and c_2 have the same origin ASes, leading us to presume the missing hops are operated by AS_c, while we are unable to draw the same inference in the second link. Finally, we label $IR_7 \overset{E}{\longleftrightarrow} d$ since IR_8 responds with an Echo Reply, whose source address in this case happened to be an off-path address, i.e., an interface not used to receive the incoming traceroute probe packets.

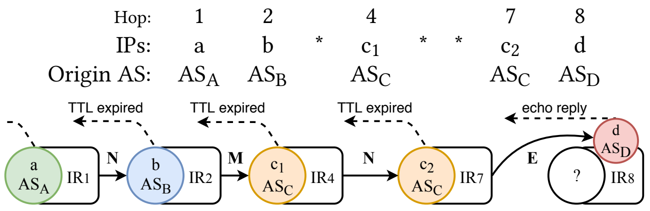


Figure 4: Deriving link labels from a traceroute: Nexthop, Echo, and Multihop.

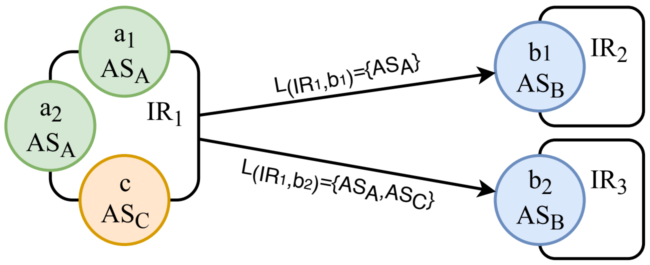


Figure 5: Using the paths from figure 2, path segment $a_1 - b_1$ crosses link (IR_1, b_1), so the AS set $L_{(IR_1,b_1)}$ contains AS_A, while segments $a_2 - b_2$ and $c - b_2$ both cross link (IR_1, b_2), so the AS set $L_{(IR_1,b_2)}$ contains AS_A and AS_C.

4.3 Assign Origin AS Sets to IRs

When we create a link, $IR_i \leftrightarrow j$, where i and j have different origin ASes, it is not immediately clear if IR_i is operated by AS_i or AS_j. Making that inference often requires analyzing the AS relationship between AS_i and AS_j, so we add the origin AS of interface i to the *link's origin AS set*, $L_{IR_i,j}$. Each origin AS set is specific to a link between an IR, in this case the IR containing the interface i, and a subsequent interface j. Keeping the origin AS set specific to link $IR_i \leftrightarrow j$, instead of creating a single origin AS set for j, helps prevent incorrect inferences in the event j ever appears as a third-party address. At the completion of this phase, $L_{IR_i,j}$ contains all origin ASes of the IR_i's interfaces seen immediately prior to j in a traceroute. In figure 5, $L_{(IR_1,b_1)}$ contains AS_A since segment $a_1 - b_1$ crosses AS_A. while $L_{(IR_1,b_2)}$ contains both AS_A and AS_C since those ASes are crossed by the path segments $a_2 - b_2$ and $c - b_2$.

We use the origin AS set to reason about *all* of the potential AS relationships between an IR and a subsequent interface's origin AS. For example, if an origin AS set contains $\{AS_A, AS_B\}$, and c has origin AS AS_C, then we expect at least one of $\{AS_A, AS_B\}$ to have a relationship with AS_C, or that a hidden AS exists between the ASes in the origin AS set and AS_C.

Origin AS sets illustrate a primary challenge of synthesizing the bdrmap and MAP-IT approaches. Specifically, neither previous approach had to worry about choosing among multiple origin ASes

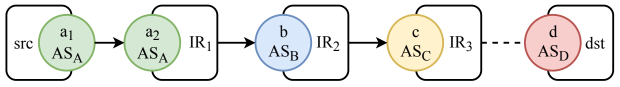


Figure 6: Traceroute with source address a_1 and destination address d. AS_D is the destination AS and c's IR_3 only appears at the end of our traceroutes. We can use AS_D to help determine IR_3's operating AS.

for a given router in the graph; `MAP-IT` did not even consider routers, and `bdrmap` only mapped the borders of a single, known origin AS.

4.4 Assign Destination ASes to IRs

To correctly label routers found only at the end of traceroute (§6.1.1), we build a destination AS set for each IR. This destination AS set contains the origin ASes of the traceroute destination addresses that resulted in a reply from at least one of the IR's interfaces.

We first compile destination AS sets for each interface. In Fig. 6, we add AS_D to the destination AS sets for a_2, b, and c. The lone exception is when a traceroute ends in an Echo Reply, in which case we do not record the destination AS for the last IR. In this case the destination AS adds no value, since it is always the same as the interface's origin AS, owing to the fact that the source address of an echo reply is simply the destination address probed.

We aggregate the destination AS sets for each interface on an IR into a single destination AS set for that IR. The possibility of prefix reallocation by ISPs [23] complicates this aggregation process. To detect likely reallocated prefixes, we look for interfaces with exactly two destination ASes, where one of the ASes matches the interface's origin AS, and the other AS has a customer cone of at most five ASes. This restriction on the customer cone size ensures we capture ASes who are small enough to likely receive reallocated prefixes from their provider. When these two ASes have no BGP-observable relationship, we assume that the relationship between them is missing due to prefix aggregation, which occurs when the provider aggregates the reallocated prefix into its own BGP announcements. If so, we remove the destination AS with the largest customer cone, which we infer to be the reallocating provider. After removing reallocated provider ASes, the IR destination AS set is simply the union of its interface destination AS sets.

5 PHASE 2: ANNOTATE LAST HOPS

In many datasets, the vast majority of IRs (≈ 98% in CAIDA's February 2018 ITDK [6]) have no outgoing links, caused by several factors: the IR was the destination or last reachable hop on a path probed by traceroute; or intermediate nodes rate-limited, blocked, or dropped ICMP responses. Phase 2 uses the destination AS sets compiled in the previous phase to annotate each IR, without outgoing links, with their operating AS regardless of the reason. The intuition behind this phase is to find a single AS with a known AS relationship with the IR's other origin ASes. The following heuristics first derive a list of acceptable candidates, then infer the best among them.

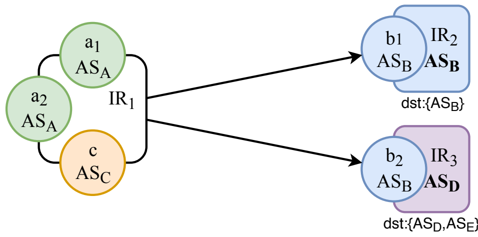


Figure 7: Using paths from Figure 2, IR_2 was seen by paths going to AS_B so its destination AS set is $\{AS_B\}$. Since AS_B matches the origin AS of interface b_1, IR_2 is annotated with AS_B. IR_3 was seen by paths going to both AS_D and AS_E so its destination set includes both. AS_D has a relationship with AS_B so IR_3 is annotated with AS_D.

5.1 When the Destination AS Set is empty

Since we do not use echo replies to build the destination AS set (§4.4), when all interfaces on an IR are only seen in echo replies, the destination AS set will be empty. We have only the origin AS set to reason about the IR. In the February 2018 ITDK, 73.3% of last hop IRs have an empty destination AS set. We do not know of any techniques for improving the mappings for these IRs without additional probing.

If one or more ASes in the origin AS set has a relationship with all other ASes in the set, we select that AS. In the event of a tie, we select the AS with the smallest customer cone, inferring that AS to be a customer of the other ASes. Otherwise, we look for an AS not in the origin AS set that has a relationship to all ASes in the set, and infer that AS to be connected to the other ASes. Finally, we select the AS with the most interface AS mappings in the set, breaking ties by selecting the AS that has the smallest customer cone.

5.2 When the Destination AS Set is not empty

Destination ASes for an IR enable greater accuracy than relying on the IR's interfaces alone, because destination ASes can provide topological constraints that inform router ownership. The order in which we describe the different cases is both the order in which they appear in the algorithm (Alg. 1), and their frequency order in our datasets.

Algorithm 1 Annotating Last Hop Router, ir

```
1:  $D \leftarrow \text{DESTINATIONASES}[ir]$
2:  $O \leftarrow \{i \in ir \mid i.\text{ASN}\}$
3:  if $|O \cap D| = 1$ then return the single AS
4:  $D_{rel} \leftarrow \{ d \in D \mid \exists o \in O : \text{HASRELATIONSHIP}(d, o) \}$
5:  if $|D_{rel}| > 0$ then
6:      return $\max_{d \in D_{rel}} |\text{CUSTOMERCONE}[d] \cap D|$
7:  $a \leftarrow \min_{asn \in D} \text{CONESIZE}[asn]$
8:  $C \leftarrow$ customers of any $o \in O$
9:  if $|\text{PROVIDERS}[a] \cap C| = 1$ then return the single AS
10: return $a$
```

Overlapping ASes (line 3): As we visit each IR, we first select the ASes in common between the origin and destination AS sets. When there is only one overlapping AS, we infer that AS to operate the router. In case there are multiple overlapping ASes, we select the AS with the smallest customer cone, assuming this AS is using a reallocated prefix from the larger AS.

Relationship Between Origins and Destinations (lines 4-6): If there are no overlapping ASes, we next look for destination ASes that have a relationship with any of the origin ASes. If there is only a single AS, we use it as the AS annotation. If there are multiple, we use the AS with the largest customer cone, inferring it is a transit provider for the others. In fig 7, the relationship between interface b_2's AS_B and AS_D in IR_3's destination justifies annotating IR_3 with AS_D.

No Relationship ASes (lines 7-10): The final case is when there is no AS relationship between any destination and origin ASes. Initially, we look for an AS between the origins and destinations, specifically looking for ASes that are both a provider to at least one destination AS, and also a customer of at least one origin AS. If we find exactly one such AS, we annotate the IR with it. Otherwise, we select the destination AS with the smallest customer cone.

6 PHASE 3: GRAPH REFINEMENT

The graph refinement loop has two steps. The first step iterates over the IRs, using their outgoing links to annotate IRs with their operating AS (§6.1). The second step relies on IR annotations to update the AS annotation of each interface with the interconnecting AS (the other side of the link) (§6.2).

Prior to entering the graph refinement loop, `bdrmapIT` initializes all interface annotations with the origin AS of the interface. Throughout iterations of the graph refinement loop, annotations propagate across the graph, enabling `bdrmapIT` to refine the annotations, improving its accuracy (§6.3). We iterate until we reach a repeated state, i.e., when all of the annotations at the end of one iterations are the same as the annotations at the end of a previous iteration.

6.1 Annotate IRs

The first phase of refinement is to annotate all IRs with an AS (Alg. 2). Intuitively, we use the current AS annotations of the IRs and interfaces to determine the most frequently appearing AS for an IR's set of subsequent interfaces, similar to MAP-IT's approach. We also leverage adapted `bdrmap` heuristics and framing assumptions to apply exceptions and tiebreakers.

First, we sum the votes of subsequent interfaces (§6.1.1). We assume that typically, the AS with the most votes, representing the largest number of links from an IR, is the IR operator. Next, we change votes if we encounter a reallocated prefix (§6.1.2). We then check if the votes match one of our exception conditions (§6.1.3) that violate the assumptions of our majority-vote annotation technique. If the votes do not match any exception condition, we give each IR interface a vote, using its origin AS, and select the AS with the highest number of votes, breaking ties if necessary (§6.1.4). Finally, we check for a hidden AS in §6.1.5, possibly replacing the selected AS with a hidden AS. `bdrmapIT` uses the final AS selection as the IR's AS annotation.

Algorithm 2 Annotating IR, *ir*

```
1:  $V$: counter for AS votes
2:  $M$: map of $ir$ origin ASes to subsequent ASes
3:  for all interface $j \in$ SUBSEQUENT[$ir$] do
4:      $a \leftarrow$ IRLinkHeuristics($ir, j$)                    ▷ §6.1.1
5:      if $a$ != NULL then
6:          INCREMENT($V[a]$)
7:      $\forall o \in S_{ir,j}$ : add $o$ to $M[a]$
8:  Fix reallocated prefixes                                  ▷ §6.1.2
9:  for all $i \in ir$.INTERFACES do INCREMENT($V[i.\text{AS}]$)
10: Look for exception cases                                  ▷ §6.1.3
11: $R \leftarrow ir.\text{ORIGINS} \cup \{v \in V \mid \exists o \in M[v] : \text{REL}(o, v)\}$
12: if $R$ != $ir$.ORIGINS then return $\max_{v \in R} V[v]$       ▷ §6.1.4
13: $a \leftarrow \max_{v \in V} V[v]$                              ▷ §6.1.4
14: return Look for hidden AS between $M[a]$ and $a$
```

6.1.1 Apply Link Vote Heuristics. The first step is to count the link votes based on three heuristics represented in Alg. 3. As explained in §4.2, when computing the link votes for an IR, we only use the $\overset{N}{\longleftrightarrow}$ links, relying on $\overset{E}{\longleftrightarrow}$ and $\overset{M}{\longleftrightarrow}$ links only when they are the only links available. This step begins by checking for the three cases (line 2-8, detailed next) in which we do not use the AS annotation on the interface as the vote. Usually, none of these cases apply, and we rely on the interface's annotation (line 9).

Algorithm 3 IRLinkHeuristics(ir, j)

```
1: if $j.\text{AS} \in L_{ir,j}$ then return $j$.AS
2: if $j \in$ IXP addresses then return $\max_{a \in L_{ir,j}}$ CONESIZE[$a$]
3: $AS_j \leftarrow$ ANNOTATION[$j$.IR]
4: if $AS_j$ is unannounced then return NULL
5: if $j$ is unannounced then return $AS_j$
6: if $j.\text{ASN} \neq AS_j \wedge \exists a \in L_{ir,j}$ : HASREL($a, AS_j$) then
7:     $D \leftarrow$ the set of destination ASes for link $E_{ir,j}$
8:     if $j.\text{ASN} \notin D$ then return $AS_j$
9: return ANNOTATION[$j$]
```

IXP Address (line 2): When *ir*'s subsequent interface *j* is an IXP public peering address, we instead use the AS in $L_{ir,j}$ (the origin AS set for interfaces on *ir* seen prior to *j* in a traceroute) with the largest customer cone; this AS is likely the top of the transit hierarchy. This choice reflects conventional assumptions [17] that in general AS paths are valley-free, contain at most one peering link, and that networks do not forward packets from a provider to a peer. Since we have a strong indication that *i* is used for a public peering link, we try to identify the likely transit provider AS among the origin ASes of the *ir* interfaces.

Unannounced Addresses (line 5): After ensuring the subsequent interface address does not match a prefix in our IXP dataset, we check if the address fails to match any prefix in BGP announcements or RIR delegations, which we call unannounced addresses. In the datasets used in §7, this occurs for 0.1% of the interface

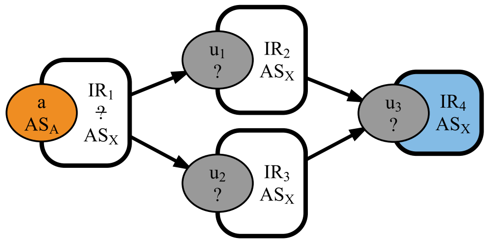


Figure 8: Annotating IRs with unannounced interface addresses. IR_4 was annotated by the last hop heuristic. In the first iteration of the graph refinement loop IR_2 and IR_3 are annotated with AS_X, enabling the annotation of IR_1.

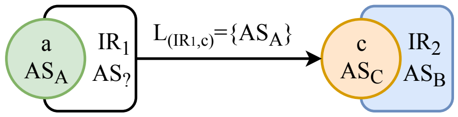


Figure 9: Interface c is potentially a third-party address because its origin AS_C, its IR's annotation AS_B, and the ASes $\{AS_A\}$ in $L_{IR_1,c}$ are all different.

addresses. While the empty origin AS for the subsequent interface provides no value, we can instead give a vote to its IR's AS annotation.

There are two ways that IRs with unannounced interface addresses receive annotations, either using destination ASes in §5 or using subsequent ASes in this step. In this heuristic, we are concerned with IRs that have a link to a subsequent interface with an unannounced address, in which case we give a vote to the IR's AS annotation. Due to the iterative nature of the graph refinement loop, using the AS annotation of the subsequent interface's IR enables us to annotate IRs with links to unannounced addresses, even when they are several hops removed from an interface with an address in our IP-AS mappings, or an IR annotated in §5. As shown in Fig. 8, by the second iteration of the graph refinement loop we used the AS annotation for IR_4 to correctly annotate IR_1 with AS_X, even though IR_1 only has links to unannounced addresses.

Third-Party Addresses (lines 6-8): At this point, we know that the subsequent interface address is not in our IXP dataset, and has a matching prefix in either BGP announcements or RIR delegations. The goal in this step is to assess whether we should use the origin AS as a constraint, which we do unless we believe the router used a third party address to reply.

Third-party addresses typically result from asymmetric routing, when the interface used to respond to a traceroute probe (egress) is different from that which received the probe (ingress) [19,26]. When the ingress and egress interfaces use the same AS address space, or the router puts the ingress address in the reply source field, asymmetric routing presents no problems in IR annotation. Difficulties arise when the egress and ingress interfaces come from different AS address spaces, and the router uses the egress interface address as the source address of the reply.

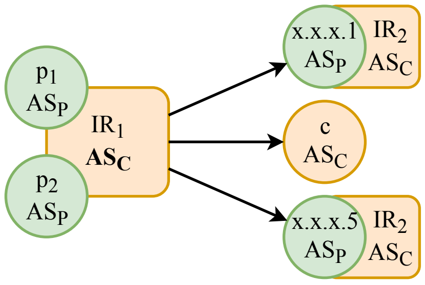


Figure 10: When AS_C is a customer of AS_P we annotate IR_1 with AS_C.

Any interface with an origin AS that is both not in the link's origin AS set, and differs from its IR's annotation, is a potential third-party address (Fig. 9). `bdrmapIT` uses a two-step test to infer whether c is a third-party address. First, there must be an AS relationship between at least one AS in $L_{IR_1,c}$ and AS_B. This AS relationship indicates that the traceroute probe could get to AS_B from the origin AS without going through AS_C. Second, AS_C must not appear in the destination ASes specific to IR_1 and c. This test indicates that probes transmitted from IR_1 to c were never destined to AS_C. While not exhaustive, this test gives a reasonably strong indication that c is likely a third-party address, so we should not include its annotation in the voting. Instead, we include a vote for IR_2's annotation, in this case AS_B. If c's IR does not yet have an annotation, only possible in the first iteration of the graph refinement loop, we skip the third-party tests entirely.

6.1.2 *Correct Reallocated Prefixes.*

In the previous step we determined the vote for each subsequent interface independently, but in this step we evaluate all of the subsequent interfaces together. As in §4.4, we try to identify situations where a provider reallocated some of its address space to a customer (Fig. 10), but continues to announce a containing prefix into BGP. To prevent incorrect annotations of a provider router with a customer AS, we take a conservative approach. Our test first looks at all of IR_1's subsequent interfaces that map to an AS seen in its origin AS set, in this case `x.x.x.1` and `x.x.x.5`. Then this step collects the IR annotations for those interfaces, along with the /24 prefix for their addresses. If all of the annotations are the same, the single annotation is a customer of an IR origin AS, and all of the addresses have the same /24 prefix, we conclude that the prefix was reallocated. In this case, all of the matching subsequent interfaces have the prefix `x.x.x/24`, and their IRs are annotated with AS_C, so we change their votes from AS_P to the customer AS_C.

To avoid mistakenly annotating an IR with a customer AS, we require a single prefix for all subsequent interfaces, as well as multiple links. Often, the unannounced reallocated prefixes are smaller than /24, but matching against a /24 catches smaller prefixes without incurring too much risk of matching too large a prefix.

6.1.3 *Check for Exceptions.*

From developing and using `MAP-IT` we learned that, in general, the AS that receives the highest number of votes operates the IR. Adapting `bdrmap` heuristics for more general use in `bdrmapIT` led us to consider two exceptions to this general rule.

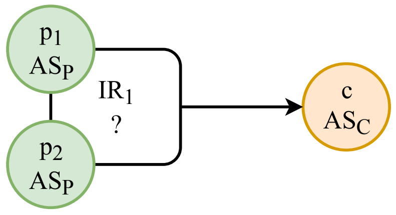


Figure 11: A multihomed customer can present an exception to majority voting, as when AS_C is a multihomed customer of AS_P, selecting the most frequent AS results in IR_1 inferred to be owned by AS_P instead of AS_C.

Multihomed to a Provider: The most common exception is for links between transit providers and customer ASes, where selecting the AS with the most votes results in an incorrect choice. In accordance with industry convention, transit link interfaces usually use addresses from the provider's address space. The result is that border IRs operated by a customer AS often have more interfaces with addresses from their providers' address space than links to subsequent interfaces with addresses from the customer's address space.

When the customer is a large ISP the IR voting system usually does not make false inferences, but when the IR belongs to a stub AS, our voting system can make false inferences. When a customer IR is multihomed to a transit provider, the IR will have multiple interfaces each with an address from the provider; if traceroute paths observe fewer links to IRs with addresses in the customer network, as in Fig. 11, a pure voting system will make a false inference.

We identify and account for these exceptional cases. When there is only a single subsequent AS_j, we check to see if AS_j is a customer of any IR origin AS. Returning to the example in Fig. 11 we check if AS_C is a customer of AS_P, and annotate IR_1 with AS_c if it is.

We do not use the single subsequent AS exception when there is no relationship between it and any origin AS, instead relying on votes (§6.1.4) or looking for a hidden AS (§6.1.5). We choose not to select an AS yet since BGP AS paths typically contain the vast majority of transit relationships.

Multiple Peers/Providers: The second exception is when the IR interfaces all have the same origin AS, there are multiple subsequent ASes, and all of the subsequent ASes are either peers or providers of that AS. In this case, we expect that the origin AS is the AS that operates the IR, since it is the common denominator between the subsequent ASes. Conversely, when the IR has multiple interface origin ASes, and there is a single subsequent AS that is a peer or provider of every origin AS, we select the subsequent AS for the same reason. We annotate the IR with the selected AS provided that it has at least half as many votes as the AS with the most votes. If it has less than half the votes, it suggests that it is not actually the operating AS, so we do not apply the exception.

6.1.4 *Apply IR Vote Heuristics.*

After determining the subsequent interface votes, adjusting reallocated prefixes, and checking for exception conditions, we use the votes to determine the AS annotation for the IR. As long as at least one subsequent AS has an observed relationship with an IR origin AS, then the election is held between the IR origin ASes and any subsequent ASes that have a relationship with an IR origin AS. This constraint helps ensure that the selected AS will have a relationship with at least one IR origin AS. If the new restricted set only contains IR origin ASes, then we revert to using all of the ASes with a vote, but check for a hidden AS (§6.1.5) following this step.

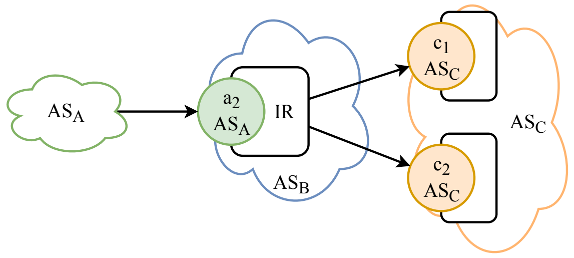


Figure 12: Hidden ASes occur when traceroutes traverse an IR in AS_B, but traceroute never observes any IP addresses from AS_B on it or any IR adjacent, suggesting an AS path of A-C, instead of A-B-C

We use the selected AS, which has the most votes, as the annotation for the IR. Our justification follows from the observation that an interface address in AS_A indicates that its IR, the IRs connected to it, or both, are operated by AS_A, since interdomain link interfaces use the address space of only one of the two networks. Viewed through this lens, every interface on, and subsequent to, the IR is circumstantial evidence of the operating AS. Selecting the AS with the most votes also selects the AS with the most circumstantial evidence.

Occasionally, multiple ASes will tie for the highest vote. We break the tie by selecting the most likely customer AS from the group, by choosing the AS with the smallest customer cone. Since transit link interfaces are usually addressed from the provider's address space, and we expect that most interdomain links seen in traceroute are transit links, we try to select the customer AS.

6.1.5 *Check for a Hidden AS.*

Finally, we check to see if our selection has a relationship to any member of the IR's origin AS set. If so, we use this selection for the IR annotation. Otherwise, we look for the possibility of a hidden AS. Occasionally, despite a traceroute traversing an AS, it reports no IP addresses from that AS (Fig. 12). We most often encounter hidden ASes when a transit link between a small ISP and its customer uses the customer's address space.

To avoid an incorrect annotation, we attempt to find an AS that bridges between the selected AS and subsequent ASes by finding an AS that is a customer of the selected AS, and a provider of a subsequent AS. When there is a single such AS, we change our selection to that AS. Otherwise, we leave our selection unmodified.

6.2 Annotate Interfaces

Following the router annotation step, we update the interface AS annotations to align the interface AS annotations with the router it connects to. As long as the interface address is not an IXP address, we assume the interface connects to one router, and therefore one

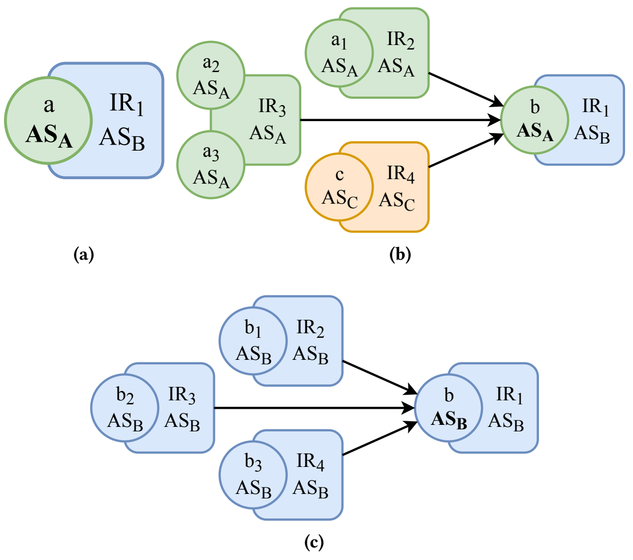


Figure 13: (a) If the interface's origin AS is different than the IR's annotation, we annotate with the interface's origin AS. (b,c) If they are the same, we annotate with a single AS from the connected IRs.

AS. However, the operator of the connected router might be obscured, either due to mistakes in the router annotations, or as a result of it appearing as a third-party address. This step selects a single AS annotation for the interface.

An interface origin AS will either come from the AS operating its router, or from a different AS directly connected to the interface that provides the interface address for interconnection. Thus, if the origin AS for an interface differs from the AS annotation for the interface's IR, we use the interface address' origin AS as the interface AS annotation, since that AS operates the connected IR. That is, if the interface address does not come from the AS operating the router it is on, it must come from the AS operating the router it connects to. In Fig. 13a, we previously inferred that IR_1 is operated by AS_B, leading us to conclude that a connects to a router operated by AS_A, so we annotate a with its origin AS_A.

When the interface origin AS is the same as the current AS annotation for IR_1 (Fig. 13b and Fig. 13c), we select one of the ASes from the IRs connected by links in our graph. Similar to §6.1, we use a voting system, but in this step we give each connected IR a vote for each of its interfaces seen prior to interface b in a traceroute. In Fig. 13b, AS_A receives three votes and AS_C, which might be an errant annotation, receives one vote. To determine the AS annotation of the interface, we select the AS with the most votes, breaking ties using the tied AS with the largest customer cone that also has a BGP-observed relationship to the interface origin AS. If no tied AS has a relationship to the interface AS, then we use the interface address' origin AS as the annotation to avoid negatively impacting the IR annotations with an incorrect inference.

Finally, so far we have focused on potential interdomain links. It is possible, as in Fig. 13c, that the interface origin AS and IR AS annotations are the same, because the same AS operates its router and the connected router. In these cases we annotate b with its origin AS.

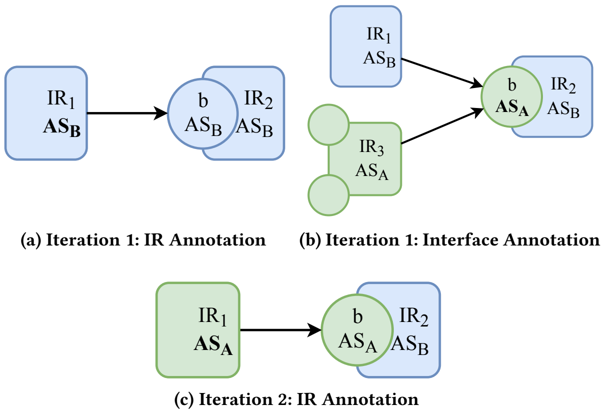

IR1
ASB
b
ASB
IR2
ASB
(a) Iteration 1: IR Annotation
IR1
ASB
IR3
ASA
b
ASA
IR2
ASB
(b) Iteration 1: Interface Annotation
IR1
ASA
b
ASA
IR2
ASB
(c) Iteration 2: IR Annotation

Figure 14: `bdrmapIT` refines the graph as it progresses from the first iteration to the second. The annotation for IR_1 is corrected from AS_B (a) to AS_A (c).

6.3 Refine the Graph

`bdrmapIT` repeatedly updates IR annotations (§6.1) and interface annotations (§6.2) until no modifications are made in an iteration. Fig. 14a illustrates the approach. During the IR annotation stage of the first iteration, IR_1 has only a single link to the subsequent interface b, with AS annotation AS_B. If AS_B is either a customer of AS_A, or a peer with a smaller customer cone, we might incorrectly annotate IR_1 with AS_B. Fortunately, in the interface annotation stage (Fig. 14b), b has links to two IRs. Since IR_3 has two interfaces, AS_A receives the most votes, changing b's annotation from its origin AS to AS_A. When we return to IR annotations in the second iteration of the graph refinement loop (Fig. 14c), IR_1 uses the new annotation for b, which corrects the annotation for IR_1 to AS_A.

7 EVALUATION

We validate our approach against ground truth from four networks: a Tier-1 network, a large access network, and two research and education (R&E) networks. Except for one R&E network, we reused the 2016 ground truth acquired for the `bdrmap` evaluation, which was gathered by first running `bdrmap` from a VP in each network. We created the 2018 ground truth dataset by first running `bdrmapIT` on traceroutes initially collected by `bdrmap`. In both cases, the resulting inferences were sent to the network operators for each VP network to obtain a validation dataset. We did not ask them to validate missing inferences due to the burden on the network operator, although a small number of interdomain links appear in our ground truth that `bdrmap` did not identify. The remaining R&E network, labeled R&E 1, provided us with router configurations of its primary AS, which includes internal and interdomain links involving its backbone.

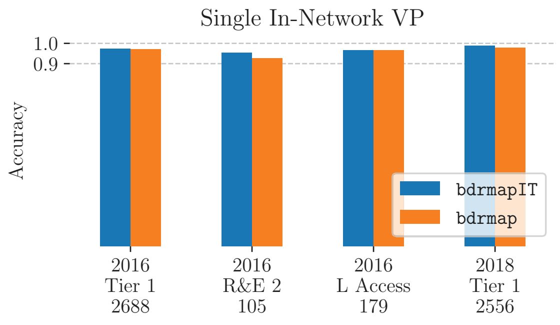


Figure 15: **(all data) `bdrmapIT` is more accurate than `bdrmap` for the 4 ground truth networks. Bottom number reports links visible in the paths.**

To evaluate `bdrmapIT`, we ran three separate experiments using datasets from the spring of 2016 and spring of 2018. We used validation data from the same time period as the dataset.

(1) We regression tested against `bdrmap` to ensure that the adapted heuristics in `bdrmapIT` perform at least as well as the original `bdrmap` heuristics (§7.1);
(2) We demonstrated the power of our new approach on Internet-wide datasets with no VPs in our validation networks, showing that `bdrmapIT` has high accuracy and vastly outperforms `MAP-IT` (§7.2);
(3) We show that our accuracy does not diminish when datasets have fewer traceroute VPs than a full ITDK (§7.3).

7.1 bdrmapIT Validation on bdrmap Data

The first experiments compare mappings generated by `bdrmapIT` to inferences drawn by `bdrmap`, ensuring that our adaptations of the `bdrmap` heuristics do not adversely affect their accuracy. `bdrmap` has been running in several networks since 2016; we feed the traceroutes and alias resolution from those `bdrmap` runs as input to `bdrmapIT`. We used the data that `bdrmap` gathered as an existing dataset, ensuring that `bdrmapIT` and `bdrmap` base their mappings on identical traceroute data.

Our validations (Fig. 15) confirm that `bdrmapIT` performs at least as accurately as `bdrmap` in its limited problem domain, which is mapping the border of a single network using traceroutes from a single VP in that network. In fact, `bdrmapIT` performs slightly more accurately than `bdrmap`, primarily due to mapping past the VP AS border, enabling better hidden AS and third party identification. While these results verify that adapting the `bdrmap` heuristics to the `MAP-IT` framework did not weaken their ability to map the border of the VP network, they do not demonstrate the true benefits of `bdrmapIT`'s combined heuristics. The primary goal in the creation of `bdrmapIT` is accurate border mapping for all ASes in Internet-wide traceroute datasets, which is shown in the next set of experiments.

7.2 bdrmapIT Validation on ITDK Data

The second set of experiments demonstrate the benefit of `bdrmapIT`'s adapted heuristics to mapping network borders in Internet-wide datasets. To highlight the differences between `bdrmapIT` and `bdrmap`, we used traceroute datasets without VPs in any of the validation networks. We compare `bdrmapIT`'s results to `MAP-IT`, which was also designed for Internet-wide border mapping.

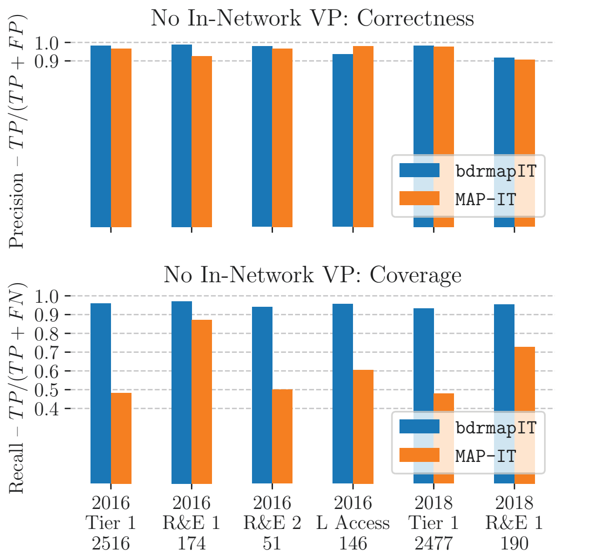


Figure 16: **(no in-net VPs) `bdrmapIT` has far better coverage than MAP-IT. We do not use in-network VPs. Under the network label is the number of links visible in the paths.**

For these experiments we used the publicly available CAIDA ITDK datasets for March 2016 [1] and February 2018 [6], which include traceroutes run over 15 days and alias resolution for some interfaces seen in the traceroutes. From each ITDK we removed traceroutes from a VP in one of our ground truth networks, ensuring that no traceroutes originated in any network we validated against. This left us with traceroutes from 109 of 111 VPs in the 2016 ITDK, and from 141 of 146 VPs in the 2018 ITDK. All the traceroutes used ICMP Paris traceroute [15,39], which controls the IP and transport headers to reduce load balancing, and a combination of iffinder [21] and MIDAR [22] for alias resolution.

Similar to `MAP-IT`'s validation [31], we present the precision and recall of our inferences. Precision in this context is the fraction of inferred interdomain links that are correct, i.e., they are not internal to a network, and we correctly identified the connected networks. Recall is the number of correctly identified interdomain links that appear in the dataset. When computing the recall, we exclude interfaces which only appeared as Echo Replies. Unlike in the previous experiments, which only validate correctness, since we obtained the ground truth based on `bdrmap`'s inferences, these experiments test `bdrmapIT`'s ability to find interdomain links in a dataset.

Fig. 16 shows that `bdrmapIT` clearly outperforms MAP-IT, with better precision for all of the ground truth networks except the large access network, and vastly better recall, demonstrating the benefit of the adapted `bdrmap` heuristics. Overall, `bdrmapIT` achieved 91.8%-98.8% precision, and 93.2%-97.1% recall.

All the adapted heuristics play a role in `bdrmapIT`'s higher accuracy, but the largest improvement comes from the use of destination

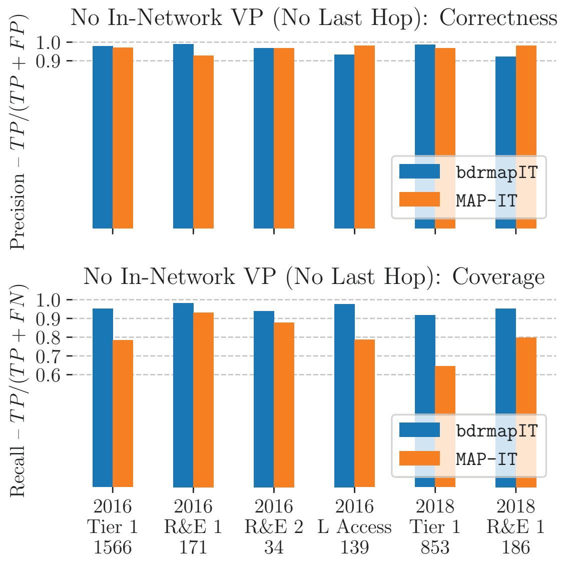


Figure 17: (no in-net VPs and no last hops) bdrmapIT has better coverage of interdomain links seen in the middle of the paths. Under the network label is the number of links.

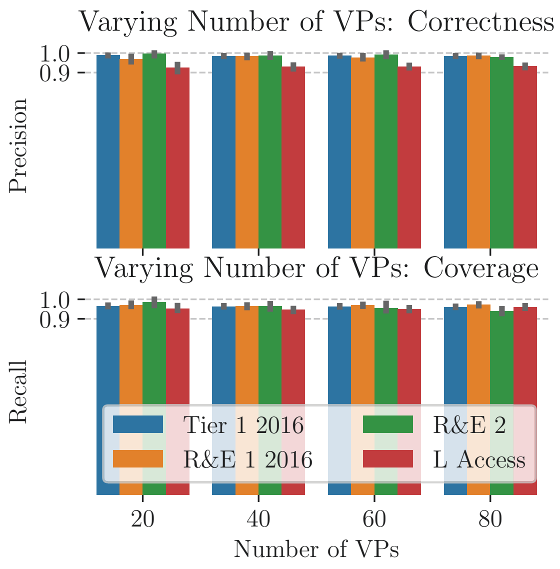


Figure 18: bdrmapIT performance is does not diminish as the number of VPs is reduced. The bars show the average for the precision and recall, along with the standard error between the five groups.

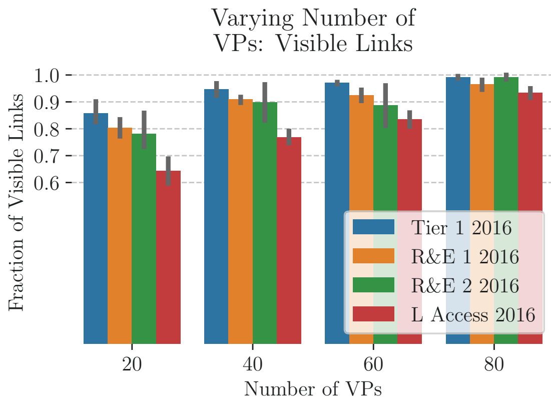


Figure 19: The number of visible links (seen in traceroutes) increases with the number of VPs. The bars show the number of interdomain links visible in the dataset using only the VPs in the set, along with the standard error.

ASes in the last hop heuristic (§5). MAP-IT does not use destination information, so it is unable to identify those links. Another prominent reason is that bdrmapIT's enhanced ability to leverage AS relationships, specifically in the third party heuristic (§6.1.1) and the multihomed customer exception (§6.1.3), improves on MAP-IT's coverage of low visibility links at the Internet edge. Links between an ISP and an edge AS are especially problematic for MAP-IT, since traceroute often reveals more interfaces from the provider's address space on a border router than customer addresses past the border, if an address from the operating AS even appears in a traceroute.

Finally, Fig. 17 shows the difference in coverage when we exclude the interdomain links which only appear as the last hop in the traceroute dataset. bdrmapIT still substantially outperforms MAP-IT, indicating that the adapted heuristics, and our overall more aggressive inference strategy, leads to significantly better results.

7.3 Effect of Decreasing VPs

The next set of experiments evaluated whether bdrmapIT's performance was reliant on the number of VPs included in the ITDK datasets. We validated bdrmapIT's performance using groups of 20, 40, 60, and 80 VPs, running five experiments in each group using five randomly chosen sets of VPs. For all experiments we excluded the VPs in our ground truth networks.

The results are shown in Fig 18. Each graph shows the average of the five sets of VPs for each group, along with the standard error. Surprisingly, bdrmapIT's accuracy does not diminish as the number of VPs decreases. In the groups with 20 VPs, the precision ranges from 92.4%-99.6% and the recall is between 95.4%-98.6%. Similarly, when we increase the number of VPs to 80, the precision (93.1%–98.5%) and recall (94.0%–97.2%) remain roughly equivalent, with the means falling within a standard deviation of each other. Although the number of interdomain links visible in the dataset drops with the reduction in VPs (Fig 19), bdrmapIT's ability to correctly identify those that appear does not diminish. This is an important result, as researchers might have only a few VPs at their disposal.

7.4 Importance of Alias Resolution

Finally, we investigate the impact of alias resolution on bdrmapIT's accuracy. First, we investigate the impact of using a different alias resolution technique on the same set of traceroutes. Specifically, we use kapar [20] along with midar and iffinder, while the results

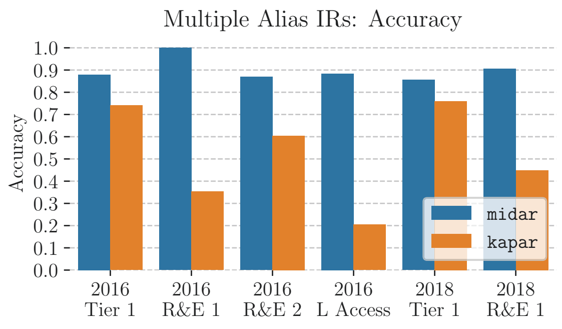


Figure 20: Comparison between alias resolution with and without `kapar`, excluding all IRs without multiple aliases. The decreased alias group precision using `kapar` decreases the `bdrmapIT`'s accuracy for the ITDK datasets, compared to using `midar` and `iffinder` alone.

presented in §7.2 use only the `midar` and `iffinder` techniques to infer router aliases. Second, we show that `bdrmapIT` performs nearly equivalently with the `midar` and `iffinder` alias resolution as it does without any alias resolution.

`kapar` Alias Resolution: Along with the alias resolution datasets we used in §7.2, the CAIDA ITDK includes a second alias resolution dataset that includes `kapar`. Unlike `midar`, which produces highly precise alias groups, `kapar` attempts to increase the number of grouped aliases, which can result in less precise groupings [8]. To determine the impact of less accurate, but larger alias groups on `bdrmapIT`'s inferences, we ran experiments for both the 2016 and 2018 ITDKs using the alias resolution which includes `kapar`.

The results, shown in Fig 20, clearly demonstrate that the less precise IRs generated by `kapar` decrease the accuracy of `bdrmapIT`'s inferences. To highlight the differences between the alias groupings, in Fig 20 we only include IRs with multiple aliases. In our ground truth datasets, `kapar` has a tendency to mistakenly group interfaces into a single IR, when in actuality they are used on different physical routers. Since `bdrmapIT` ensures that each router receives a single AS annotation, and then uses that information to determine interdomain links, imprecise alias resolution results in inaccurate inferences.

No Alias Resolution: Our final experiment aims to determine the impact of using `midar` and `iffinder` alias resolution as compared to not using any alias resolution. To do so, we ran `bdrmapIT` on the ITDK datasets, but treated each interface as a separate IR. The results are nearly identical, with less than 0.1% difference in accuracy between using alias resolution with `midar` and `iffinder`, and using an interface graph with no alias resolution.

Interestingly, the aggregation resulting from alias resolution can impact the results both positively and negatively. Occasionally, the additional IR links enable `bdrmapIT` to more accurately determine the IR operator, when one or more IR interface would not have sufficient constraints for `bdrmapIT` to make a correct inference. Conversely, reallocated addresses and third party addresses seen subsequent to a single interface can add confusion, causing `bdrmapIT` to infer the incorrect operator for an IR group, while without alias resolution the mistake would be limited to part of the IR. In our experiments, the negative impacts of alias resolution occurred exclusively at the edge of the Tier 1 network, where reallocated prefixes are common. Further investigation is necessary to determine when `bdrmapIT` with alias resolution performs better than using an interface graph.

8 CONCLUSION

We addressed the surprisingly challenging problem of mapping the borders of IP networks, which currently hampers both research and regulatory efforts. In addressing this challenge, we presented `bdrmapIT`, a traceroute analysis technique designed to infer the operating AS for routers and identify links between Internet networks. Our method synthesizes two previous approaches, `bdrmap` and MAP-IT, leveraging the strengths of each technique to create a general-purpose solution.

To evaluate `bdrmapIT`, we performed experiments from in-network and out-of-network VPs, validating the accuracy of our technique and demonstrating that `bdrmapIT` outperforms its predecessors. We performed additional experiments demonstrating that `bdrmapIT`'s performance does not diminish as we reduce the number of VPs. Our results suggest that `bdrmapIT` can form the foundation upon which to address other network diagnostic challenges, including congestion measurement [32,38], resilience assessment [14,25,33,36], and traffic estimation [35]. We publicly release our source code.

REFERENCES

[1] 2016. Internet Topology Data Kit - March 2016. http://www.caida.org/data/internet-topology-data-kit/.
[2] 2018. AFRINIC Extended Allocation and Assignment Reports. ftp://ftp.afrinic.net/pub/stats/afrinic.
[3] 2018. APNIC Extended Allocation and Assignment Reports. ftp://ftp.apnic.net/pub/stats/apnic.
[4] 2018. ARIN Extended Allocation and Assignment Reports. ftp.arin.net/pub/stats/arin.
[5] 2018. Euro-IX IXP Directory. https://www.euro-ix.net/tools/ixp-directory.
[6] 2018. Internet Topology Data Kit - February 2018. http://www.caida.org/data/internet-topology-data-kit/.
[7] 2018. LACNIC Extended Allocation and Assignment Reports. ftp.lacnic.net/pub/stats/lacnic.
[8] 2018. Macroscopic Internet Topology Data Kit (ITDK). http://www.caida.org/data/internet-topology-data-kit/.
[9] 2018. Packet Clearing House: Internet Exchange Directory. https://prefix.pch.net/applications/ixpdir/menu_download.php.
[10] 2018. PeeringDB. https://peeringdb.com/api.
[11] 2018. RIPE Extended Allocation and Assignment Reports. ftp://ftp.ripe.net/pub/stats/ripencc.
[12] 2018. RIPE RIS Raw Data. https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-raw-data.
[13] 2018. University of Oregon Route Views Project. http://www.routeviews.org/.
[14] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 2000. Error and attack tolerance of complex networks. *Nature* 406 (June 2000).
[15] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. 2006. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*.
[16] Kai Chen, David R. Choffnes, Rahul Potharaju, Yan Chen, Fabian E. Bustamante, Dan Pei, and Yao Zhao. 2009. Where the Sidewalk Ends: Extending the Internet As Graph Using Traceroutes from P2P Users. In *Proceedings of ACM CoNEXT*.
[17] Lixin Gao. 2001. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM ToN* 9, 6 (2001), 733–745.
[18] B. Huffaker, A. Dhamdhere, M. Fomenkov, and k. claffy. 2010. Toward Topology Dualism: Improving the Accuracy of AS Annotations for Routers. In *Proceedings of the Passive and Active Measurement Conference (PAM)*.
[19] Y. Hyun, A. Broido, and k. claffy. 2003. On Third-party Addresses in Traceroute Paths. In *PAM*. San Diego, CA.
[20] Ken Keys. 2010. Internet-scale IP alias resolution techniques. *ACM SIGCOMM CCR* 40, 1 (2010), 50–55.
[21] Ken Keys. 2018. iffinder. https://www.caida.org/tools/measurement/iffinder/.
[22] K. Keys, Y. Hyun, M. Luckie, and k. claffy. 2013. Internet-Scale IPv4 Alias Resolution with MIDAR. *IEEE/ACM ToN* 21, 2 (Apr 2013), 383–399.
[23] Thomas Krenc and Anja Feldmann. 2016. BGP Prefix Delegations: A Deep Dive. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*.
[24] Anukool Lakhina, John W. Byers, Mark Crovella, and Peng Xie. 2003. Sampling Biases in IP Topology Measurements. In *Proceedings of IEEE INFOCOM*.
[25] Matthew Luckie and Robert Beverly. 2017. The Impact of Router Outages on the AS-level Internet. In *Proceedings of ACM SIGCOMM*.
[26] Matthew Luckie and kc claffy. 2014. A Second Look at Detecting Third-Party Addresses in Traceroute Traces with the IP Timestamp Option. In *PAM*. 46–55.
[27] Matthew Luckie, Amogh Dhamdhere, Bradley Huffaker, David Clark, and kc claffy. 2016. bdrmap: Inference of Borders Between IP Networks. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*.
[28] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and kc claffy. 2013. AS Relationships, Customer Cones, and Validation. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*.
[29] Z Morley Mao, David Johnson, Jennifer Rexford, Jia Wang, and Randy Katz. 2004. Scalable and Accurate Identification of AS-Level Forwarding Paths. In *Proceedings of IEEE INFOCOM*.
[30] Zhuoqing Morley Mao, Jennifer Rexford, Jia Wang, and Randy H Katz. 2003. Towards an Accurate AS-Level Traceroute Tool. In *Proceedings of ACM SIGCOMM*.
[31] Alexander Marder and Jonathan M Smith. 2016. MAP-IT: Multipass Accurate Passive Inferences From Traceroute. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*.
[32] Measurement Lab Consortium. 2014. ISP Interconnection and its Impact on Consumer Internet Performance - A Measurement Lab Consortium Technical Report. http://www.measurementlab.net/publications/.
[33] Lin Quan, John Heidemann, and Yuri Pradkin. 2013. Trinocular: Understanding Internet Reliability Through Adaptive Probing. In *Proceedings of ACM SIGCOMM*.
[34] Chris Ritzo. 2018. Paris Traceroute has a bug, and it causes some bad data. https://www.measurementlab.net/blog/pt-bug/.
[35] Mario Sanchez, Fabian Bustamante, Balachander Krishnamurthy, Walter Willinger, Georgios Smaragdakis, and Jeffrey Erman. 2014. Inter-Domain Traffic Estimation for the Outsider. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*.
[36] Aaron Schulman and Neil Spring. 2011. Pingin' in the rain. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*.
[37] S. Sundaresan, A. Dhamdhere, M. Allman, and k. claffy. 2017. TCP Congestion Signatures. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*.
[38] Srikanth Sundaresan, Danny Lee, Xiaohong Deng, Yun Feng, and Amogh Dhamdhere. 2017. Challenges in Inferring Internet Congestion Using Throughput Measurements. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*.
[39] Fabien Viger, Brice Augustin, Xavier Cuvellier, Clémence Magnien, Matthieu Latapy, Timur Friedman, and Renata Teixeira. 2008. Detection, Understanding, and Prevention of Traceroute Measurement Artifacts. *Computer Networks* 52, 5 (2008), 998–1018.
[40] Yu Zhang, Ricardo Oliveira, Hongli Zhang, and Lixia Zhang. 2010. Quantifying the Pitfalls of Traceroute in AS Connectivity Inference. In *Proceedings of the Passive and Active Measurement Conference (PAM)*.