# Inferring Cloud Interconnections: Validation, Geolocation, and Routing Behavior

Alexander Marder[1], kc claffy[1], and Alex C. Snoeren[2]

[1] CAIDA / UC San Diego
[2] UC San Diego

**Abstract.** Public clouds fundamentally changed the Internet landscape, centralizing traffic generation in a handful of networks. Internet performance, robustness, and public policy analyses struggle to properly reflect this centralization, largely because public collections of BGP and traceroute reveal a small portion of cloud connectivity.
This paper evaluates and improves our ability to infer cloud connectivity, bootstrapping future measurements and analyses that more accurately reflect the cloud-centric Internet. We also provide a technique for identifying the interconnections that clouds use to reach destinations around the world, allowing edge networks and enterprises to understand how clouds reach them via their public WAN. Finally, we present two techniques for geolocating the interconnections between cloud networks at the city level that can inform assessments of their resilience to link failures and help enterprises build multi-cloud applications and services.

## 1 Introduction

The growing deployment of low-latency and high-throughput applications, the upfront and maintenance costs of computing resources, and constantly evolving security threats make it increasingly complex and costly for organizations to host services and applications themselves. Public cloud providers ease that burden by allowing organizations to build and scale their applications on networks and hardware managed by the cloud provider. At the core of cloud computing are virtual machines (VMs) and containers that run on physical hardware in a data center [47]. Clouds locate these data centers in globally distributed geographic regions [7, 8, 12]. The three major cloud providers, Amazon AWS, Microsoft Azure, and Google Cloud Platform (GCP), interconnect their regions using global backbones [6, 9, 52].

Public clouds fundamentally changed the Internet landscape from peer-to-peer to a cloud-centric model. According to a recent estimate [49], the ten highest-paying customers in AWS—all popular video and content generators—combine to spend over $100 million per month, and many enterprises store operations data and host internal applications in public clouds. Existing measurement platforms, with vantage points (VPs) located outside cloud networks, capture only a small fraction of the paths that connect public clouds to end users and

enterprises, and the importance of the clouds necessitates that the Internet measurement community considers how to effectively capture this.

The goal of this paper is to evaluate and improve our ability to infer cloud connectivity, in the hope that it bootstraps Internet measurements and analyses that more accurately reflect the cloud-centric Internet. We also build on those inferences, identifying the interconnections that clouds use to reach destinations around the world. Such analysis enables edge networks and enterprises to understand how clouds reach them, and potentially respond to fallout from congestion on a cloud interconnection. Furthermore, we geolocate the interconnections between cloud networks at the city level, providing techniques that can inform assessments of their resilience to link failures and help enterprises build multi-cloud applications and services. We make the following contributions:

1. We validate the state-of-the-art in identifying network interconnections (bdrmapIT) on Azure, identifying path changes as a prominent source of error.

2. We demonstrate that changing the traceroute probing method to reduce the number of simultaneous traceroutes reduces the impact of path changes on the observed topology, and improves the accuracy of bdrmapIT's AS operator inferences for the interconnection addresses in our validation dataset by 8.6%.

3. We use traceroute to identify next-hop ASes for each Internet network from AWS, Azure, and GCP, finding that clouds still rely on tier 1 and tier 2 networks, and that next-hop ASes can be region-dependent.

4. We geolocate all observed AWS-Azure and Azure-GCP interconnections, and 34.4% of the AWS-GCP interconnections, discovering that clouds interconnect on every populated continent, and often interconnect in the same cities.

## 2 Background and Previous Work

Our work builds on prior work that inferred AS-links from BGP, identified network interconnections in traceroute paths, studied cloud backbone networks with traceroute, and geolocated network infrastructure.

**BGP Route Announcements Reveal AS Connectivity:** The public BGP route announcement collectors, Routeviews [4] and RIPE RIS [3], collect announcements received from the ASes that peer with the collectors (VP ASes), and researchers infer AS connectivity from adjacent ASes in collected AS paths [16, 24]. We could infer cloud neighbors directly from the cloud networks through routes they propagate to public collectors, but cloud networks share few routes with public route collectors. We can also infer cloud connectivity indirectly from announcements that clouds originate into BGP, but VP ASes are unlikely to see cloud neighbors that enter into paid or settlement-free peering with the cloud [19, 25, 27, 32, 37, 55, 58]. Furthermore, VP ASes typically only propagate their chosen best-path for each prefix to collectors, and any VP AS that interconnects with a cloud network will likely select their direct interconnection as the best path to that cloud, and will not propagate alternate AS paths to the public collectors.

**Inferring Router Ownership From Traceroute Paths:** Substantial prior work attempted to infer AS interconnections from traceroute paths. Mao *et al.* [39, 40] aligned traceroutes from VP ASes with BGP route announcements seen by that same AS to better determine address space ownership. Chen *et al.* [18] generalized and expanded Mao's methodology to align AS-level links seen in traceroute with those in BGP AS paths. Later work focused on inferring the AS operators of routers in traceroute paths. Huffaker *et al.* [30] used alias resolution to convert the IP address paths in traceroute to router graphs, and proposed and validated four techniques to map routers to AS operators. Marder *et al.* [44] and Luckie *et al.* [36] independently developed and validated heuristics to extract constraints from traceroute to more accurately infer AS operators. Marder and Luckie later integrated and extended their approaches, creating the current state-of-the-art bdrmapIT, and validated their bdrmapIT technique [43]. Most recently, Luckie *et al.* [38] used the AS operator inferences from Huffaker *et al.*'s technique and bdrmapIT as training data to learn regular expressions for extracting AS operators embedded in hostnames in the form of AS numbers.

**Revealing Cloud Connectivity With Traceroute:** VPs outside the cloud cannot reveal many of the paths and interconnections that clouds use to reach the Internet. Yeganeh *et al.* [56] conducted traceroutes from AWS to every /24 to reveal interconnected networks, using a new unvalidated approach to infer network interconnections. In subsequent work [57], they compared the quality of service of default interconnections between cloud networks and third-party transit between clouds, switching to bdrmapIT to perform interconnection IP addresses inferences. Arnold *et al.* [15] inferred directly connected networks from traceroute paths by converting traceroute IP addresses to ASes using longest-matching prefix in BGP route announcements and IXP participant IP addresses recorded in PeeringDB [2]. They then augmented the AS-level connectivity graph in CAIDA's AS Relationship dataset [1] with peer relationships between each cloud and the newly inferred neighbors, using the graph to estimate that clouds can avoid their transit providers listed in CAIDA's AS Relationship dataset to reach 76% of the Internet networks. They validated their neighbor inferences with feedback from Azure and GCP, with 11%–15% false neighbor inferences. Assuming nearly perfect accuracy for IXP participant addresses in PeeringDB, these false neighbor inferences almost entirely result from false private interconnection inferences.

We show that the traceroute technique used by prior studies is prone to path change corruptions, and we validate our cloud interconnection inferences (§4). Rather than use unvalidated AS interconnection inference techniques, we use the previously validated bdrmapIT tool to infer private interconnections between cloud public WANs and their neighbors, and perform additional validation to understand bdrmapIT's accuracy for cloud networks. Finally, while Arnold *et al.* speculated how clouds *could* reach other ASes [15], we report how clouds currently *do* reach other networks.

**Geolocating Network Infrastructure:** Commercial IP geolocation databases focus nearly exclusively on edge hosts, with poor accuracy for network in-

frastructure [22, 26, 48]. Some networks encode geographic information in router interface DNS hostnames, but the geographic codes are difficult to automatically extract and interpret, as they use a mix of IATA codes, CLLI codes, and common location abbreviations. Rocketfuel includes the undns tool [51] that uses hand-crafted regular expressions to extract geolocations from hostnames. More general approaches avoid manually constructing regular expressions. DRoP [31] automatically learns rules to extract geolocation codes from hostnames, and HLOC [50] searches hostnames for geolocation codes. Other approaches use RTT to approximate distance between VPs and routers. Gueye *et al.* [28] and RIPE IPMap [45] triangulate RTTs to estimate location, and Katz-Basset *et al.* [33] refined RTT-based estimates using topology constraints. We use a combination of geolocation codes extracted from Azure DNS hostnames and traceroute paths to geolocate the interconnections between cloud providers.

## 3 Validating bdrmapIT With Azure Hostnames

Our analysis relies on bdrmapIT AS operator inferences to identify cloud interconnections and neighbors, so we first validate bdrmapIT's inferences on Azure to gain confidence in its efficacy and look for opportunities to improve our techniques. bdrmapIT addresses the difficult problem of inferring the networks that operate each router observed in traceroute, but relies on general assumptions of router configurations, internal traffic engineering, and network topology that might not hold in cloud WANs. Furthermore, prior bdrmapIT evaluations on transit interconnection inferences might not translate to cloud interconnection inferences. Initial bdrmapIT evaluations used CAIDA's Ark traceroutes and ground truth from ISP operators, and later experiments also validated bdrmapIT against pseudo ground truth derived from ISP DNS hostnames [38,41,42]. Traceroutes from CAIDA's Ark VPs mostly reveal transit interconnections—those between providers and customers—so transit interconnections dominate their reported accuracy. Clouds primarily peer with other networks, and we expect that their peering interconnections vastly outnumber their transit interconnections. Importantly, bdrmapIT leverages the industry convention that transit providers supply the IP subnets for interconnection with customers, but no known convention exists for peering interconnections [35]. To date, no study has evaluated bdrmapIT's accuracy using traceroutes that originate in the cloud.

For this initial experiment, we created a VM in every Azure region and used Scamper [34], the traceroute tool used in prior cloud studies [15, 56, 57], to conduct traceroutes from every VM to each of the 11.5 M /24s covered by a prefix in a BGP route announcement collected by RouteViews or RIPE RIS over 1-5 August 2020. Our choice of /24 granularity reflects our assumption that clouds are unlikely to receive many prefixes longer than /24. Each traceroute to a /24 targeted a random address to provide comprehensive coverage of Azure's neighboring networks, and we instructed Scamper to use Paris-style traceroute probes to prevent load-balancing from corrupting the traceroute paths.

To identify interconnection addresses between clouds and their neighbors, we used a combination of bdrmapIT AS operator inferences and IXP participant IP addresses listed in PeeringDB [2] and IXPDB [23] to map traceroute path IP addresses to ASes. In the event of a conflict between PeeringDB and IXPDB, a contact at both IXPDB and PeeringDB advised us to use the mapping in IXPDB, since IXP operators update information in IXPDB while IXP members populate information in PeeringDB, potentially causing stale entries. bdrmapIT requires AS address spaces as input, and we supplied prefix origin ASes derived from BGP announcements collected by RouteViews and RIPE RIS. For addresses with no covering prefix in BGP, we relied on the potentially stale longest matching prefix in RIR extended delegations. 0.8% of addresses did not have a covering prefix in BGP or RIR. We used whois [20] and RADb [46] to determine ownership for 53.4% of those addresses; this was the only manual step in this process.

We used Azure DNS hostnames to provide pseudo ground truth for our interconnection inferences, and successfully resolved hostnames for 59.5% of the 5749 Azure IP addresses seen in our initial traceroutes. Azure tags many of its network interconnection address hostnames with the name of the neighboring network, and we used the tags visible in traceroute paths from Azure VMs to identify Azure addresses on routers operated by neighbors; e.g., in a traceroute starting from an Azure VM, the tag in `internet2.dal-96cbe-1b.ntwk.msn.net` indicates that `104.44.12.159` belongs to an Internet2 router interconnected with Azure. Our evaluation focused on comparing bdrmapIT inferences to the tags extracted from Azure hostnames. We used the regular expression `([^-]*?)\..*\.ntwk\.msn\.net` to extract the interconnection tags from Azure hostnames, finding 214 tags corresponding to 419 address hostnames. For each IP address with a hostname containing an interconnection tag, we manually validated that bdrmapIT's AS operator inference aligns with the name of the inferred AS or the organization that owns it. These tags are nearly always network names rather than AS numbers, preventing us from using Luckie, *et al.*'s technique [38] to identify the operating AS automatically.

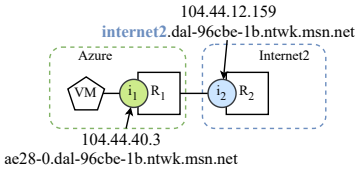



Fig. 1: In traceroute paths from Azure, the `internet2` tag indicates that `104.44.12.159` belongs to a router operated by Internet2. We use this as validation for bdrmapIT's router operator inferences from Azure traceroutes.

### 3.1 Investigating AS Operator Inference Errors

Our initial evaluation on Azure interconnections yielded 87.4% AS operator accuracy, with 53 errors. One source of error was that bdrmapIT occasionally filtered out valid neighboring ASes in favor of ASes seen adjacent to Azure in BGP AS paths. bdrmapIT relies heavily on AS connectivity inferred from BGP to constrain the choice of AS operator for a router, but the largely incomplete

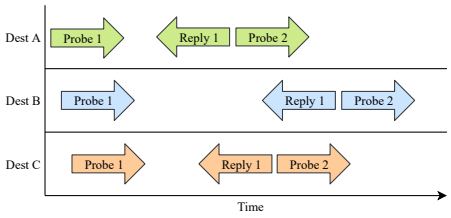



Fig. 2: Scamper increases efficiency by parallelizing traceroute probing across destinations, but a path change can corrupt all active traceroute paths.

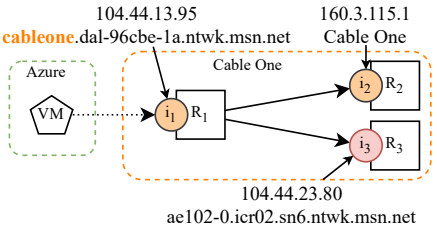



Fig. 3: We observed the Azure address `104.44.23.80` after the border router in Cable One, likely indicating traceroute path corruption. This caused bdrmapIT to incorrectly conclude that Azure operates router $R_1$.

connectivity constraints led to six false inferences in our validation set. We modified bdrmapIT to remove these constraints only for the major cloud networks, but this change can apply to edge networks with largely incomplete neighbor constraints in BGP AS paths, like other cloud and content delivery networks. This change increased the AS operator inference accuracy to 88.8%, correcting all six of the AS operator inferences without introducing additional error.

Using an interface graph constructed from the traceroutes to investigate the remaining errors led us to conclude that path changes during traceroutes likely caused most of the errors. While Paris probes avoid corruptions due to load-balancing along a path, they cannot prevent corruptions due to path changes in router forwarding tables. Scamper probing is especially susceptible to corruptions caused by path changes. Like UNIX traceroute, Scamper waits for the response to the probe with Time to Live (TTL) $i$ before sending the probe with TTL $i+1$, but for efficiency it parallelizes across traceroute destinations (Fig. 2). This concurrency enables rapid path discovery, necessary for temporally coherent snapshots of cloud topologies, but a path change can corrupt any of the traceroutes active at any given time.

To look for evidence of potential path changes, we generated a directed interface graph from the 355.8 M Azure traceroutes, creating directed edges between an address and every address that immediately followed it in a traceroute, but not when one or more unresponsive hops separate the addresses. We found 56 (13.4%) Azure interconnection addresses in our validation dataset followed by at least one Azure address in a traceroute. These interconnection addresses are on routers operated by neighboring networks, so an uncorrupted traceroute path would most likely not contain a subsequent Azure address. Fig. 3 shows a potentially corrupted traceroute path, where we observed an Azure IP address following the interconnection with Cable One. Observing Azure addresses after routers in neighboring ASes does not necessarily indicate that path changes corrupted

a traceroute, and can result from off-path addresses and load-balancing as well, so we conduct an additional experiment to rule out alternative explanations.

### 3.2 Fast And Straight Traceroute (FAST) Traceroute Probing

We developed a new traceroute tool, FAST, to test our hypothesis that path changes corrupted the cloud traceroute paths by mitigating the impact of path changes on the observed topology. FAST sends all probes from TTL 1 to 32 to a destination at a fixed packets per second (pps) rate, irrespective of replies, before moving on to the next traceroute destination, and uses packet capture to record probes and replies, allowing it to construct traceroute paths with accurate RTTs. Unlike similar tools such as Yarrp [17], FAST's guaranteed sequential probing allows it to construct traceroute paths during probing while consuming few resources on the cloud VMs.

| | AWS | Azure | GCP |
|---|---|---|---|
| Regions | 20 | 32 | 21 |
| VM Type | t3a-small | B2s | e2-micro |
| vCPUs | 2 | 2 | 2 |
| Memory | 2 GB | 4 GB | 1 GB |

Fig. 4: We set up VMs in every cloud region available to us using similar VM types in each cloud.

To efficiently reveal traceroute paths, we determined a probing rate for FAST that balances topology discovery with probing speed by conducting traceroutes from a VM in every region of AWS, Azure, and GCP (Fig. 4) to one address in 100,000 distinct prefixes announced into BGP. Our results (Fig. 5) indicate that probing at 5000 pps reveals nearly all of the hops found by probing at slower rates, but probing faster induced rate-limiting in Azure. At 5000 pps, FAST can complete probing to every routed /24 in less than 21 hours.

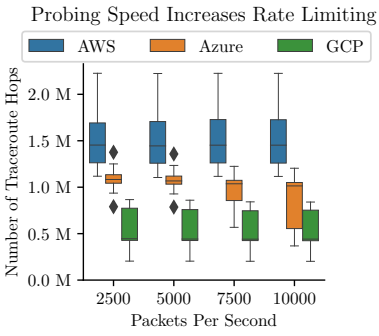



Fig. 5: We observed fewer traceroute hops for Azure probing above 5000 pps. GCP inflates probe TTLs (§4.1), causing relatively few responses for all probing rates.

To isolate the impact of path changes, we changed only the traceroute tool from Scamper to FAST, but conducted traceroutes from the same Azure regions to the same destinations. Generating an interface-graph from the new set of traceroutes appears to confirm our hypothesis that path changes corrupted the scamper traceroutes. In the FAST traceroutes, we never observed an Azure address after a router known to belong to a neighboring AS. Furthermore, path changes played a large role in bdrmapIT's inaccurate AS operator inferences. bdrmapIT's inferences on the FAST traceroutes were 97.4% accurate, compared to 88.8% with the Scamper traceroutes.

| Dest: 158.130.69.163 | | |
|---|---|---|
| 1 | 128.91.238.217 | [UPenn] |
| 2 | 128.91.48.6 | [UPenn] |

(a) Los Angeles to UPenn.

| Dest: 146.97.33.5 | | |
|---|---|---|
| 1 | 216.239.59.1 | [GCP] |
| 2 | 172.253.65.167 | [GCP] |
| 3 | 209.85.143.66 | [GCP] |
| 4 | 108.170.246.168 | [GCP] |
| 5 | * | |
| 6 | 146.97.33.62 | [JANET] |
| 7 | 146.97.33.5 | [JANET] |

(b) Los Angeles to JANET.

| Dest: 158.130.69.163 | | |
|---|---|---|
| 1 | 209.85.253.197 | [GCP] |
| 2 | 172.253.65.176 | [GCP] |
| 3 | 108.170.227.150 | [GCP] |
| 4 | 108.170.248.11 | [GCP] |
| 5 | 162.252.69.196 | [I2] |
| 6 | * | |
| 7 | 128.91.238.218 | [I2] |
| 8 | 128.91.238.217 | [UPenn] |
| 9 | 128.91.48.6 | [UPenn] |

(c) Belgium to UPenn.

Fig. 6: A traceroute from GCP Los Angeles to the University of Pennsylvania (UPenn) revealed no GCP IP addresses (a), but traceroutes from Los Angeles to JANET in the UK (b), and Belgium to UPenn (c), each revealed GCP addresses.

## 4 Learning About Clouds From Interconnections

Armed with confidence in our interconnection inferences, we set up VMs in every region available to us for AWS, Azure, and GCP, the three largest cloud providers, and used FAST to conduct traceroutes from every VM to a random address in every routed /24. We configured our VMs to use the WAN as much as possible; Azure networking defaults to cold-potato routing and we selected GCP's premium network tier, but in AWS we used the default WAN behavior. We did not use AWS Global Accelerator [5], and we plan to investigate the affect of Global Accelerator in future work. These experiments derived routed address space using collected BGP route announcements from 1-5 October 2020. We used the same combination of bdrmapIT, PeeringDB, and IXPDB as in §3 to infer interconnection addresses, and used these interconnection inferences to analyze the neighboring networks that each cloud uses to reach public Internet networks, and to geolocate the interconnections between the three cloud providers.

### 4.1 GCP Inflates Traceroute Probe TTLs

One challenge for our analysis is that GCP, unlike AWS and Azure, inflates the TTL values of traceroute probes after they leave VMs such that the hop #1 traceroute address belongs to a later router in that path, rather than to the first router hop [13][3]. This behavior violates a core traceroute assumption that hop #1 corresponds to the first router probed. While invisible Multiprotocol Label Switching (MPLS) tunnels exhibit similar behavior, hiding router hops between the tunnel entry and exit routers [21,53,54], MPLS tunnels do not affect hop #1 since the probe with TTL 1 could not yet enter an MPLS tunnel. This practice of rewriting probe TTLs has likely caused researchers to incorrectly conclude that GCP routers do not respond to traceroute [29], or that hop #1 is a router just past the GCP border [57].

[3] We observed different behavior in February, 2021 (§A).

Fig. 6a shows the GCP TTL inflation with a traceroute from a VM in Los Angeles, where hop #1 reported an address that router configurations from Internet2 show belong to a University of Pennsylvania (UPenn) router [10], despite no direct interconnection between GCP and UPenn [11]. In fact, the UPenn router at hop #1 reported an interface address used to interconnect with Internet2 [10], indicating that the probes traversed Internet2 to reach UPenn, but the inflated TTL caused probes to expire only after reaching UPenn. Traceroutes from other GCP VMs to the same UPenn destination, such as in the Belgium region, exposed apparent GCP internal IP addresses, only reaching UPenn at hop 8 (Fig. 6c). All of our VMs use GCP's premium network tier, but not all revealed internal GCP addresses, contradicting reported behavior that only GCP's standard tier inflates traceroute TTLs [14]. Our ability to observe internal GCP addresses from the Belgium VM toward UPenn, and from the VM in Los Angeles toward JANET in the UK (Fig. 6b), suggests that the opportunity to view internal and interconnection GCP addresses depends on the combination of GCP region and traceroute destination. We leave an analysis of the interconnection information lost to GCP's TTL inflation for future work.

### 4.2 Inferring How Clouds Reach Internet Networks

We define the *cloud transit degree* for a cloud neighbor AS as the number of unique traceroute destination ASes for which the neighbor is the next-hop AS. This metric is an indication of the relative importance of that neighbor to the cloud network. In Fig. 7, the cloud network uses AS #1 to reach three ASes including AS #1, giving it a CTD of 3, while AS #2 has a CTD of 2. Here, the cloud uses both AS #1 and AS #2 to reach AS #3, so we count AS #3 once for each AS. This situation occurs when clouds choose different next-hop networks depending on the VM's region.

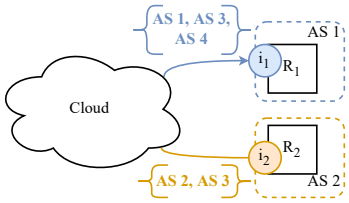



Fig. 7: AS #1 is the next-hop network in traceroute paths to three ASes, so its cloud transit degree (CTD)=3. AS #2 is the next-hop network for two ASes, so its CTD=2.

We only used traceroutes with a cloud interconnection in the path to compute the CTDs, so we discarded any traceroute where an unresponsive hop separates the last hop inside the cloud network from the first hop outside the cloud network. Fig. 8a shows the fraction of included traceroutes in each cloud. For every neighbor AS, we maintain a set of destination ASes reached through that neighbor, so at the first hop in the neighbor AS we add the traceroute's destination AS to that neighbor's set. Finally, we compute the CTD for each neighbor as the cardinality of its destination set.

Fig. 8b shows the number of unique ASes for each cloud across their different regions. The different variances reflect the traffic engineering policies of each cloud. AWS uses hot-potato routing, so we not only saw different neighbors from each region, but we saw different numbers of neighboring ASes as well. Conversely, Azure uses cold-potato routing, so Azure transits packets destined

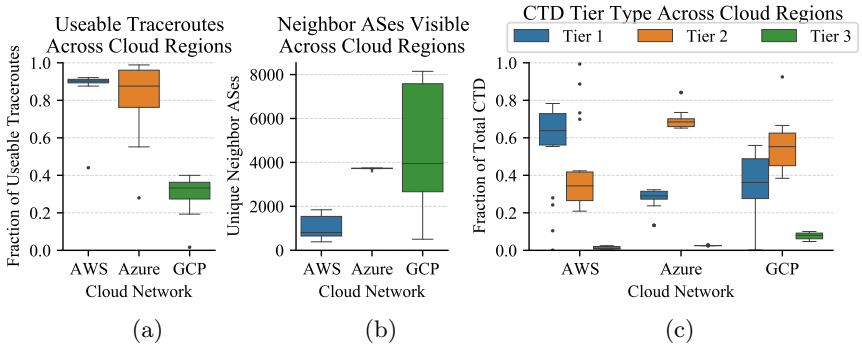



Fig. 8: We excluded many of the GCP traceroutes since the traceroute path often began outside GCP (a). Unlike AWS and GCP, we observed nearly all of the same neighbor ASes from every Azure region (b). All clouds rely on tier 1 and tier 2 networks, but AWS relies more heavily on tier 1s in most regions (c).

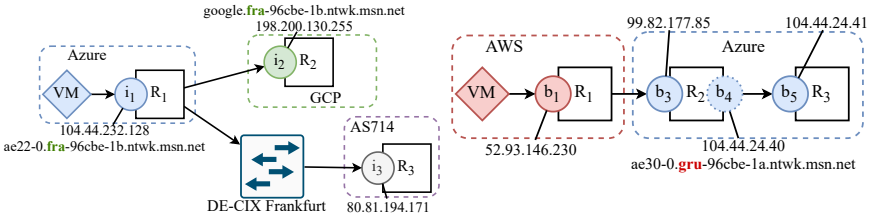
for a neighboring AS across its global backbone and hands them off to the neighbor directly. GCP also employs cold-potato routing, but certain regions included more internal routers in traceroute paths than others. We only included an AS as a neighbor when we saw a GCP interconnection address, as traceroute paths can start in unconnected networks (Fig. 6).

Fig. 8c shows the fraction of the total CTD accounted for by tier 1, tier 2, and tier 3 networks. For the purposes of this analysis, we define tier 1 networks as the 19 ASes inferred to be at the top of the AS hierarchy in CAIDA's AS relationship dataset for October 1, 2020. Tier 2 networks include the 10,627 other ASes with at least one customer in the dataset, with the remaining networks classified as tier 3. Our analysis reveals that all three clouds rely heavily on ISPs, although we expect that the clouds primarily peer with these ISPs, rather than interconnect with them for Internet transit. AWS shows wide variance across regions, heavier reliance on tier 1 networks (due to hot potato routing), and heavy tier 2 network use in certain regions. Azure relies on tier 1 and tier 2 networks consistently across regions, and GCP appears better connected to edge networks.

In total, we discovered an order of magnitude more cloud neighbor ASes in our traceroutes from cloud VMs than were visible in RouteViews and RIPE RIS collections of BGP route announcements from 1-5 October, 2020 (Table 1). We also found that GCP appears to interconnect with more than twice as many networks as AWS and Azure. Importantly, our results indicate that the visible connectivity of cloud networks, and their reliance on specific neighbors, is region-dependent. To properly measure and analyze the cloud requires gathering data from each region, and considering each region separately.

| | AWS | Azure | GCP |
|---|---|---|---|
| Traceroute | 4110 | 3889 | 8620 |
| BGP | 327 | 300 | 381 |

Table 1: We observed an order of magnitude more unique cloud AS neighbors traceroutes than in public BGP collections.




(a) IXP and hostname geolocation. (b) Hostname geolocation.

Fig. 9: In (a) the interconnection address $i_2$ and the IXP address $i_3$ share a common predecessor, so we infer $i_2$ is also located in Frankfurt. The hostnames for $i_2$ and $i_1$ also indicate Frankfurt. In (b) we use `gru` in the hostname for $b_4$ to infer that the interconnection occurs in Sao Paulo.

### 4.3 Geolocating Cloud Interconnections

Next, we use IXP location constraints and geolocation tags in Azure hostnames to infer the locations of interconnections between the clouds. For IXP constraints, we identify all addresses that preceded an interconnection address in a traceroute that also preceded an IXP address, and infer that the interconnection is located at the IXP location recorded in PeeringDB. Our reasoning follows from the fact that interconnected routers operated by two different networks are often located in the same facility or city. In Fig. 9a, bdrmapIT inferred that address `198.200.130.255` interconnects Azure and GCP, and the prior address `104.44.232.128` also preceded an address used for public peering at DE-CIX Frankfurt in a different traceroute, so we conclude that the interconnection using address `198.200.130.255` occurs in Frankfurt. Remote peering at IXPs, where a network participates at multiple IXPs through a port at a single IXP, creates the possibility that our method could identify multiple IXPs. We expect to more often observe local IXP peering than remote peering, so in the event our technique identified multiple IXPs, we select the most frequently appearing IXP city location.

We also used geographic locations embedded in Azure IP hostnames to geolocate interconnections involving Azure, such as the reference to Frankfurt in the hostname for `198.200.130.255`, `google.fra-96cbe-1b.ntwk.msn.net`. We collected hostnames for three groups of addresses most likely to reside in the same city as the interconnection: (1) the interconnection address, (2) addresses that precede the interconnection address, and (3) the other address in the /31 subnet of addresses subsequent to the interconnection address. For the latter group, while the subsequent addresses might not reside in the interconnection city, we assume the other address in the point-to-point subnet likely belongs to the same router as the interconnection address. In Fig. 9b, we use the hostname for `104.44.24.40` which we infer belongs to the same router as the interconnection address `99.82.177.85`, despite not appearing in the traceroutes. Using a handcrafted regular expression, we extracted the geolocation codes, and mapped each

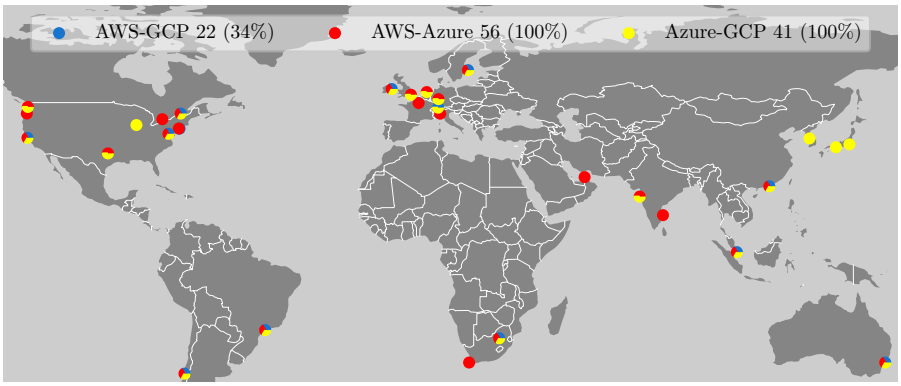



Fig. 10: Unique city locations for interconnections between the clouds. The clouds often interconnect in the same cities, indicated with pie-chart markers. We could only geolocate 22 of the 64 inferred AWS-GCP interconnections (34.4%).

code to a city. This technique always inferred a single city for each interconnection address. Using both techniques allowed us to infer geolocations for every Azure link, and the two techniques never inferred conflicting cities.

Our two geolocation strategies yielded the city locations in Fig. 10. We found that the clouds interconnect in all six populated continents and tend to interconnect in the same locations. The IXP constraints provided locations for 87.8% and 80.4% of the Azure-GCP and Azure-AWS interconnections respectively, but only geolocated 34.4% of the AWS–GCP interconnections, since many of their interconnection IP addresses did not share a common predecessor with an IXP address. Interconnections visible from AWS VMs rarely shared a predecessor with an IXP address, and the GCP traceroutes often lacked internal hops. The congruity between the two techniques indicates that these techniques can accurately geolocate many of the cloud interconnections visible in traceroute.

## 5 Limitiations

Our analysis performs inferences on top of inferences, and an error at any step can lead to false conclusions. Acknowledging the potential for compounding error, we validated as many of our interconnection and geolocation inferences as possible. In general, we expect our analysis to reflect the reality revealed by our traceroutes, despite imperfect accuracy.

One limitation of our validation is that we rely on the accuracy of Azure's DNS hostnames, but operators might not update them when an IP address switches from one router interface to another. While this might apply to our case, the 97.4% congruity between bdrmapIT AS operator inferences and the

hostname tags suggests that Azure maintains its hostnames well. The coverage of our validation is a more fundamental limitation, as our validation dataset covers a single cloud network and only 30.3% of the total number of bdrmapIT inferred Azure private interconnection IP addresses. Our reported bdrmapIT AS operator accuracy might not generalize to the other Azure interconnections that bdrmapIT inferred, let alone to AWS and GCP interconnections.

When inferring cloud neighbors, our traceroutes might not reveal all of the interconnections between the cloud networks, and between the clouds and neighboring networks. In GCP, most traceroute paths either start outside of GCP or the VMs do not receive a reply from the apparent interconnection router hop (Fig. 8a). Specific to AWS, its hot-potato routing means that traffic to a connected AS might leave the WAN at a different neighbor closer to the the VM. For all three clouds, a traceroute only reveals a single active path, and our probing might miss alternate active paths. Furthermore, our probing can only reveal networks interconnected with cloud public WANs, but some networks interconnect with clouds in a more private fashion.

Yeganeh *et al.* [56] described cloud exchanges as multipoint-to-point interconnections that use address space provided by the exchange operator, and speculated that bdrmapIT cannot draw accurate AS operator inferences for routers at cloud exchanges. We do not expect cloud exchanges to pose problems for bdrmapIT's AS operator inferences, since it determines AS ownership by looking forward from a router to addresses seen subsequently in a traceroute. This allows bdrmapIT to determine ownership for IXP public peering addresses, and it should perform similarly for the cloud exchanges Yeganeh *et al.* described. A potential consequence of cloud exchanges is that our methodology for inferring next-hop networks might select the cloud exchange provider as the next-hop AS if the exchange does not use address space belonging to a cloud or list its address space in PeeringDB or IXPDB.

## 6 Conclusion

Public clouds play a central role in the modern Internet, but we know little about how they interconnect to each other or other networks. Understanding cloud connectivity is critical to studying the modern Internet, including for network planning and diagnosis, and resiliency assessments. This study lays a foundation for future analyses by validating and improving a technique to infer cloud interconnection IP addresses. We analyzed next-hop ASes that the clouds use to reach other networks and proposed techniques to geolocate interconnections between the clouds. We found that clouds interconnect with each other on all six populated continents, and that next-hop ASes can be region-dependent, indicating that properly analyzing cloud networks requires measurements from every region. We will make FAST and the code for our analysis available at `https://alexmarder.github.io/cloud-pam21/`.

## Acknowledgments

This work was supported by DARPA CA HR00112020014, NSF OAC-1724853, NSF CNS-1901517, and NSF CNS-1925729.

## A Recent GCP Traceroute Behavior

| Dest: 158.130.69.163 | |
|---|---|
| 1 72.14.237.86 | [GCP] |
| 2 162.252.69.188 | [I2] |
| 3 * | |
| 4 * | |
| 5 128.91.238.218 | [I2] |
| 6 128.91.238.217 | [UPenn] |
| 7 128.91.48.6 | [UPenn] |

(a) Los Angeles to UPenn.

| Dest: 146.97.33.5 | |
|---|---|
| 1 * | [GCP] |
| 2 * | [GCP] |
| 3 * | [GCP] |
| 4 * | [GCP] |
| 5 128.91.238.217 | [UPenn] |
| 6 128.91.48.6 | [UPenn] |

(b) Virginia to UPenn.

Fig. 11: Unlike the traceroute in October, 2020, the traceroute from GCP Los Angeles to UPenn in February, 2021 revealed an internal GCP IP addresses (a). The first responsive hop in the traceroute from Virginia was an address on a UPenn router, but the path contained unresponsive hops until that point (b).

We conducted the traceroutes in §4.1 in October, 2020. Revisiting our examples in February, 2021, we noticed a different behavior. Many paths still do not contain any internal GCP addresses, but the paths no longer appear to start in neighboring networks. As seen in the traceroute path from GCP Los Angeles to UPenn (Fig. 11a), hop #1 is an internal GCP address followed by the interconnection with Internet2 at hop #2 [10], rather than a UPenn address. The first responsive hop in the path from our GCP Virginia VM (Fig. 11b) is the same UPenn address that we previously observed as hop #1 in §4.1, but hop #1 is now an unresponsive address. This behavior makes interpreting GCP traceroutes more intuitive, as they follow conventional traceroute semantics, but observing GCP internal addresses still appears to depend on the combination of VM region and traceroute destination.

## References

1. The CAIDA AS relationships dataset. https://www.caida.org/data/as-relationships/
2. PeeringDB. https://peeringdb.com/
3. Routing information service (RIS). https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris
4. University of Oregon route views project. http://www.routeviews.org/routeviews/
5. AWS global accelerator. https://aws.amazon.com/global-accelerator (Oct 2020)

6. Azure global network. https://azure.microsoft.com/en-us/global-infrastructure/global-network/ (May 2020)
7. Azure regions. https://azure.microsoft.com/en-us/global-infrastructure/regions/ (May 2020)
8. Cloud locations. https://cloud.google.com/about/locations (May 2020)
9. Global infrastructure. https://aws.amazon.com/about-aws/global-infrastructure/ (May 2020)
10. Internet2 - visible network raw data access. https://vn.net.internet2.edu/xml/Internet2/2020/10/15/14/43/show_interfaces.gz (Oct 2020)
11. Pennnet network architecture. https://upenn.app.box.com/v/RouterCoreDiagram (2020)
12. Regions and availability zones. https://aws.amazon.com/about-aws/global-infrastructure/regions_az/ (May 2020)
13. VPC network overview. https://cloud.google.com/vpc/docs/vpc (May 2020)
14. Arnold, T., Gürmeriçliler, E., Essig, G., Gupta, A., Calder, M., Giotsas, V., Katz-Bassett, E.: (how much) does a private wan improve cloud performance? In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. pp. 79–88. IEEE (2020)
15. Arnold, T., He, J., Jiang, W., Calder, M., Cunha, I., Giotsas, V., Katz-Basset, E.: Cloud provider connectivity in the flat internet. IMC (2020)
16. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. science **286**(5439), 509–512 (1999)
17. Beverly, R.: Yarrp'ing the internet: Randomized high-speed active topology discovery. In: IMC. pp. 413–420 (2016)
18. Chen, K., Choffnes, D.R., Potharaju, R., Chen, Y., Bustamante, F.E., Pei, D., Zhao, Y.: Where the sidewalk ends: Extending the internet as graph using traceroutes from p2p users. In: Proceedings of the 5th international conference on Emerging networking experiments and technologies. pp. 217–228 (2009)
19. Dimitropoulos, X., Krioukov, D., Fomenkov, M., Huffaker, B., Hyun, Y., Claffy, K., Riley, G.: As relationships: Inference and validation. ACM SIGCOMM Computer Communication Review **37**(1), 29–40 (2007)
20. d'Itri, M.: whois. https://github.com/rfc1036/whois
21. Donnet, B., Luckie, M., Mérindol, P., Pansiot, J.J.: Revealing MPLS tunnels obscured from traceroute. ACM SIGCOMM Computer Communication Review (2012)
22. Du, B., Candela, M., Huffaker, B., Snoeren, A.C., claffy, k.: RIPE IPmap active geolocation: Mechanism and performance evaluation. SIGCOMM Comput. Commun. Rev. **50**(2) (May 2020)
23. Euro-IX: Ixpdb. https://ixpdb.euro-ix.net/en/
24. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. ACM SIGCOMM computer communication review **29**(4), 251–262 (1999)
25. Gao, L.: On inferring autonomous system relationships in the internet. IEEE/ACM Transactions on networking **9**(6), 733–745 (2001)
26. Gharaibeh, M., Shah, A., Huffaker, B., Zhang, H., Ensafi, R., Papadopoulos, C.: A look at router geolocation in public and commercial databases. In: Proceedings of the 2017 Internet Measurement Conference. pp. 463–469 (2017)
27. Giotsas, V., Luckie, M., Huffaker, B., Claffy, K.: Inferring complex as relationships. In: Proceedings of the 2014 Conference on Internet Measurement Conference. pp. 23–30 (2014)

28. Gueye, B., Ziviani, A., Crovella, M., Fdida, S.: Constraint-based geolocation of internet hosts. IEEE/ACM Transactions On Networking **14**(6), 1219–1232 (2006)
29. Haq, O., Raja, M., Dogar, F.R.: Measuring and improving the reliability of wide-area cloud paths. In: WWW. pp. 253–262 (2017)
30. Huffaker, B., Dhamdhere, A., Fomenkov, M., et al.: Toward topology dualism: improving the accuracy of as annotations for routers. In: International Conference on Passive and Active Network Measurement. pp. 101–110. Springer (2010)
31. Huffaker, B., Fomenkov, M., Claffy, K.: Drop: Dns-based router positioning. ACM SIGCOMM Computer Communication Review **44**(3), 5–13 (2014)
32. Jin, Y., Scott, C., Dhamdhere, A., Giotsas, V., Krishnamurthy, A., Shenker, S.: Stable and practical AS relationship inference with ProbLink. In: 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). pp. 581–598 (2019)
33. Katz-Bassett, E., John, J.P., Krishnamurthy, A., Wetherall, D., Anderson, T., Chawathe, Y.: Towards ip geolocation using delay and topology measurements. In: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement. pp. 71–84 (2006)
34. Luckie, M.: Scamper: a scalable and extensible packet prober for active measurement of the Internet. In: IMC (2010)
35. Luckie, M., Dhamdhere, A., Clark, D., Huffaker, B., Claffy, K.: Challenges in inferring internet interdomain congestion. In: IMC (2014)
36. Luckie, M., Dhamdhere, A., Huffaker, B., Clark, D., claffy, k.: bdrmap: Inference of borders between IP networks. In: IMC (2016)
37. Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., Claffy, K.: As relationships, customer cones, and validation. In: Proceedings of the 2013 conference on Internet measurement conference. pp. 243–256 (2013)
38. Luckie, M., Marder, A., Fletcher, M., Huffaker, B., kc claffy: Learning to extract and use ASNs in hostnames. In: Proceedings of the 2020 Internet Measurement Conference (2020)
39. Mao, Z.M., Johnson, D., Rexford, J., Wang, J., Katz, R.: Scalable and accurate identification of AS-level forwarding paths. In: IEEE INFOCOM 2004. vol. 3, pp. 1605–1615. IEEE (2004)
40. Mao, Z.M., Rexford, J., Wang, J., Katz, R.H.: Towards an accurate AS-level traceroute tool. In: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications. pp. 365–378 (2003)
41. Marder, A.: Sharp Snapshots of the Internet's Graph with HONE. Ph.D. thesis, University of Pennsylvania (2019)
42. Marder, A.: vrfinder: Finding outbound addresses in traceroute. In: SIGMETRICS (2020)
43. Marder, A., Luckie, M., Dhamdhere, A., Huffaker, B., kc claffy, Smith, J.M.: Pushing the Boundaries with bdrmapIT: Mapping Router Ownership at Internet Scale. In: IMC (2018)
44. Marder, A., Smith, J.M.: MAP-IT: Multipass accurate passive inferences from traceroute. In: Proceedings of the 2016 Internet Measurement Conference. pp. 397–411. ACM (2016)
45. NCC, R.: RIPE IPMap. `https://ipmap.ripe.net/`
46. Network, M.: RADb: The Internet routing registry. `https://www.radb.net/`
47. Norton, W.B.: Cloud interconnections. `https://www.caida.org/workshops/wie/1612/slides/wie1612_wnorton.pdf` (Sep 2016)

48. Poese, I., Uhlig, S., Kaafar, M.A., Donnet, B., Gueye, B.: Ip geolocation databases: Unreliable? ACM SIGCOMM Computer Communication Review **41**(2), 53–56 (2011)
49. Saunders, B.: Who's using amazon web services? [2020 update]. https://www.contino.io/insights/whos-using-aws (Jan 2020)
50. Scheitle, Q., Gasser, O., Sattler, P., Carle, G.: Hloc: Hints-based geolocation leveraging multiple measurement frameworks. In: 2017 Network Traffic Measurement and Analysis Conference (TMA). pp. 1–9. IEEE (2017)
51. Spring, N., Mahajan, R., Wetherall, D.: Measuring isp topologies with rocketfuel. ACM SIGCOMM Computer Communication Review **32**(4), 133–145 (2002)
52. Taneja, S., Pretzer, X.: Google cloud networking in depth: Understanding network service tiers. https://cloud.google.com/blog/products/networking/google-cloud-networking-in-depth-understanding-network-service-tiers (May 2019)
53. Vanaubel, Y., Luttringer, J.R., Mérindol, P., Pansiot, J.J., Donnet, B.: Tnt, watch me explode: A light in the dark for revealing mpls tunnels. In: 2019 Network Traffic Measurement and Analysis Conference (TMA). pp. 65–72. IEEE (2019)
54. Vanaubel, Y., Mérindol, P., Pansiot, J.J., Donnet, B.: Through the wormhole: Tracking invisible MPLS tunnels. In: Proceedings of the 2017 Internet Measurement Conference. pp. 29–42. ACM (2017)
55. Xia, J., Gao, L.: On the evaluation of as relationship inferences [internet reachability/traffic flow applications]. In: IEEE Global Telecommunications Conference, 2004. GLOBECOM'04. vol. 3, pp. 1373–1377. IEEE (2004)
56. Yeganeh, B., Durairajan, R., Rejaie, R., Willinger, W.: How cloud traffic goes hiding: A study of Amazon's peering fabric. In: IMC. pp. 202–216 (2019)
57. Yeganeh, B., Durairajan, R., Rejaie, R., Willinger, W.: A first comparative characterization of multi-cloud connectivity in today's internet. In: PAM. pp. 193–210. Springer (2020)
58. Zhang, B., Liu, R., Massey, D., Zhang, L.: Collecting the internet as-level topology. ACM SIGCOMM Computer Communication Review **35**(1), 53–61 (2005)