Learning to Extract Geographic Information from Internet Router Hostnames

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

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

Alexander Marder
CAIDA, UC San Diego
amarder@caida.org

Zachary Bischof
CAIDA, UC San Diego
z@chary.io

Marianne Fletcher
University of Waikato
mfletche@wand.net.nz

k claffy
CAIDA, UC San Diego
kc@caida.org

ABSTRACT

Geolocating Internet routers is a long-standing and notoriously difficult challenge, and current solutions lack the accuracy and adaptability to yield reliable results. We revisit this problem, designing a solution capable of accurately and comprehensively extracting geographic information that network operators embed into router interface hostnames. We train our system using dictionaries that map geographic codes to known locations, and constrain inferences with delay measurements conducted from a distributed set of vantage points. While most operators use known geographic codes, some devise their own mnemonic codes for locations, which our system also extracts and interprets.

We evaluate our system on Internet-wide topology datasets, automatically learning regular expressions (regexes) for 1023 different domain suffixes with IPv4 routers, and 241 different domain suffixes with IPv6 routers. We received ground truth from operators of 13 domain suffixes, all of whom confirmed the correctness of our learned regexes, and that our system correctly interpreted 78.6% of the custom geographic codes. For these 13 suffixes, our solution more accurately extracts and interprets geographic information than the previous state-of-the-art, correctly geolocating 94.0% of router hostnames with a geohint compared to DRoP (56.6%) and HLOC (73.1%). This work advances the ability of researchers and network operators to characterize the location of critical Internet infrastructure, a foundational building block of network performance, security, and resilience analysis. We release the source code of our system and our inferred regexes.

ACM Reference Format:
Matthew Luckie, Bradley Huffaker, Alexander Marder, Zachary Bischof, Marianne Fletcher, and k claffy. 2021. Learning to Extract Geographic Information from Internet Router Hostnames. In *The 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '21), December 7–10, 2021, Virtual Event, Germany.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3485983.3494869

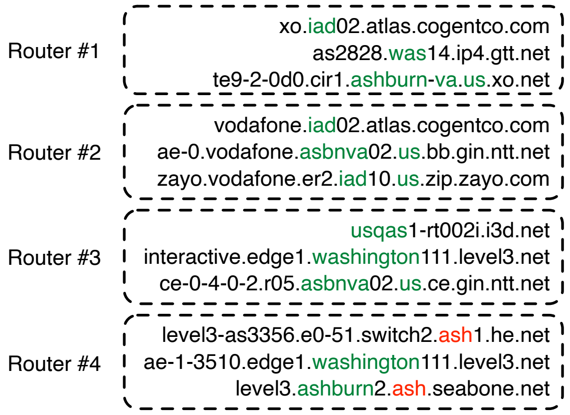

Router #1
xo.iad02.atlas.cogentco.com
as2828.was14.ip4.gtt.net
te9-2-0d0.cir1.ashburn-va.us.xo.net
Router #2
vodafone.iad02.atlas.cogentco.com
ae-0.vodafone.asbnva02.us.bb.gin.ntt.net
zayo.vodafone.er2.iad10.us.zip.zayo.com
Router #3
usqas1-rt002i.i3d.net
interactive.edge1.washington111.level3.net
ce-0-4-0-2.r05.asbnva02.us.ce.gin.ntt.net
Router #4
level3-as3356.e0-51.switch2.ash1.he.net
ae-1-3510.edge1.washington111.level3.net
level3.ashburn2.ash.seabone.net

Figure 1: Fundamental challenge of inferring router geolocation from hostnames: undocumented, inconsistent, and colliding use of various dictionaries. Routers may have hostnames with an IATA code ("iad", "was"), CLLI prefix ("asbnva"), LOCODE ("usqas"), or a city name ("ashburn", "washington"). Some operators also encode country ("us") and state ("va"). However, "ash" (router #4) is an IATA code for Nashua, NH, US that the operators of he.net and seabone.net used to label routers in Ashburn, VA, US.

1 INTRODUCTION

Identifying the physical location of Internet routers is critical for assessing the resilience of network infrastructure to natural disasters [41], the optimality of forwarding paths [19, 34], and the degree to which communications transit a country that performs censorship or surveillance [22]. The process of identifying the physical location of an Internet Protocol (IP) address, *geolocation*, is well-studied, both in academia and industry. Commercial geolocation databases are well-suited to identifying the physical location of end-host IP addresses, as there are a myriad of commercial applications that support those efforts, such as location-based advertising and digital rights management. However, these commercial databases cannot accurately geolocate Internet routers [15].

Some network operators encode the physical location of a router in Domain Name System (DNS) hostname strings because it helps them and other network operators understand and diagnose Internet performance problems [50]. Figure 1 provides examples of routers with geographic hints *(geohints)* embedded in hostnames that place the routers in Ashburn, VA, US. The operators use different conventions: gtt.net and zayo.com embed International Air

Transport Association (IATA) airport codes, ntt.net embeds Common Language Location Identifier (CLLI) prefixes, i3d.net embeds United Nations Code for Trade and Transport Locations (UN/LOCODEs), and level3.net embeds city names; ntt.net, zayo.com, and xo.net embed country codes, while xo.net also embeds state codes.

Beyond the different types of geohints, each operator independently decides where in the hostname they place the geohint, as well as any other information to embed in the hostname. The resulting diversity in conventions makes it difficult for researchers and network operators to reason about the physical location of Internet routers. A further challenge is operators that usually follow a convention, such as the IATA airport codes, but deviate from the defined codes with custom geohints.

Geolocation is so fundamental to analysis of networks, that researchers have spent two decades developing geolocation methods. We review the literature in §3, but four studies illustrate the depth and importance of the problem. In 2002, Spring *et al.* developed and released Rocketfuel [49], which included the undns tool and a *manually-assembled* database of regular expressions (regexes) that extracted geohints, among other information, from hostnames. Dozens of subsequent studies (§3.2) relied on this database to evaluate the performance, security, and robustness of networks, but its manual construction was unsustainable; its regexes have not been updated since 2014. In 2004, Gueye *et al.* published CBG, which builds distance constraints using delay measurements from vantage points with known locations to infer the approximate location of a system through multilateration on constraints. However, researchers have limited access to densely deployed measurement infrastructures that allow for Internet-scale geolocation. In 2014, Huffaker *et al.* published DRoP to automate learning geolocation regexes [20] using previously gathered delay measurements to constrain inference, but their design tradeoffs limited its scope and accuracy [6, 10, 46]. In 2017, Scheitle *et al.* published HLOC, which evaluated router hostnames for possible geohints using delay measurements collected at run-time, without using regexes [46]; their method relies on a dictionary of geohints and does not correctly interpret hostnames when operators use custom geohints.

In this paper, we take a fresh look at router geolocation, combining recent developments in learning other features in hostnames [30, 32] with delay measurements to automatically extract and interpret geohints embedded into hostnames using regexes. Using regexes to learn the structure of geohints in hostnames has at least three advantages over interpreting hostnames with delay measurements at run-time. First, regexes are available for others to use, who may not have access to a measurement infrastructure. Second, when an operator uses custom geohints, the string extracted by the regex is likely a geohint, providing a basis to infer a geolocation mapping for that string. Third, regexes can geolocate routers that do not respond to delay measurements if the network provides hostnames for those routers, and other routers that do respond to delay measurements. We make the following four contributions:

We design and implement a method that automatically learns regexes that extract geohints from hostnames. Our method, informed by a dictionary of airport codes, city names, LOCODEs, CLLI prefixes, facilities, and state and country codes, automatically learns regexes that extract these hints from hostnames. We assembled four sets of training data – IPv4 router-level topologies for August 2020 and March 2021, and IPv6 router-level topologies for November 2020 and March 2021. For the August 2020 IPv4 training data, 8.8% of 2.56M routers had hostnames with apparent geohints; our method learned 461 regexes that extracted IATA codes, 96 regexes that extracted CLLI prefixes, 372 regexes that extracted city names, and 10 regexes that extracted LOCODEs, covering 86.8% of these routers.

We design and implement a method that automatically learns when an operator deviates from geohint dictionaries. Our method learns three-letter codes that override the IATA dictionary (e.g., "ash" for "Ashburn, VA" instead of "Nashua, NH"), five-letter codes that override the LOCODE dictionary (e.g., "jptky" for "Tokyo, JP" instead of "Tokuyama, JP"), and six-letter codes that override the CLLI dictionary (e.g., "mancen" for "Manchester, UK"). We find extensive evidence that operators deviate from these dictionaries: of the 461 regexes that extracted apparent IATA airport codes, 147 (38.2%) included at least one hint that did not correspond to an IATA airport code serving the location.

We validate our method using ground truth from network operators. We received ground truth covering 13 networks of different classes and scale. The responses show that our inferred conventions captured the operators' intent to embed geohints in router hostnames, with 94.0% of router hostnames correctly interpreted. We validated our inferences of suffix-specific geohints (e.g., when an operator uses "ash" for "Ashburn, VA") and received confirmation that 92 of 117 (78.6%) of our learned geohints were correct for these 13 networks.

We publicly release the source code implementation and a website containing the inferred naming conventions. To promote use of our method, we publicly release our source code [28] and data [31]. Our website allows researchers to obtain the regexes, understand how they work, and accelerate the community's understanding of where network operators deploy their infrastructure.

2 GEOLOCATION HINT TYPES

Each operator independently decides what information they store in their PTR records, including the form of any geohint. Steenbergen found that the most commonly used types of geohints are IATA and ICAO airport codes, CLLI codes, and customized abbreviations related to the city name [50].

IATA codes: This 3-letter airport code is the most common type of geohint found in router hostnames. The OurAirports database [35] contains 9,150 IATA codes, and annotates 91.9% with the primary city they serve. Of those, 58.5% contain three characters of the city name in order, such as "lon" for London, UK or "prg" for Prague, CZ. The remaining 41.5% do not follow this convention (e.g., "lax" for "Los Angeles, CA" or "yyz" for Toronto, CA), leading some operators to create an abbreviation based on the city name in order to allow others to interpret the location of the router.

ICAO codes: This 4-letter airport code is more structured than the IATA code, with the first one or two letters representing the continent or country – e.g., E for airports in Europe and K for airports in the U.S. ICAO codes are not as human-readable as IATA codes; equivalent ICAO codes for "lon", "prg", and "lax" are "egll", "lkpr", and "klax". In contrast to prior work [20, 46], we find no evidence that operators systematically embed ICAO codes in hostnames.

LOCODEs: This 5-letter code is structured similarly to the ICAO code. The first two characters represent the country, and the last three characters represent a location in that country. Generally, airports with an IATA code have the IATA code embedded in the LOCODE – e.g., "gblon", "uslax". Not all locations are obviously decoded; e.g., the LOCODE for Ashburn, VA is "usqas" as shown in router #3 in figure 1. The UN freely publishes coordinates for LOCODEs.

CLLI codes: This 11-character code can geolocate telecommunications equipment at the granularity of a building. The first six characters are a *CLLI prefix* – the first four characters identify a city or town, while the next two characters identify a state (in the US or Canada) or a country. For example, "asbnva" in figure 1 identifies Ashburn (asbn) in Virginia (va), while "londen" identifies London (lond) in England (en). Network operators register a CLLI code with a physical address, including coordinates, with iconectiv, the company that administers the CLLI database. Some network operators embed CLLI codes, ranging in length between 6 and 11 characters, into router hostnames.

City or town names: Operators may embed city or town names such as "ashburn" and "washington" in figure 1. Unlike the other codes, city names do not uniquely identify geographic locations. In our dictionary (§5.1.1) there are 10 locations named Washington and 2 named Ashburn. Unless the operator embeds a country or state code, as xo.net does in figure 1, the geohint can be ambiguous.

Facility names: Operators may embed the name or street address of the facility where the router is present. For example, they may embed "equinix" or "529bryant" to indicate that a router is present at the Equinix colocation facility at "529 Bryant St, Palo Alto, CA."

Country and state names: Operators may embed just the name or an abbreviation of a country or state when they have a limited deployment within a country or state – e.g., "australia" / "aus" or "queensland" / "qld".

3 RELATED WORK

Internet researchers extensively use geolocation databases, e.g., to correlate weather events with Internet outages [41], and understand the geographic distribution of systems infected by botnets [3, 9] and DNS manipulation [44]. In 2011, Poese *et al.* found that although the commercial databases were generally accurate for end-hosts at the country-level, their city-level accuracy was lower [45]. The use of geolocation databases for router geolocation is even more problematic [15]. The remainder of this section focuses on techniques relevant to geolocation of routers.

3.1 Delay-based Geolocation

Delay-based geolocation methods (e.g., [11, 12, 17, 18, 23, 42, 54]) use delay measurements from known locations to geolocate a system with an unknown location. In 2004, Gueye *et al.* introduced the seminal technique for constraint-based geolocation (CBG) of Internet hosts [17, 18]. Using delay measurements to build distance constraints from vantage points (VPs) with known locations, the CBG technique infers the system is located at the centroid of these distance constraints through multilateration, and provides an error estimate based on the width of the region in which the system could be. In this work, we use speed of light constraints to infer if a string in a hostname could represent the location of the system.

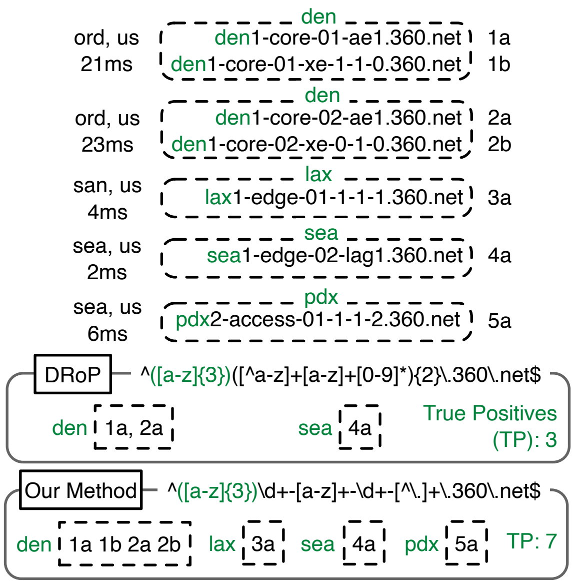
In 2006, Katz-Bassett *et al.* introduced topology-based geolocation (TBG) of Internet hosts [23]. They used traceroute to infer the directness of paths from VPs with known locations toward systems with unknown locations, and infer the location of intermediate routers and target systems using topological constraints from the traceroute paths. They also described how using undns rules to geolocate intermediate routers could reduce the error estimate by providing location estimates when the inferred network topology lacks sufficient constraints to geolocate routers. In 2007, Wang *et al.* built the Octant technique [54] that also uses the position of intermediate routers to geolocate end hosts, assisted by undns rules. In this work, we automatically learn rules and geohints that could inform TBG.

A recent trend is to use simpler methods because collecting and analyzing Internet topology is complex. In 2006, Katz-Bassett noted that a simple Shortest Ping [23] method – where the target is inferred to be in the same location as the VP with the shortest round trip time (RTT) to the target – was sufficient, and that the accuracy and precision of CBG [17, 18] was heavily influenced by the closest VP. In 2018, Trammell showed the significant element of luck involved in delay-based geolocation, as the single closest VP usually provides all of the geolocation benefit in practice [51]. RIPE Atlas' Single Radius geolocation engine uses this approach to provide an on-demand geolocation system [11].

3.2 Geolocation with Hostnames

Hostname-based geolocation techniques (e.g., [13, 42, 46, 49]) use geohints in hostnames to geolocate a system with an unknown location. In 2002, Spring *et al.* presented the seminal technique – the Rocketfuel ISP mapping system [49], and its undns tool. The undns tool included a manually-assembled database of regexes to extract information from hostnames, including router geolocation, router roles (backbone, gateway, customer), router names (strings that indicate two interfaces belong to the same router), and router points-of-presence (which routers are at the same location). Due to the openness of undns, researchers extended the database to support their research [14, 33, 34, 48, 58]. The database contains regexes that extract geohints for 241 different suffixes, and a dictionary that maps geohints to a location name. A significant volume of research was supported by undns geolocation regexes (e.g., [2, 8, 19, 22, 23, 25, 34, 39, 40, 54, 57]). However, constructing the database manually is time-consuming, the database does not contain lat/longs for the location names, and the undns regexes have not been updated since 2014.

In 2017, Scheitle *et al.* introduced the HLOC technique [46]. Rather than derive parsing rules, HLOC uses geolocation dictionaries to identify possible geohints at run time, and delay measurements from RIPE Atlas VPs near those locations to determine if the geohint is consistent. They included a manually-constructed dictionary containing 468 text strings to not consider as containing possible geohints, to limit the number of RIPE Atlas VPs used. Returning to the examples in figure 1, their dictionary contains "level", "atlas", "vodafone". Our method learns the *structure* of hostnames, so that our system will not consider hostname portions that



ord, us 21ms
den
den1-core-01-ae1.360.net 1a
den1-core-01-xe-1-1-0.360.net 1b
ord, us 23ms
den
den1-core-02-ae1.360.net 2a
den1-core-02-xe-0-1-0.360.net 2b
san, us 4ms
lax
lax1-edge-01-1-1-1.360.net 3a
sea, us 2ms
sea
sea1-edge-02-lag1.360.net 4a
sea, us 6ms
pdx
pdx2-access-01-1-1-2.360.net 5a
DRoP
^([a-z]{3})([^a-z]+[a-z]+[0-9]*){2}\.360\.net$
den 1a, 2a
sea 4a
True Positives (TP): 3
Our Method
^([a-z]{3})\d+-[a-z]+-\d+-[^\.]+\.360\.net$
den 1a 1b 2a 2b
lax 3a
sea 4a
pdx 5a
TP: 7

Figure 2: Comparing regexes inferred for 360.net. DRoP's rule extracts relative to the end of the suffix, and does not emit a sequence to cover sequences of digits, so only matches a subset of the hostnames.

do not contain geohints. They did not validate that their method extracted correct hints, just that other methods suggested locations that violated speed-of-light constraints. Finally, HLOC cannot learn geohints that operators derive that are not in their input dictionary.

3.3 Learning Geographic Conventions

In 2014, Huffaker *et al.* published the DRoP technique, which automatically learned regexes to extract geohints that operators embedded in hostnames for router IP addresses [20]. Their approach at a high-level is similar to ours; they automatically determine if an apparent geohint in a hostname is reasonable given topological and delay constraints from vantage points with known locations, and if the operator embeds geohints in a consistent position in the hostname. The method used four key metrics: RTT, RTT variation, hop count, and hop count variation. Their intuition was that routers with the same geohint would have a similar RTT and hop count. They trained a classifier using these metrics with ground truth from six networks rather than define heuristics that accept or reject a geohint. They used their method once to build regexes for 1,398 suffixes observed in July 2013. Their approach serves as inspiration for our work, but it has limitations that we address.

First, their regex building engine was too simplistic – it assumed that a suffix always placed the geohint at the same position relative to the end of the hostname. Further, the regexes were not specific, and only emitted one sequence per rule. Figure 2 illustrates a DRoP regex for 360.net, which matches only three of seven hostnames because it expects two segments separated by punctuation. Second, they only required a majority (>50%) of the extracted hints to be consistent with training data, which is problematic because their RTT measurements roughly constrained locations to within a continent. Third, their method assumed that operators used geohint databases verbatim, and did not derive new geohints that collided with an existing geohint (e.g., "ash" in router #4 in figure 1). Finally, they limited their RTT measurements to those observed in the traceroutes used to build the ITDK, to avoid the overhead of launching new probing of these interfaces from other VPs. However, the VP that observes a router in traceroute is rarely the closest VP to that router (§5.1.4), so collecting followup RTT measurements with ping from other VPs can better constrain inference.

Subsequent studies have identified problems with locations inferred using DRoP regexes [6, 10, 46]. In 2015, Cai probed 4,638 distinct locations that DRoP inferred for router hostnames, and found 46% were outside feasible boundaries imposed by CBG (§3.1) – e.g., DRoP built a regex to extract "chi" from cr1.chi2ca.sbcglobal.net and interpreted "chi" to place the router in "Chicago, IL" rather than "Chico, CA" [6]. In 2017, Scheitle *et al.* also compared locations extracted by DRoP against CBG-feasible boundaries and reported that most DRoP-inferred locations were incorrect [46].

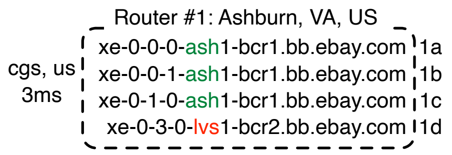
In 2018, Dan *et al.* described a machine learning algorithm that provided a sorted list of potential locations for a given hostname [10]. This system relied on a precise but proprietary list of 16 million end-user hostnames with known locations collected by Microsoft. Their system first split the hostname into constituent terms, matched those terms against a geolocation dictionary to generate a feature set, and then ran the resulting hostname and features against a binary classifier. Their system differed from DRoP in two significant ways. First, DRoP focused on router hostnames for which sizable ground truth is not available. Second, DRoP produced a set of domain specific rules, while the 2018 study [10] produced a single trained classifier whose dictionary they assumed applied to all domains. Because they did not train with or validate against router hostnames, their method has unknown applicability to router geolocation.

3.4 Learning Other Naming Conventions

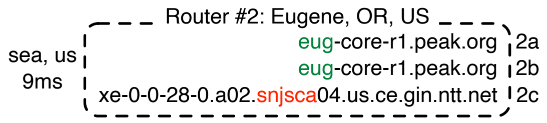
In 2019, Luckie *et al.* introduced an approach to extract meaningful structure from router hostnames, which they called *hostname orthography* and embedded in a software module called *Hoiho*. Their first goal was to identify hostname substrings common across all interfaces of the same router – the *router name* [30]. Their method evaluates automatically-generated regexes against sets of hostnames for IP addresses that other alias resolution techniques previously inferred to identify interfaces on the same router. In 2020, the same authors extended the Hoiho tool to automatically learn regexes to extract the *autonomous system number* (ASN) that operates the router [32]. The Hoiho framework provides an ideal platform to accelerate research on the hostname-based router geolocation challenge, and we leveraged it for our implementation (§5).

4 CHALLENGES

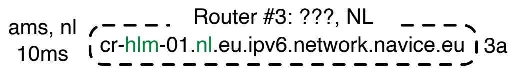
Our method learns if a network uses a naming convention that includes a geohint by evaluating automatically-generated candidate regexes against a set of training data. Conceptually, we infer the regex is extracting a geohint if two conditions hold: (1) if the


Router #1: Ashburn, VA, US
cgs, us
3ms
xe-0-0-0-ash1-bcr1.bb.ebay.com 1a
xe-0-0-1-ash1-bcr1.bb.ebay.com 1b
xe-0-1-0-ash1-bcr1.bb.ebay.com 1c
xe-0-3-0-lvs1-bcr2.bb.ebay.com 1d

(a) Stale hostname with an incorrect location string (lvs).


Router #2: Eugene, OR, US
sea, us
9ms
eug-core-r1.peak.org 2a
eug-core-r1.peak.org 2b
xe-0-0-28-0.a02.snjsca04.us.ce.gin.ntt.net 2c

(b) Hostname identifies the location of the provider's router (snjsca, San Jose CA) this router is connected to.



(c) Custom abbreviated location hints are lossy (hlm-nl).

Figure 3: Challenges for methods trying to learn naming conventions with embedded geohints.

regex extracts a geohint consistent with delay constraints, and (2) the regex extracts multiple such geohints for routers in different locations. This inference method is challenging for six key reasons.

1. Operators use different dictionaries and make up their own geohints. Without a standard set of geohints and single dictionary for interpreting them, network operators are left to choose a dictionary for their hostnames, and might use multiple dictionaries across their networks. Geohints also overlap across dictionaries, preventing the simple combination of location dictionaries into a single lexicon; e.g., the city name "london" refers to "London, UK", but overlaps with the CLLI for "London, Ontario, CA." Operators also repurpose geohints, such as using the IATA code "ash" to refer to "Ashburn, VA", or make up their own geohints. A successful solution must not only extract a geohint from a hostname, but also determine the dictionary capable of interpreting the geohint, recognize when networks use geohints incorrectly, and identify and interpret custom geohints.

2. We do not have ground truth to train a model. A traditional machine-learning (ML) approach requires ground truth to train a model. The model is then able to make a decision when presented with a new piece of information. We manually labeled the routers in figure 3 with their city-level locations. However, operators do not publish the location of each router beyond the hints they supply in hostnames, or publish the dictionary of names they use. Instead, we must rely on delay-based constraints to infer apparent geohints in hostnames.

3. The location expressed in a hostname may not be the location of the router. Prior work found that 0.5% of hostnames for a large network had incorrect geohints [55]. First, the hostname may be stale – it might have been assigned when its corresponding address was on a different router. For example, hostname 1d in figure 3a suggests the router is in Las Vegas, NV, while hostnames 1a-1c suggest the router is in Ashburn, VA. Note that hostnames 1a-1c are consistent with an RTT of 3ms from a VP near College Park, MD, while hostname 1d is not. Second, the geohint might represent the location of the provider's router when the address is provided to a neighbor for interconnection. Figure 3b gives an example using a router operated by Peak Internet in Eugene, OR and connected to NTT with interface 2c. The hostname for 2c identifies the address as assigned to a customer edge device connected to NTT's router named "r02.snjsca04.us". Here, the geohint corresponds to the location of the provider's router (San Jose, CA), and not the location of the customer router (Eugene, OR).

4. Abbreviated geohints are lossy. For instance, an operator abbreviating "Haarlem, NL" to "hlm" removes more than half of the letters of the city. This lossy compression leads to ambiguity that challenges inference methods. For example, router #3 in figure 3c could also correspond to "Helmond, NL" or "Hilversum, NL".

5. A hostname may contain a geohint by chance. Because there are so many 3-letter IATA codes, there is scope for unintentional collisions. For example, "gig" (gigabit ethernet), "eth" (ethernet), "cpe" (customer premises equipment) are frequently used by operators in router hostnames to indicate these properties, and they are all also IATA codes.

6. Delay measurements are limited by available vantage points. As of 2021, RIPE Atlas provides ≈10K VPs distributed around the world, with a focus on solving the needs of the operational community that funds RIPE. RIPE Atlas limits probing according to credits that operators and researchers accrue through hosting a VP. CAIDA's Archipelago (Ark) infrastructure provides ≈100 VPs also globally distributed around the world, without the same type of probing restrictions, but with sparser geographic coverage.

5 OVERVIEW OF METHOD

Our method learns if an operator uses a naming convention that embeds the router's geolocation in a hostname, by evaluating automatically generated candidate regexes against sets of router hostnames with the same suffix, e.g., ntt.net. Our method relies on the use of four data sources to seed and guide a four-stage process of learning naming conventions in hostnames. The four data sources are: a dictionary of geohints annotated with lat/long values (§5.1.1); a list of public DNS suffixes (§5.1.2); a curated corpus of routers and hostnames of interfaces on those routers (§5.1.3); and RTT measurements we gathered from a collection of globally distributed vantage points in the Internet (§5.1.4). The five stages of our method are (figure 4):

(1) assemble and load required input data (§5.1),
(2) identify apparent geohints in hostnames (§5.2),
(3) build and evaluate regexes that extract geohints (§5.3),
(4) learn geohints not in the reference dictionary (§5.4),
(5) select best convention per suffix (§5.5).

We designed and implemented methods for identifying apparent geohints in hostnames, methods for building and evaluating geolocation regexes, and methods for learning the location of an operator-specific geohint when the operator deviates from the geolocation dictionary. Figure 4 shows how we integrated our contributions into the Hoiho framework (§3.4). This existing open source software base served as a platform for our research, and also enabled us to contribute our new methodology back to the community as new modules of this software framework.

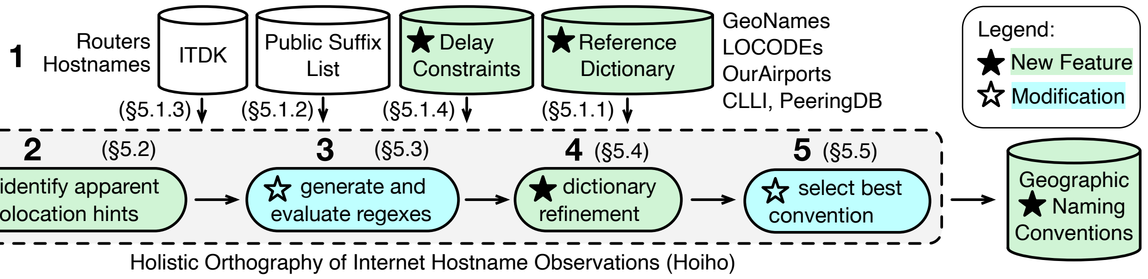


Figure 4: Overview of method for inferring naming conventions that extract geohints from hostnames.

	Aug '20 IPv4	Mar '21 IPv4	Nov '20 IPv6	Mar '21 IPv6
Routers	2.56M	2.57M	559K	525K
w/ hostnames	1.41M (55.0%)	1.39M (54.1%)	84K (15.1%)	84K (16.0%)
w/ RTT	2.10M (81.9%)	2.10M (81.7%)	265K (47.3%)	237K (45.2%)
Vantage Points	106	100	46	39

Table 1: Summary of ITDKs used in this work.

5.1 Stage 1: Assemble Required Input Data

We provide details on each data source and the role it plays in our method (§5.1.1-§5.1.4) before providing details on each stage of the method (§5.2-§5.4).

5.1.1 Reference Location Dictionary. We primarily used publicly available location sources to assemble our dictionary. We obtained IATA and ICAO airport codes from OurAirports [35], which included lat/longs and ISO-3166 country and state codes for 9,150 and 7,588 airports, respectively. We obtained a dictionary of city and town names from GeoNames [53], and retained 444,338 locations annotated with population and lat/long values. We obtained LOCODEs from the UN [52], which defined 111,807 codes as of December 2020; the UN annotated 87,023 codes with lat/longs, and we obtained a further 23,159 lat/longs from GeoNames [53]. We licensed a reference dictionary from iconectiv that maps the first six letters of a CLLI code (a CLLI prefix) to a city or town name [21]; we annotated these CLLI prefixes with lat/longs by joining the city or town name with GeoNames [53]. For clarity, a city or town is the most granular information available in the data we licensed. Finally, we obtained the name, street address, and lat/longs of 1,608 facilities from PeeringDB [1] with at least three networks present.

5.1.2 Mozilla Public Suffix List. Our method determines hostname suffixes using the Mozilla public suffix list [36], which lists effective top-level domains (e.g., .com, .net.au) under which operators can register their own domain suffixes (cogentco.com, ccnw.net.au).

5.1.3 CAIDA's Internet Topology Data Kit. We used CAIDA's Internet Topology Data Kit (ITDK [7]) as our input router-level graphs. The ITDKs contain inferred routers, collected by Ark vantage points during a 14-day measurement window, with hostnames for the router interfaces. We used four ITDKs: IPv4 ITDKs built in August 2020 and March 2021 with routers inferred using MIDAR [24] and

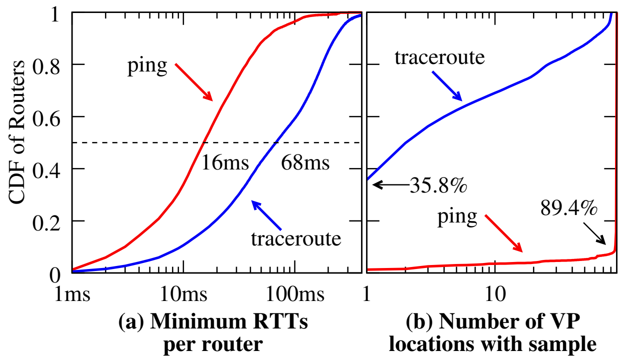


Figure 5: RTT measurements from Ark VPs to the 82.9% of IPv4 routers responsive to ping in the August 2020 ITDK demonstrate the value of additional ping measurements in our geolocation method. The median RTT in these traceroutes (68ms) represents a physical area 180x larger (πr^2) than the median RTT of pings to routers (16ms). 35.8% of routers were observed by only one VP in traceroutes used to assemble this ITDK. We obtained RTTs from all VPs for 89.4% of routers that were responsive to ping.

Mercator [16], and IPv6 ITDKs built in November 2020 and March 2021 with routers inferred using Speedtrap [29]. Table 1 summarizes the ITDKs; ≈55% and ≈16% of IPv4 and IPv6 routers, respectively, have hostnames.

5.1.4 Round Trip Time (RTT) Delay Measurements. We used CAIDA Ark vantage points to collect RTT measurements for router interfaces that were part of the ITDKs. We simultaneously collected these measurements during the measurement windows for each ITDK. We probed all routers from all VPs, as we could not know a priori which VP would observe the smallest RTT. Our goal was to obtain RTT samples to as many routers as possible, so we used ICMP echo probes soliciting ICMP echo responses, UDP probes to unused ports soliciting ICMP port unreachable responses, and TCP ack probes to port 80 soliciting TCP reset responses. We used the minimum of three RTT samples from each router/VP pair. To minimize the impact of our RTT measurements, we only used UDP or TCP probes if we did not receive a response to our ICMP or UDP probes. We discarded RTTs collected using TCP probes from seven VPs because we detected the VP's access router spoofing TCP reset packets in response to our probes – the RTTs were 1-2ms

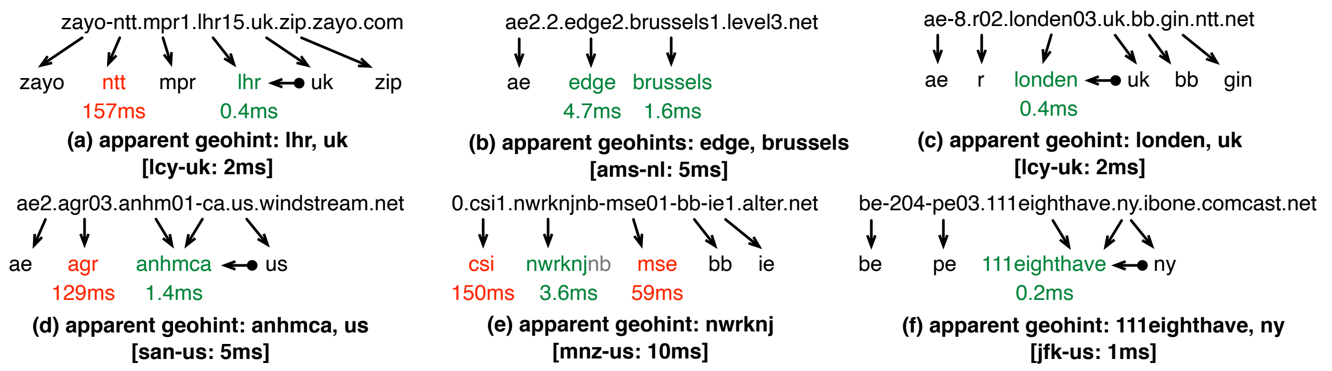

zayo-ntt.mpr1.lhr15.uk.zip.zayo.com
zayo
ntt
mpr
lhr
uk
zip
157ms
0.4ms
(a) apparent geohint: lhr, uk
[lcy-uk: 2ms]
ae2.2.edge2.brussels1.level3.net
ae
edge
brussels
4.7ms
1.6ms
(b) apparent geohints: edge, brussels
[ams-nl: 5ms]
ae-8.r02.londen03.uk.bb.gin.ntt.net
ae
r
londen
uk
bb
gin
0.4ms
(c) apparent geohint: londen, uk
[lcy-uk: 2ms]
ae2.agr03.anhm01-ca.us.windstream.net
ae
agr
anhmca
us
129ms
1.4ms
(d) apparent geohint: anhmca, us
[san-us: 5ms]
0.csi1.nwrknjnb-mse01-bb-ie1.alter.net
csi
nwrknjnb
mse
bb
ie
150ms
3.6ms
59ms
(e) apparent geohint: nwrknj
[mnz-us: 10ms]
be-204-pe03.111eighthave.ny.ibone.comcast.net
be
pe
111eighthave
ny
0.2ms
(f) apparent geohint: 111eighthave, ny
[jfk-us: 1ms]

Figure 6: Identifying apparent geohints in hostnames. Our method determines which strings in a hostname are *RTT-consistent*: have a measured RTT no smaller than the theoretical best-case RTT from each VP. Our method handles split CLLI prefixes, and identifies state/country codes that are part of the geohint.

regardless of the distance of the probed router from the VP. Future work could automatically filter these responses, by identifying highly connected components of VPs whose RTT-feasible locations are consistent, and discarding the remainder. Table 1 summarizes coverage; ≈82% and ≈46% of IPv4 and IPv6 routers, respectively, have RTT samples. We used all available Ark VPs – ≈103 and ≈43 VPs for IPv4 and IPv6 ITDKs, respectively.

Figure 5 summarizes key properties of the RTT dataset for the August 2020 ITDK. Figure 5a shows that 50% of the responding routers had a ping-based RTT from the closest VP of ≤16ms, placing the router within 1,600km (1,000 miles) of the closest VP. For 50% of these routers, the minimum RTT in a traceroute response was 68ms, 4.25x larger than the ping-based RTT, which represents a 180x larger physical area (πr^2). Figure 5b shows that 35.8% of the routers were only observed in traceroutes from one VP, whereas we obtained RTT samples from nearly all (89.4%) VPs for those routers responsive to ping using followup RTT measurements. This implies that a VP observing a router in traceroute (which DRoP used, §3.3) is rarely the closest VP to that router, and that conducting additional RTT measurements beyond those captured with traceroute provides tighter constraints (as we did).

5.2 Stage 2: Identify Apparent Geohints

With the data assembled in §5.1, our method begins by identifying apparent geohints in router hostnames, by comparing alphabetic strings prior to the hostname's suffix with RTT measurements.

For example, for zayo-ntt.mpr1.lhr15.uk.zip.zayo.com in figure 6a, our method considers "zayo", "ntt", "mpr", "lhr", "uk", and "zip". Our method then searches our dictionary for matching entries for the geohint, considering whether the measured RTTs are consistent with the location implied by the geohint. For each router-VP pair, our method calculates the theoretical best-case RTT between the candidate geohint's location and the VP's location according to the speed of light in a fiber optic cable. If the theoretical best-case RTT is smaller than the measured RTT for all VPs, then the measured RTT is *RTT-consistent*. For example, while "ntt" and "lhr" are IATA codes, only "lhr" is RTT-consistent. Our location dictionary (§5.1.1) contains ISO-3166 code GB-ENG for "lhr"; because "GB" and "UK" are equivalent, our method includes "uk" as part of the apparent geohint. Our method tags the hostname with the expected extractions, so that when evaluating regexes (§5.3), it can penalize a regex that does not extract an apparent state/country code.

```
(a)  ^.+\.([a-z]{3})\d+\.([a-z]{2})\.[a-z]{3}\.zayo\.com$
(b)  ^.+\.([a-z]+)\d*\.level3\.net$
(c)  ^.+\.([a-z]{6})\d+\.([a-z]{2})\.[a-z]{2}\.gin\.ntt\.net$
(d)  ^.+\.([a-z]{4})\d+-([a-z]{2})\.([a-z]{2})\.windstream\.net$
(e)  ^.+\.([a-z]{6})[a-z\d]+-[a-z]+\d+-[^\.]+\.alter\.net$
(f)  ^[^\.]+\.(\d+[a-z]+)\.([a-z]{2})\.[a-z]+\.comcast\.net$
```

Figure 7: Regexes extracting geohints from the corresponding hostnames in figure 6.

Figure 6b shows a hostname with two apparent geohints with locations ("Edge, GB" and "Brussels, BE") that are RTT-consistent. Our method tags both, and determines which geohint is the actual location in the next stage.

Figures 6c-6e illustrate different ways that operators can embed (portions of) CLLI codes. Figure 6c shows that NTT embeds the first 6 letters of a CLLI code, as well as a country code. Figure 6d shows that alter.net embeds the first 8 letters of a CLLI code; our method therefore considers whether the first 6 letters of a longer string could be a CLLI prefix. Similarly figure 6e shows that Windstream splits a 6-letter CLLI prefix into its 4- and 2-letter components; our method also considers whether adjacent 4- and 2-letter components could be a CLLI prefix.

Finally, figure 6f shows a hostname with a facility street address embedded. Our method processes all strings separated with punctuation within a hostname, identifying apparent geohints by first comparing each string against street addresses in PeeringDB facility records; for example, PeeringDB contains "111 8th Ave". Our method tags this apparent geohint because the location of the facility is RTT-consistent.

5.3 Stage 3: Build and Evaluate Regexes

We build on prior work that automatically learned regexes to extract *router names* [30] and *autonomous system numbers* [32] from hostnames. We extended Hoiho's regex builder to generate regexes that extract geohints over the course of four stages, which increase the specificity and coverage of the regexes as the method proceeds. Figure 7 shows the final regexes that our method built for the

VP, RTT	Hostname	Router
cgs, us 9ms	gcr-company.gigabitethernet4-1.core1.ash1.he.net	#1
cgs, us 3ms	100ge1-2.core1.ash1.he.net	#2
cgs, us 3ms	100ge10-1.core2.ash1.he.net	#3
cgs, us 5ms	46–labs-llc.ve401.core2.ash1.he.net	#4

		Facility	Population	TP	FP	PPV
	Existing Hint: IATA ash (Nashua, NH, US)		-	#1	#2, #3, #4	25%
▶	Candidate Location: Ashburn, VA, US	★	43,511	#1, #2, #3, #4	-	100%
	Candidate Location: Ashland, VA, US	★	7,503	#1, #2, #3, #4	-	100%
	Candidate Location: Ashland, NJ, US		8,202	#1, #2, #3, #4	-	100%

(a) Learning that "ash" corresponds to "Ashburn, VA, US" for he.net. The existing geohint had a PPV of 25%. "Ashburn, VA, US" has a PPV of 100%, facilities, and the largest population among candidate locations in the set.

VP, RTT	Hostname	Router
zrh, ch 6ms	ae-7.r02.mlanit01.it.bb.gin.ntt.net	#1
zrh, ch 6ms	ae-3.r21.mlanit02.it.bb.gin.ntt.net	#2

		Facility	Population	TP	FP	PPV
▶	Candidate Location: Milan, IT	★	1,236,837	#1, #2	-	100%
	Candidate Location: Montesilvano Marina, IT		45,991	#1, #2	-	100%

(b) Learning that "mlanit, it" corresponds to "Milan, IT" for ntt.net. "mlanit" is an unknown (UNK) CLLI code, and the actual CLLI code for Milan, IT in the CLLI dictionary is different, i.e. NTT implemented their own.

Figure 8: Learning geohints not in reference dictionary (§5.4). Our method learns when an operator re-purposes an existing geohint, such as "ash" in (a), and when an operator creates a new geohint, such as "mlanit" in (b).

hostnames in figure 6; these suffixes contain hundreds of other hostnames that – in combination with the hostnames in figure 6 – informed the selection of a final regex. Because the approach to build the regexes is similar to previous work [30, 32], we provide the details in appendix A, and instead focus on how our method evaluates regexes, as this informs learning new geohints (§5.4) and selecting per-suffix regexes (§5.5).

Because of the heterogeneity of hostname structure present within any given suffix, as well as the many different possible interpretations of a given string within a hostname, our method may build many hundreds or thousands of candidate regexes. Further, our method may need to use multiple regexes simultaneously to capture different geohint formats within a suffix. We define a *naming convention* (NC) as one or more regexes that extract geohints for a given suffix. We developed the following four per-hostname classifications with the goal of selecting a NC that extracts all available geolocation information from a hostname, including country or state codes, which can help our method accurately learn geohints when present (§5.4).

Our method assigns a *true positive* (TP) if a regex extracted a geohint that is plausible given the measured RTTs for the router, and the regex extracted any state and/or country code that was also part of the apparent geohint – e.g., if it extracted "lhr, uk" from the hostname in figure 6a. Our method assigns a *false positive* (FP) if a regex extracted a geohint that is not RTT-consistent – e.g., if it extracted "ntt" from the hostname in figure 6a. Our method assigns a *false negative* (FN) if a regex did not extract a geohint when our method tagged an apparent geohint (§5.2) or did not extract a state and/or country code from the hostname that was also part of the apparent geohint – e.g., if it did not extract anything or it extracted "lhr" from the hostname in figure 6a. Finally, our method assigns *unknown* (UNK) if a regex extracted a geohint from the hostname but the geohint is not in our dictionary – e.g., "ldn, uk".

5.4 Stage 4: Learn Operator Geohints

This stage of our method attempts to learn geohints and their meaning when operators embed geohints that are either not in our dictionary, or map to a different location than the location in our dictionary. Our intuition is that operators generally use geohints in our dictionary, but may deviate from the dictionary for readability (§2). That is, (1) the fraction of locations where the operator uses a geohint of their own is small relative to their overall geohint dictionary, (2) the NCs our method built using the reference dictionary serve as a useful starting point to learn per-suffix geohints, and (3) the geohint is an abbreviation of an existing city, state, or country name (a *place name*).

Figure 8 illustrates the approach. Our method begins with all NCs that identify at least three unique RTT-consistent geohints with a Positive Predictive Value (PPV = TP / (TP + FP)) $> 40\%$ – these are NCs where our method has confidence the operator is embedding geohints. Our method then considers the routers with geohints that (1) are not RTT-consistent (FPs) – as "ash" is in figure 8a, or (2) are not in the dictionary (UNKs) – as "mlanit" is in figure 8b. Our method then consults the dictionary, and for each geohint, considers if the extraction could be an abbreviation for a place name in the dictionary. Existing work in the natural language processing (NLP) community (e.g., [38, 43]) describes methods for automatically learning acronyms and abbreviations given surrounding text. Using the previous sentence as an example, those methods determine that "NLP" is an abbreviation for natural language processing. Because we lack that contextual information in hostnames, we instead use the following heuristics.

First, our method considers that the extraction could be an abbreviation of a place name if all of the characters in the extraction appear in the place name in order, and if the first character matches (e.g., "ash" matches "Ashburn, VA", and "mlan" matches "Milan"). If the place name consists of multiple words (e.g., "New York") then we also require the first letter of a word to match before matching other characters in that word (e.g., we allow "nyk" but not "nwk"). When considering abbreviations for a regex that extracts place names (e.g., "ftcollins" for "Fort Collins") our method requires the abbreviation to match at least four contiguous characters in the place name.

Second, our method evaluates the RTT hints for the candidate locations, counting the number of routers RTT-consistent (TP) or not RTT-consistent (FP) with the location. Our method ranks the candidate locations, first by those known to have a facility (via PeeringDB), then by population, then by TPs. Our method chooses the highest-ranked location, provided that the PPV of that geohint is at least 80% (because we expect the geohints to be generally correct), and is better than the existing geohint by more than one TP (we allow for the fact that an existing geohint might be correct and the operator has stale hostnames). Our intuition is that the method should choose locations known to host facilities, breaking ties using population using the observation by Lakhina *et al.* that router deployment is correlated with population density [26]. Figure 8 shows that our method chooses "Ashburn, VA" over "Ashland, VA" because it has a larger population.

Finally, if the regex does not extract a country or state code (figure 8a) our method requires three congruent routers before inferring a new geohint. If the regex does extract a country or state code (figure 8b) our method only requires one congruent router before inferring a new geohint. Our intuition is that the presence of a country or state code reduces the chance that we are extracting strings that are not geohints and thus the chance of over-fitting.

5.5 Stage 5: Ranking and Classifying

Our metric for ranking NCs is the number of Absolute True Positives (ATP = TP - (FP + FN + UNK)). Our goal is to find a NC that extracts RTT-consistent geohints for as many hostnames as possible within a suffix, without extracting strings that are not geohints. Our method chooses a NC from the set of NCs inferred per suffix by selecting the highest ranked NC, only choosing a lower ranked NC if it is made up of fewer regexes with nearly the same set of matches (the lower ranked NC has no more than three TPs fewer).

Our method classifies a NC as *good* if it extracted at least three unique location hints consistent with training data with a PPV ≥90%, *promising* if it extracted at least three unique location hints consistent with training data with a PPV ≥80%, and *poor* otherwise. The good and promising NCs are *usable* because they usually extract a geohint consistent with the router's location.

6 RESULTS

We applied our method to the four ITDKs described in §5.1.3. Table 2 shows the coverage of usable NCs inferred by our method on the ITDK routers. We note that while a lower fraction of IPv6 routers have hostnames, those hostnames are more likely to have an apparent geohint (§5.2) than IPv4 routers – ≈35% (IPv6) compared to ≈16% (IPv4). Our automatically generated usable NCs extracted geohints from ≈85% of the IPv4 routers with apparent geohints, and ≈89% of the IPv6 routers with apparent geohints; these represent ≈7.4% and ≈4.9% of the total number of IPv4 and IPv6 routers in these ITDKs, respectively. Table 3 summarizes the classifications made for the NCs inferred from the ITDKs. Our method inferred the operator was systematically embedding geohints in their router hostnames for 49.6% of the 1825 suffixes with an apparent geohint in the August 2020 ITDK, and these NCs extracted 86.8% of the apparent geohints in that ITDK (table 2). Similarly, while our method learned usable NCs for 61.3% of suffixes with a geohint in the November 2020 IPv6 ITDK, these NCs extracted 89.3% of the apparent geohints in that ITDK. Overall, our method inferred usable NCs for 1023 different domain suffixes with IPv4 routers (August 2020 and March 2021), and 241 different domain suffixes with IPv6 routers (November 2020 and March 2021). These results demonstrate the potential for our method to generate geolocations that can serve as anchors for TBG methods (§3.1).

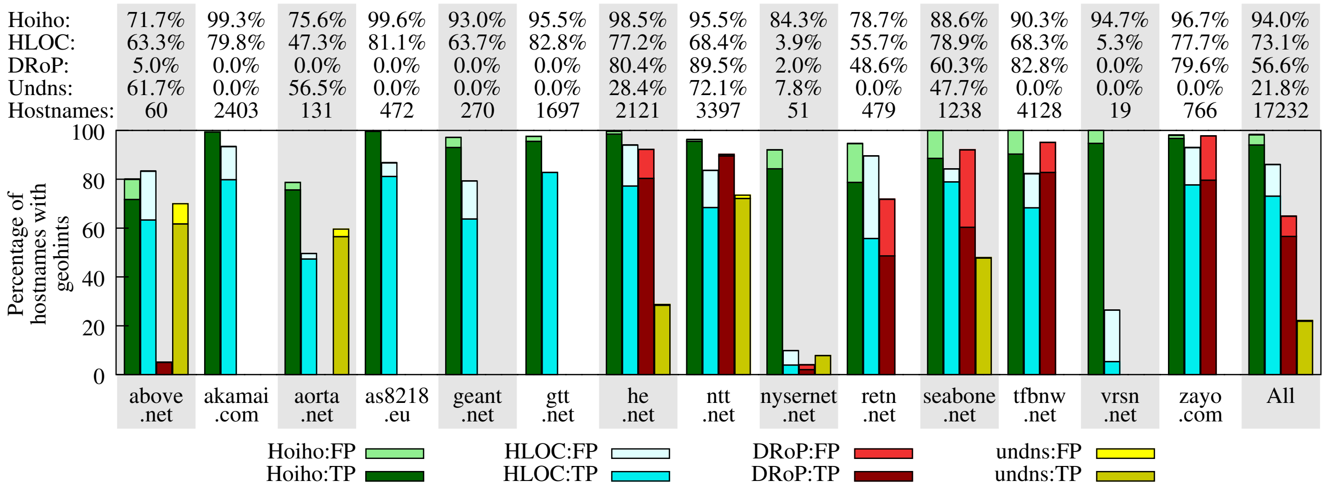
Routers	Aug '20 IPv4	Mar '21 IPv4	Nov '20 IPv6	Mar '21 IPv6
total	2.56M	2.57M	559K	525K
with hostname	1.41M (55.0%)	1.39M (54.1%)	84K (15.1%)	84K (16.0%)
with apparent geohint	225K (8.8%)	220K (8.5%)	29K (5.3%)	31K (5.8%)
geolocated	195K (7.6%)	183K (7.1%)	26K (4.7%)	27K (5.2%)

Table 2: Coverage of usable NCs, which each extracted 83.4% – 89.6% of apparent geohints from hostnames.

Classification	Aug '20 IPv4	Mar '21 IPv4	Nov '20 IPv6	Mar '21 IPv6
Good	795 (43.6%)	777 (43.2%)	195 (56.4%)	189 (56.2%)
Promising	111 (6.1%)	120 (6.7%)	17 (4.9%)	16 (4.8%)
Poor	919 (50.4%)	903 (50.2%)	134 (38.7%)	131 (39.0%)
Total	1825	1800	346	336

Table 3: Classification of NCs. Our method built "good" or "promising" NCs for 906 (49.6%) of the 1825 suffixes with an apparent geohint using the August 2020 ITDK.

Table 4 summarizes the NCs learned for the August 2020 IPv4 ITDK. Of the 795 NCs our method classified as *good*, 411 (51.7%) used an IATA code to identify the city where the router is located, 309 (38.9%) used city names, 96 (12.1%) used CLLI prefixes, 10 (1.3%) used LOCODEs, and 2 (0.3%) used street addresses of facilities. Of these NCs, 31 (3.9%) used a mix of geohint types, such as that of alter.net illustrated in figure 13 in appendix A; because each regex in a NC extracts a single geohint type, these NCs consist of multiple regexes. Table 4 shows the fraction of NCs that also embedded a state and/or country code. Nearly a quarter (23.6%) of good NCs that embedded an IATA code also embedded a state and/or country code. Interestingly, 5 of 96 (5.2%) good NCs that embedded a CLLI prefix also embedded a country or state code; these were mostly redundant, as CLLI prefixes are mostly used in the US, and US CLLI prefixes already contain a state. The use of IATA codes was more dominant for the NCs learned for the November 2020 IPv6 ITDK – 75.9% of good NCs used an IATA code, 16.9% used a city name, 6.2% used a CLLI prefix, and 2.6% used a LOCODE.



	above .net	akamai .com	aorta .net	as8218 .eu	geant .net	gtt .net	he .net	ntt .net	nysernet .net	retn .net	seabone .net	tfbnw .net	vrsn .net	zayo .com	All
Hoiho:	71.7%	99.3%	75.6%	99.6%	93.0%	95.5%	98.5%	95.5%	84.3%	78.7%	88.6%	90.3%	94.7%	96.7%	94.0%
HLOC:	63.3%	79.8%	47.3%	81.1%	63.7%	82.8%	77.2%	68.4%	3.9%	55.7%	78.9%	68.3%	5.3%	77.7%	73.1%
DRoP:	5.0%	0.0%	0.0%	0.0%	0.0%	0.0%	80.4%	89.5%	2.0%	48.6%	60.3%	82.8%	0.0%	79.6%	56.6%
Undns:	61.7%	0.0%	56.5%	0.0%	0.0%	0.0%	28.4%	72.1%	7.8%	0.0%	47.7%	0.0%	0.0%	0.0%	21.8%
Hostnames:	60	2403	131	472	270	1697	2121	3397	51	479	1238	4128	19	766	17232

Figure 9: Comparing router geolocation inferences from HLOC, DRoP, undns, and our method in Hoiho. We considered router hostnames that we knew from operator feedback contained geohints. An inference is successful if the method geolocates a router within 40km of its ground truth location. For all domains, our method in Hoiho had more true positives than HLOC, DRoP, and undns. The gap between the top of each bar and 100% represents false negatives – geohints a method did not use.

Geohint	Annotation	Good		Promising	
IATA	- none	314	(39.5%)	47	(42.3%)
	- state	8	(1.0%)	1	(0.9%)
	- country	89	(11.2%)	2	(1.8%)
	- total	411	(51.7%)	50	(45.0%)
City	- none	289	(36.4%)	62	(55.9%)
	- state	14	(1.8%)	0	
	- country	4	(0.5%)	1	(0.9%)
	- both	2	(0.3%)	0	
	- total	309	(38.9%)	63	(56.8%)
CLLI prefix	- none	91	(11.4%)	0	
	- state	2	(0.3%)	0	
	- country	3	(0.4%)	0	
	- total	96	(12.1%)	0	
LOCODE	- none	10	(1.3%)	0	
Facility	- none	1	(0.1%)	0	
	- state	1	(0.1%)	0	
	- total	2	(0.3%)	0	
Overall		795		111	

Table 4: Fraction of NCs that embed a geohint and also embed a state and/or country code. Operators are more likely to embed a country or state code when embedding an IATA hint than a city name or CLLI prefix.

6.1 Learned Naming Conventions

We contacted operators at 19 networks, receiving validation from 13. All 13 responses confirmed that our NCs captured the intent of the operators to embed geohints in hostnames. We confirmed the dictionary types and the customized geohints in each suffix (§6.2) that the network operators used.

In order to evaluate the accuracy of our method, we compared locations inferred by our method with the validation data provided by these 13 networks. We also used validation data we inferred for seabone.net, whose operators included city names in Italian (e.g., "Atene" for "Athens, GR") alongside their 3-letter geohints (e.g., "ate") embedded in hostnames. We also evaluated locations for each router inferred using the accessible academic state of the art: HLOC (§3.2, [46]), DRoP (§3.3, [20]), and undns (§3.2, [49]). We are currently unable to compare with Microsoft's method (§3.3, [10]) as Microsoft requires a non-disclosure agreement.

This analysis considers the router hostnames that we knew from operator feedback contained geohints. Figure 9 is a stacked bar chart with the percentage of true positives on the bottom, false positives on top, and the gap between the top of the false positives and the 100% line representing the false negatives. As in [20], a geolocation is considered a true positive if its inferred location is within 40 km of the true location. Our method correctly geolocates more routers across all domains than HLOC, DRoP, and undns, with our method correctly geolocating an average of 94.0% of router hostnames with a geohint compared to DRoP's average of 56.6% and HLOC's average of 73.1%. Our method also yields fewer false positives than HLOC and DRoP. Of the router hostnames that each method returned a location for, the PPV of the methods were 98.3% (undns), 95.6% (Hoiho), 87.2% (DRoP), and 85.1% (HLOC).

The observable false negatives for above.net and aorta.net for our method in figure 9 are because those operators are inconsistent in their naming convention and our method did not see enough samples confidently to learn a pattern. The observable false negatives for DRoP across all suffixes are largely because their published ruleset is old (2013), and their regexes are simplistic (§3.3, figure 2). DRoP's observable false positives are because the method did not attempt to learn when the operator deviated from the dictionary. Similarly, the observable false negatives for undns across all suffixes are because the ruleset was manually constructed, last updated in 2014, and the published ruleset contains a subset of the location codes used within each suffix. However, because the location codes

	Hint	#	Location	Alternatives
⊗	ash	12	Ashburn, VA, US	iad:66
⊗	tor	10	Toronto, ON, CA	yyz:19
	wdc	9	Washington, DC, US	was:10
⊗	tok	8	Tokyo, JP	nrt:14, hnd:6
	zur	8	Zurich, ZH, CH	zrh:30
⊗	ldn	7	London, GB	lon:96

Table 5: Most frequently learned three-letter geohints, with number of suffixes our method inferred use the geohint. ⊗ means an airport has that IATA code in our dictionary. We show the closest IATA codes for that location.

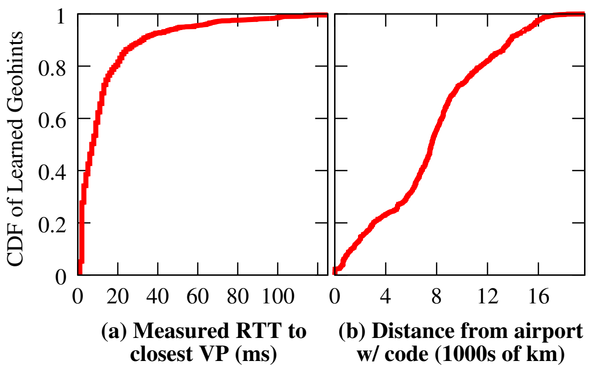


Figure 10: Properties of learned geohints inferred for ITDK router nodes in August 2020. 80% were within 22ms of the closest VP. 50% are at least 7600km from the airport with the corresponding IATA code.

for each suffix in undns were manually interpreted, nearly all were correct; in our validation data, only one (kslrml for ntt.net) was incorrect (mapped to "Kuala Lumpur, MY" not "Kuala Selangor, MY").

Finally, HLOC's observable false positives are because it also does not attempt to learn when an operator deviates from the dictionary, and it contains confirmation bias. HLOC considers possible geohints in each hostname, selects Atlas VPs closest to those geohint locations, and then reports if the geohints are RTT-consistent; it does not use Atlas VPs further away from the possible location that could determine that the geohint is not RTT-consistent. For example, a router with "de-cix1.rt.act.fkt.de.retn.net" located in "Frankfurt am Main, HE, DE" (fkt.de) was considered by HLOC as possibly located in "Waco, TX, US" or "Chiclayo, PE" because the locations implied by the "act" and "cix" airport codes were consistent with the RTTs from the Atlas VPs it used. HLOC's observable false negatives are because it does not find a geohint in the hostname, and because it may not be able to get an RTT sample from the VP; the clearest example of this is nysernet.net in figure 9, whose routers can only be probed from R&E networks, but HLOC did not select a probe in one of these networks.

We investigated how much our learned geohints improved the performance of our method. If we had not learned geohints, then our method would have only correctly geolocated 82.4% of hostnames with a geohint, and the PPV of our method would have been 94.5%.

aorta	as8218	geant	gtt	he	ntt
3/4 (75%)	3/3 (100%)	8/8 (100%)	12/12 (100%)	4/4 (100%)	17/18 (94.4%)

retn	tfbnw	zayo	seabone	Overall
25/34 (73.5%)	2/14 (14.3%)	4/4 (100%)	14/15 (93.3%)	92/117 (78.6%)

Table 6: Fraction of learned geohints within each suffix that we verified. All but seabone.net were verified by operators. We validated geohints for seabone.net by comparing the geohint with the expanded city name also encoded in each hostname.

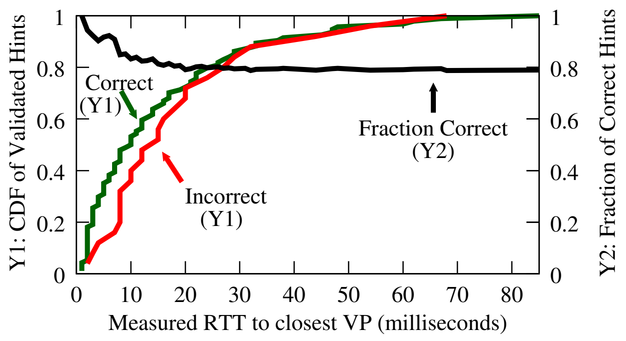


Figure 11: Properties of 117 learned geohints that we validated for ITDK router nodes in August 2020. For routers with RTT to the closest VP <= 7ms, 90% of learned geohints were correct.

6.2 Learned Geohints

Operators may use a mix of geohints from a dictionary, and their own custom geohints (§5.4). Of the 461 usable regexes that extracted apparent 3-letter IATA airport codes for the August 2020 ITDK, 147 (38.2%) included at least one hint that did not correspond to an IATA airport code serving the location. Table 5 lists six learned geohints that were shared among at least seven different suffixes, and the nearest IATA code for each location. Four of these six learned geohints have a name collision with an IATA code. Three of the most-frequently learned geohints, "ash", "tor", and "tok", have nearby airports whose IATA code has no obvious relationship to the name of city it primarily serves, so the operator motivation to introduce custom geohints for these locations is clear. However, the other three learned geohints do have nearby airports with IATA codes that have clear relationships to the names of the cities they primarily serve, and the motivation for introducing new geohints is unclear.

Figure 10a shows the shortest RTT from a VP with a known location to all learned locations. 48.6% of the learned hints were within 10ms (1000km) of a VP with a known location, and 80% were within 22ms (2200km). Figure 10b shows the importance of learning when an operator uses a custom dictionary, as 93.5% of the geohints are more than 1000km from the airport with the same code and 50% are at least 7600km from the airport.

Table 6 summarizes our validation results for these learned geohints. Overall, 92 of 117 (78.6%) of our learned geohints were correct. The inferred locations are usually correct for network operators who deploy network infrastructure where there is population density. The 12 locations for tfbnw (Facebook) that we inferred the wrong location for are routers located in data centers in small population areas whose geohints are named irregularly. Their backbone network uses traditional IATA codes, and overall we extracted correct geohints from 90.3% of their router hostnames (figure 9).

Figure 11 shows the distribution of RTTs to the closest VP for learned geohints that we validated. Learned geohints with smaller RTTs are more likely to be correct than those with larger RTTs; for RTTs <= 7ms, 90% were correct, for RTTs <= 11ms, 84% were correct, and for RTTs <= 16ms, 80% were correct. This implies that more VPs distributed around the world could lead to higher quality learned geohints using our method.

7 LIMITATIONS

Stale Hostnames: Zhang *et al.* established in 2006 that stale hostnames can distort the accuracy of geolocation inferences using those hostnames, and presented approaches to automatically detect and mitigate stale hostnames with incorrect geohints [55].

Complex Hostname Encodings: Our method currently builds regexes that extract geohints delimited by punctuation or digits in hostnames, but operators do not always delimit geohints with punctuation or digits. AT&T (figure 12a) embeds five-character geohints that contain a US state code in the last two characters. The first three characters contain an identifier for a city, sometimes expressed with a digit – e.g., rd3 for Richardson, TX – and usually not using an IATA airport code. These identifiers are challenging to interpret, even for a human. Similarly, Open Transit (figure 12b) embeds three-letter location codes, sometimes not using an IATA airport code – e.g., ash, loa, and nyk – followed by a string that identifies the role of the router (cr / tr). Without ground truth locations to train with, it is difficult to automatically infer a convention in the absence of punctuation that structures a geohint. For example, a human might recognize that the router with hostname *francetelecom.lon01.atlas.cogentco.com* is operated by France Telecom, and that its interconnecting router is in London, UK. But delay constraints available to an algorithm might suggest that the substring "fra" or "fran" could refer to "Frankfurt am main, HE, DE", or that the operator encoded "France" as a country-level geohint.

Current research in entity extraction using regexes within the machine learning community relies on a human in the loop (e.g., [4, 5, 27, 37, 47, 56]). Nevertheless, a fully automated holistic approach that first determines *francetelecom* refers to the operator of the router could allow an automated geolocation method to not consider that portion of a hostname as a possible geohint.

8 CONCLUSION

The formidable challenge of Internet router geolocation has limited progress on a wide range of research and operational pursuits for decades. Leveraging recent developments in automated learning of regexes that extract features from IP hostnames, we designed and implemented a method that automatically learns regexes for extracting geohints from hostnames, even when operators deviate from common dictionaries or create their own geohints. We contributed our results back to the community via open source software [28] and a public web site of inferred regexes and geohints [31]. The per-suffix web pages we created served as a conduit to facilitate ground truth validation from operators, who could easily verify or correct our inferences. We are also aware of colleagues now relying on our web site to more accurately characterize the Internet's router-level structure using information encoded in hostnames [59].

Our results represent significant advances, but there is still room for improvement, and several promising next steps. One future direction could explore introducing traceroute-observed topology constraints, in addition to RTT constraints, to further refine our inferred conventions. Further developments in holistic orthography of Internet hostnames may allow a future method to automatically reason that a substring in a hostname is used for a specific purpose, such as identifying the role or operator of a router, and not consider the substring as a geohint. But perhaps the most promising next step is to synthesize this new capability with tools that perform IP address alias resolution and router-level inter-domain topology mapping, allowing one to progressively improve coverage and fidelity of annotated router-level maps of Internet topology.

VP	RTT	Hostname
atl, us	7ms	atnga00002cce9-irb-2.infra.cdn.att.net
ord, us	9ms	bcvoh00002cce9-irb-2.infra.cdn.att.net
dal, us	5ms	dlltx00001cce9-irb-2.infra.cdn.att.net
jfk, us	1ms	nycny00002cce9-irb-2.infra.cdn.att.net
dal, us	4ms	rd3tx00001cce9-ae120-100.infra.cdn.att.net
sjc, us	4ms	scaca00002cce9-ae120-200.infra.cdn.att.net

Dictionary:				
atnga:	Atlanta, GA		nycny:	New York City, NY
bcvoh:	Brecksville, OH		rd3tx:	Richardson, TX
dlltx:	Dallas, TX		scaca:	Sacramento, CA

(a) AT&T

VP	RTT	Hostname
ams, nl	4ms	hundredgige0-0-0-0.amscr6.-.opentransit.net
bwi, us	4ms	hundredgige0-3-0-2.ashtr2.-.opentransit.net
san, us	4ms	hundredgige0-5-0-1.loatr1.-.opentransit.net
lcy, uk	2ms	hundredgige0-0-0-2.lontr5.-.opentransit.net
ory, fr	2ms	hundredgige0-0-0-1.partr2.-.opentransit.net
bdl, us	6ms	hundredgige0-4-0-3.nyktr2.-.opentransit.net

Dictionary:				
ams:	Amsterdam, NL		lon:	London, UK
ash:	Ashburn, VA, US		par:	Paris, FR
loa:	Los Angeles, CA, US		nyk:	New York, NY, US

(b) France Telecom Open Transit

Figure 12: Not all operators use a geolocation convention that is easily parsed with a regex due to the use of custom geohints that are not delimited with punctuation.

9 ACKNOWLEDGMENTS

We thank Young Hyun for assistance with the ITDK, our shepherd Gareth Tyson, and the anonymous reviewers for their feedback. This work is partly supported by U.S. NSF awards CNS-2105393, 1925729, 1901517, and OAC-1724853, and the U.S. DoD Defense Advanced Research Projects Agency under CA-HR00112020014. It does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

REFERENCES

[1] [n.d.]. *PeeringDB.* https://www.peeringdb.com/

[2] Ashok Anand, Archit Gupta, Aditya Akella, Srinivasan Seshan, and Scott Shenker. 2008. Packet Caches on Routers: The Implications of Universal Redundant Traffic Elimination. In *SIGCOMM*. 219–230.

[3] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *USENIX Security Symposium.*

[4] Rohit Babbar and Nidhi Singh. 2010. Clustering Based Approach to Learning Regular Expressions over Large Alphabet for Noisy Unstructured Text. In *AND*. 43–50.

[5] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. 2016. Inference of Regular Expressions for Text Extraction from Examples. *IEEE Transactions on Knowledge and Data Engineering* 28, 5 (May 2016), 1217–1230.

[6] Guan Yan Cai. 2015. *IP infrastructure geolocation.* Master's thesis. Naval Postgraduate School.

[7] CAIDA. 2021. Macroscopic Internet Topology Data Kit (ITDK). https://www.caida.org/data/internet-topology-data-kit/.

[8] Nicole Lee Caruso. 2011. *A Distributed System For Large-Scale Geolocalization Of Internet Hosts.* Master's thesis. Cornell University.

[9] Alberto Dainotti, Alistair King, Kimberly Claffy, Ferdinando Papale, and Antonio Pescapé. 2015. Analysis of a "/0" Stealth Scan from a Botnet. *IEEE Transactions on Networking* 23, 2 (April 2015), 341–354.

[10] Ovidiu Dan, Vaibhav Parikh, and Brian D. Davison. 2018. *IP Geolocation through Reverse DNS.* Technical Report. https://arxiv.org/pdf/1811.04288.pdf.

[11] Ben Du, Massimo Candela, Bradley Huffaker, Alex C. Snoeren, and kc claffy. 2020. RIPE IPmap Active Geolocation: Mechanism and Performance Evaluation. *ACM SIGCOMM Computer Communication Review* 50, 2 (April 2020), 4–10.

[12] Brian Eriksson, Paul Barford, Bruce Maggs, and Robert Nowak. 2012. Posit: a Lightweight Approach for IP Geolocation. *ACM SIGMETRICS Performance Evaluation Review* 40, 2 (Oct. 2012), 2–11.

[13] Andrew D. Ferguson, Jordan Place, and Rodrigo Fonseca. 2013. Growth Analysis of a Large ISP. In *IMC*. 347–352.

[14] Michael J. Freedman, Mythili Vutukuru, Nick Feamster, and Hari Balakrishnan. 2005. Geographic locality of IP prefixes. In *IMC*.

[15] Manaf Gharaibeh, Anant Shah, Bradley Huffaker, Han Zhang, Roya Ensafi, and Christos Papadopoulos. 2017. A Look at Router Geolocation in Public and Commercial Databases. In *IMC*. 463–469.

[16] Ramesh Govindan and Hongsuda Tangmunarunkit. 2000. Heuristics for Internet Map Discovery. In *INFOCOM*. 1371–1380.

[17] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. 2004. Constraint-Based Geolocation of Internet Hosts. In *IMC*.

[18] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. 2006. Constraint-Based Geolocation of Internet Hosts. *IEEE/ACM Transactions on Networking* 14, 6 (Dec. 2006), 1219–1232.

[19] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. 2008. Measuring and Evaluating Large-Scale CDNs. In *IMC*.

[20] Bradley Huffaker, Marina Fomenkov, and kc claffy. 2014. DRoP: DNS-based Router Positioning. *CCR* 44, 3 (July 2014), 6–13.

[21] iconectiv. 2020. http://store.commonlanguage.com/Codes/CLLI-Code-Online.html.

[22] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. 2009. *Nation-State Routing: Censorship, Wiretapping, and BGP.* Technical Report. https://arxiv.org/pdf/0903.3218.pdf.

[23] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. 2006. Towards IP Geolocation Using Delay and Topology Measurements. In *IMC*. 71–84.

[24] Ken Keys, Young Hyun, Matthew Luckie, and k claffy. 2013. Internet-Scale IPv4 Alias Resolution with MIDAR. *IEEE/ACM Transactions on Networking* 21, 2 (April 2013), 383–399.

[25] Yohei Kuga, Kenjiro Cho, and Osamu Nakamura. 2008. On inferring regional AS topologies. In *AINTEC*. 9–16.

[26] Anukool Lakhina, John W. Byers, Mark Crovella, and Ibrahim Matta. 2003. On the Geographic Location of Internet Resources. *IEEE JSAC* 21, 6 (Aug. 2003), 934–948.

[27] Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and H. V. Jagadish. 2008. Regular Expression Learning for Information Extraction. In *EMNLP*. 21–30.

[28] Matthew Luckie. 2010. Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet. In *IMC*. 239–245.

[29] Matthew Luckie, Robert Beverly, William Brinkmeyer, and k claffy. 2013. Speedtrap: Internet-scale IPv6 Alias Resolution. In *IMC*. 119–126.

[30] Matthew Luckie, Bradley Huffaker, and k claffy. 2019. Learning Regexes to Extract Router Names from Hostnames. In *IMC*. 337–350.

[31] Matthew Luckie, Bradley Huffaker, Alexander Marder, Zachary Bischof, Marianne Fletcher, and k claffy. 2021. Data supplement for "Learning to Extract Geographic Information from Internet Router Hostnames". https://www.caida.org/publications/papers/2021/hoiho/.

[32] Matthew Luckie, Alexander Marder, Marianne Fletcher, Bradley Huffaker, and k claffy. 2020. Learning to Extract and Use ASNs in Hostnames. In *IMC*.

[33] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Aravind Krishnamurthy, and Arun Venkataramani. 2006. iPlane: An information plane for distributed services. In *OSDI*.

[34] Ratul Mahajan, Ming Zhang, Lindsey Poole, and Vivek Pai. 2008. Uncovering Performance Differences among Backbone ISPs with Netdiff. In *NSDI*. 205–218.

[35] David Megginson. 2021. OurAirports. https://ourairports.com/.

[36] Mozilla Foundation. 2020. Public Suffix List. https://publicsuffix.org/list/.

[37] Karin Murthy, Deepak P., and Prasad M. Deshpande. 2012. Improving Recall of Regular Expressions for Information Extraction. In *WISE*. 455–467.

[38] David Nadeau and Peter D. Turney. 2005. A Supervised Learning Approach to Acronym Identification. *LNAI* 3501 (2005), 319–329.

[39] Abdullah Yasin Nur and Mehmet Engin Tozal. 2018. Cross-AS (X-AS) Internet topology mapping. *Computer Networks* 132 (Feb. 2018), 53–67.

[40] Abdullah Yasin Nur and Mehmet Engin Tozal. 2018. Geography and Routing in the Internet. *ACM Trans. Spatial Algorithms Syst.* 4, 4, Article 11 (Sept. 2018), 16 pages.

[41] Ramakrishna Padmanabhan, Aaron Schulman, Dave Levin, and Neil Spring. 2019. Residential Links Under the Weather. In *SIGCOMM*. 145–158.

[42] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. 2001. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *SIGCOMM*. 173–185.

[43] Youngja Park and Roy J. Byrd. 2001. Hybrid Text Mining for Finding Abbreviations and their Definitions. In *EMNLP*.

[44] Paul Pearce, Ben Jones, Frank Li, Roya Ensafi, Nick Feamster, Nick Weaver, and Vern Paxson. 2017. Global Measurement of DNS Manipulation. In *USENIX Security Symposium.*

[45] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. 2011. IP Geolocation Databases: Unreliable? *ACM SIGCOMM CCR* 41, 2 (April 2011), 53–56.

[46] Quirin Scheitle, Oliver Gasser, Patrick Sattler, and Georg Carle. 2017. HLOC: Hints-Based Geolocation Leveraging Multiple Measurement Frameworks. In *TMA*.

[47] Stanley Simoes, Deepak P, Manu Sairamesh, Deepak Khemani, and Sameep Mehta. 2018. Content and Context: Two-pronged Bootstrapped Learning for Regex-formatted Entity Extraction. In *AAAI*. 5924–5931.

[48] Neil Spring. 2012. *Grounding undns.* Technical Report.

[49] Neil Spring, Ratul Mahajan, and David Wetherall. 2002. Measuring ISP topologies with Rocketfuel. In *SIGCOMM*. 133–145.

[50] Richard A. Steenbergen. 2014. A Practical Guide to (Correctly) Troubleshooting with Traceroute. https://archive.nanog.org/sites/default/files/traceroute-2014.pdf.

[51] Brian Trammell and Mirja Kühlewind. 2018. Revisiting the Privacy Implications of Two-Way Internet Latency Data. In *PAM*. 73–84.

[52] UNECE. 2020. UN/LOCODE Code List by Country and Territory. https://unece.org/trade/cefact/unlocode-code-list-country-and-territory.

[53] Marc Wick. 2021. GeoNames. https://www.geonames.org/.

[54] Bernard Wong, Ivan Stoyanov, and Emin Gün Sirer. 2007. Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts. In *NSDI*.

[55] Ming Zhang, Yaoping Ruan, Vivek Pai, and Jennifer Rexford. 2006. How DNS Misnaming Distorts Internet Topology Mapping. In *USENIX ATC*. 34–39.

[56] Shanshan Zhang, Lihong He, Eduard C. Dragut, and Slobodan Vucetic. 2019. How to Invest my Time: Lessons from Human-in-the-Loop Entity Extraction. In *KDD*. 2305–2313.

[57] Ying Zhang, Z. Morley Mao, and Ming Zhang. 2008. Effective Diagnosis of Routing Disruptions from End Systems. In *NSDI*. 219–232.

[58] Ying Zhang, Zhuoqing Morley Mao, and Ming Zhang. 2009. Detecting traffic differentiation in backbone ISPs with NetPolice. In *IMC*.

[59] Zesen Zhang, Alexander Marder, Ricky Mok, Bradley Huffaker, Matthew Luckie, kc claffy, and Aaron Schulman. 2021. Inferring Regional Access Network Topologies: Methods and Applications. In *IMC*.

A GENERATING GEO REGEXES

Generate Base Regexes: The first phase builds base regexes that extract geohints, focusing on structure encoded in the hostname with punctuation. This phase captures apparent geohints using regex components that match alphabetic characters, expressed with [a-z] and a modifier that either matches a specific number of characters, or a sequence of 1+ characters. Our method uses ([a-z]{2})

VP+RTT	Hostname (PTR record)	VP+RTT	Hostname (PTR record)
sjc,us: 4ms	0.xe-10-0-0.gw1.sfo16.alter.net (a)	dca,us: 8ms	0.af0.rcmdva83-mse01-aa-ie1.alter.net (g)
jfk,us: 1ms	0.xe-0-0-0.il1.nyc41.alter.net (b)	mnz,us: 10ms	0.csi1.nwrknjnb-mse01-bb-ie1.alter.net (h)
nrt,jp: 3ms	0.so-0-1-3.xt1.tko2.alter.net (c)	fdh,de: 16ms	dis-00008.munich2.de.alter.net (i)
dca,us: 5ms	0.ae1.br2.iad8.alter.net (d)	ams,nl: 12ms	dis-00019.stuttgart2.de.alter.net (j)
sea,us: 4ms	0.ae1.gw3.sea7.alter.net (e)	ams,nl: 17ms	ckh.dresden.de.alter.net (k)
ams,nl: 2ms	0.ae1.br2.ams3.alter.net (f)	ams,nl: 11ms	disy-2.frankfurt.de.alter.net (l)

			PLAN	TP	FP	FN	UNK	ATP	PPV
Phase 1: Generate Base Regexes									
#1	▷	^.+\.([a-z]{3})\d+\.alter\.net$	IATA	a,b,d,e,f	c	g,h,i,j,k		-1	83%
#2	◇	^[^\.]+\.[^\.]+\.([a-z]{6})[^-]+-.+\.alter\.net$	CLLI	g,h		a,b,d,e,f,i,j,k		-6	100%
#3	☆	^[^\.]+\.([a-z]+)\d+\.([a-z]{2})\.alter\.net$	City, CC	i,j		a,b,d,e,f,g,h,k		-6	100%
#4	□	^[^\.]+\.([a-z]+)\.([a-z]{2})\.alter\.net$	City, CC	k		a,b,d,e,f,g,h,i,j	l	-9	50%
Phase 2: Merge Regexes									
#5	☆□	^[^\.]+\.([a-z]+)\d*\.([a-z]{2})\.alter\.net$	City, CC	i,j,k		a,b,d,e,f,g,h	l	-5	75%
Phase 3: Embed Character Classes									
#6	◇	^\d+\.[a-z]+\d+\.([a-z]{6})[a-z\d]+-.+\.alter\.net$	CLLI	g,h		a,b,d,e,f,i,j,k		-6	100%
Phase 4: Build Regex Sets									
#7	▷	^.+\.([a-z]{3})\d+\.alter\.net$	IATA	a,b,d,e,f g,h,i,j,k	c		l	8	83%
	☆□	^[^\.]+\.([a-z]+)\d*\.([a-z]{2})\.alter\.net$	City, CC						
	◇	^\d+\.[a-z]+\d+\.([a-z]{6})[a-z\d]+-.+\.alter\.net$	CLLI						

Figure 13: Inferring a naming convention (NC) that extracts geohints from alter.net router hostnames across four phases according to training data. The regexes that form the NC increase in specificity and coverage through each phase. We annotate each regex with a symbol to identify an evolving regex as our method refines it. §5.3 describes the evaluation metrics. Appendix A describes the four phases.

to extract country codes, ([a-z]{3}) to extract IATA codes, ([a-z]{4}) to extract ICAO codes, ([a-z]{5}) to extract LOCODES, ([a-z]{6}) to extract CLLI prefixes, and ([a-z]+) to extract city names. For other parts of the hostname surrounding the geohint, this phase uses regex components that exclude specific punctuation (e.g., [^\.]+ and [^-]+ match sequences of characters that do not contain a dot or hyphen, respectively) or match anything (i.e., .+) at most once per regex, using the approach from prior work [30, 32]. For example, for hostname (a) in figure 13, this phase builds ([a-z]{3})\d+ (regex #1) to extract "sfo" from "sfo16", and builds combinations of regex components that exclude specific punctuation for the remainder of the hostname – e.g., ^.+\.([a-z]{3})\d+\.alter\.net$. Our method annotates each regex with a *plan* to decode the geohint that a regex extracts – for example, regex #3 in figure 13 extracts a city name and country code.

Merge Regexes: The second phase merges regexes that are similar – they differ by a single simple string – in order to increase coverage. We extended Hoiho to consider the case when two regexes only differ because one regex contains \d+ while the other regex does not. This phase merges regex #3 (which matches hostnames i and j) and regex #4 (which matches hostnames k and l) to build regex #5 in figure 13 – replacing \d+ (sequence of 1 or more digits) with \d* (an optional sequence of digits) – which increases coverage by matching hostnames i, j, k, and l.

Embed Character Classes: The third phase identifies character class sequences in common for matched hostnames, replacing the components that exclude specific punctuation built in phase one (e.g., [^\.]+ and [^-]+) with components that specify character classes. For example, the first component [^\.]+ for regex #2 in figure 13 matches only digits, while the second component matches a sequence of alphabetic characters followed by a digit. Therefore, this phase replaces the first component with \d+, and the second component with [a-z]+\d+ to build regex #6. We also extended Hoiho to consider when operators use fixed-width strings outside of the captured geohints – for example, for NTT's regex in figure 7, we extended Hoiho to use [a-z]{2} to match "bb" (shown in figure 6c), as well as "ce" and "ra" strings which also appear in place of "bb" (not illustrated).

Build Regex Sets: The final phase builds sets of regexes to form a naming convention (NC) in order to increase coverage when the operator uses multiple formats, which often occurs when the operator uses different geohint types, using the per-hostname evaluation metrics (TP, FP, FN, UNK) defined in §5.3. This phase ranks regexes by the descending number of Absolute True Positives (ATP = TP - (FP + FN + UNK)), and evaluates the outcome of combining a regex with each of the regexes below it in the rank order, iterating this step until it has not built a new NC. This phase includes an expanded regex in its working set if the ATP is greater than the ATP of the regex it started with, each regex in the expanded NC extracts at least three unique geohints, and the Positive Predictive Value (PPV = TP / (TP + FP)) is no worse than 10% lower than the PPV of the NC this phase started with. For example, in the first iteration, this phase merges regexes #1 and #5 to form a NC that extracts IATA airport codes and city names from hostnames; in the next iteration, it merges #6 into the NC to form NC #7.