DRoP:DNS-based Router Positioning

Bradley Huffaker, Marina Fomenkov, kc claffy
{bradley,marina,kc}@caida.org
CAIDA, University of California, San Diego

ABSTRACT

In this paper we focus on geolocating Internet routers, using a methodology for extracting and decoding geography-related strings from fully qualified domain names (hostnames). We first compiled an extensive dictionary associating geographic strings (e.g., airport codes) with geophysical locations. We then searched a large set of router hostnames for these strings, assuming each autonomous naming domain uses geographic hints consistently within that domain. We used topology and performance data continually collected by our global measurement infrastructure to discern whether a given hint appears to co-locate different hostnames in which it is found. Finally, we generalized geolocation hints into domain-specific rule sets. We generated a total of 1,711 rules covering 1,398 different domains and validated them using domain-specific ground truth we gathered for six domains. Unlike previous efforts which relied on labor-intensive domain-specific manual analysis, we automate our process for inferring the domain specific heuristics, substantially advancing the state-of-the-art of methods for geolocating Internet resources.

Keywords

router geolocation, DNS, Internet topology, active measurement

1. INTRODUCTION

Governments, researchers, and commercial entities share an interest in mapping Internet resources to physical locations, a process termed *geolocation*. Internet resources, e.g., routers and IP addresses, can broadly be divided into those located topologically inside the network and providing transit to those on the edge. Edge (end) hosts are mostly clients and servers; knowledge of their location supports a variety of uses, from tax and regulation enforcement to advertising and nearest data center selection. Geolocation of transit-providing routers helps to estimate redundancy and robustness of critical infrastructure, diagnose and fix connectivity problems, and construct POP-level maps of the Internet [1, 2, 3].

Current geolocation techniques rely on three types of data: delay, database, topology, or hostname heuristics. Delay-based methods typically use delay measurements gathered from known sources to hosts at known geographic landmarks to triangulate a target IP address location [4, 5, 6]. Database-driven methods aggregate static mapping information from public and private sources [7, 8]. Topology inference methods assume that topologically close addresses are also physically close [1, 9, 10]. Hostname heuristic techniques use geographic hints encoded in domain names to infer locations of associated IP addresses [11, 12].

These techniques involve non-trivial hurdles for anyone trying to build and maintain an accurate geolocation service. Since private databases are generally inaccessible, most users rely on third-party (commercial) geolocation providers, which tend to focus resources on accurate geolocation of edge nodes (end hosts) rather than transit routers, so their data sources and analysis methodologies are typically best suited for inferring end host locations. But building geographically accurate maps of the Internet requires geolocating core routers, especially those belonging to large Internet Service Providers (ISPs) with global infrastructure footprints. We therefore seek ways to accurately and automatically infer router locations from data that already exists or is easy to collect. In this study we propose, test, and validate a method for inferring router location that piggybacks on our existing extensive continuous Internet IPv4 topology data collection and analysis. Specifically, we use geographic information encoded in DNS hostnames of router interfaces and active measurement data obtained by probing those interfaces from multiple monitors distributed worldwide in the course of producing our Internet Topology Data Kits (ITDK – [13]). We limited ourselves to pre-existing data collections, but our results suggest ways to optimize future operational measurement specifically to support accurate geolocation.

Many ISPs encode some kind of geographic information ("hints") into DNS mappings for their routers. Some router names may include standardized geographic indicators (i.e., airport codes or city names), while others may use non-standard naming conventions such as creative city abbreviations. Previous attempts to infer router locations [11, 12] used hand-crafted rule sets that extracted and decoded this geographic information from router hostnames, and developed methods to detect stale geographic information in hostnames [14]. We investigate and evaluate a novel approach that may be able to automate the process of finding domain-wide geographic hints in hostnames, testing their validity using active probing measurement data, and generalizing rules that map hints used by different domains (organizations) to specific geographic locations.

Our approach is as follows. We first compiled an extensive dictionary associating geographic hints (city names, abbreviations, airport codes, etc.) with specific locations (§2.1). We searched for these potentially encoded geographic hints in a large set of router hostnames extracted from a recent ITDK. We assume that each autonomous second-level domain (e.g., *above.net*) uses geographic hints consistently, but different second-level domains may represent geographic hints differently. We use latency and TTL measurements captured by Ark monitors [15] (also as part of the existing ITDK measurements), to ascertain whether a given hint (string) appears to co-locate different hostnames in which it is found, i.e., are the round-trip times (RTTs) and TTLs of packets arriving from different hosts consistent with the two hostnames being co-located?

The rest of the paper describes the datasets we used (§2), our methodology (§3), results and validation, (§4) and conclusions (§6).

2. DATASETS

2.1 Building a dictionary of location hints

By analyzing the naming conventions of three organizations who shared them (ntt.net, cogentco.com, and belwue.de), emails from operators describing geographic hints they use, a NANOG presentation [16], and previous work [11, 12], we found five types of commonly used geographic hints:

1, 2 Publicly available **IATA and ICAO codes** [17], used by the airline industry to designate airports. Operators often label routers using the code of the largest airport in the city where a router is located.

3 **CLLI** (Common Language Location Identifier) codes [18], developed by the North American telecommunications industry to precisely specify (possibly down to an individual cabinet) location and function of telecommunication equipment. Today operators typically use a truncated form that encodes the city and country. The CLLI code database is proprietary, trademarked and maintained by Telcordia.

4 **UN/LOCODE** (the United Nations Code for Trade and Transport Locations) [19], developed and maintained by the UN Economic Commission for Europe for use by the shipping and manufacturing industry. It assigns codes to locations used in trade and transport such as seaports, rail and road terminals, airports, post offices and border crossing points. The 2011 version contains codes for about 82,000 locations. A few Internet operators prefer to use the more complete UN/LOCODE database rather than the IATA codes.

5 Finally, some operators use city names.

Geographic names (or hints) may be ambiguous; for example, the same geohint could mean different cities: Moscow in Russia or in Idaho. Therefore, we include mappings for each geographic string to *all possible* corresponding coordinates we could find.

We first populated our dictionary with location names extracted from GeoNames [20], a public database with over 8M place names and corresponding {lat,long} coordinates. We downloaded their daily dump from 1 August 2013. For each *P-type* entry[1], we converted all alternative names listed in the database to ASCII, removed all non-alphabetic characters (e.g., converted "san diego" to "sandiego") and stored the resulting string-coordinate pairs in our dictionary. For multi-part names, we added an entry with only the longest part of the name (i.e., "diego" for "san diego"). Finally, for each hint, we designated the coordinates with the largest population as the major city.

Next we incorporated into the dictionary IATA and ICAO airport codes listed in Wikipedia [17], which unfortunately includes only city and airport names, not the corresponding coordinates. We found the (lat,long) coordinates for 62% of airports in the Openflight database [21], mapped 19% of the airports to cities in GeoNames and used their coordinates, and looked up the remaining 19% using Google's Maps API [22]. We generated a single dictionary entry for each IATA or ICAO airport code.

Currently, there is no public dataset for the CLLI codes. We compiled a small subset of CLLI codes from various sources, and tied city names to geographic coordinates via the GeoName database.

[1]A *P-type* entry names a city, village, or other populated place, rather than another geographic object such as a lake or mountain.

Finally, we added the character strings from the UN/LOCODE list [19], which includes geographic coordinates.

As an example of the extensive coverage compiled into the resulting dictionary, the following hints – sandiego, diego, san (IATA), ksan (IACO), sndgca (CLLI), ussan (UN/LOCODE), and additional spellings provided by geonames (sandiegas, diegas, sandiegu, diegu, sandieqo, dieqo, sandijego, dijego, sandiyego, and diyego) all connect to a single geographical location "San Diego, CA, US".

Table 1 shows the composition of our dictionary. The character strings of major cities constitute the bulk of entries (78%), followed by the LOCODE strings (19%), and by IATA codes (1.1%). Since we did not have access to the Telcordia database, the number of CLLI codes in our dictionary is miniscule.

2.2 Routers and hostnames in CAIDA's ITDK

To develop and test our methodology, we needed a set of routers to try to geolocate. We used the set of Internet routers inferred from Ark traceroute data [23] and collated in the July 2013 Internet Topology Data Kit (ITDK). Ark monitors around the world continuously collect traceroutes to randomly selected IP addresses in every routed /24 IPv4 prefix; in parallel a host at CAIDA performs a reverse DNS lookup for every observed IP address, storing the results in a database. The ITDK is a series of heavily curated data sets derived from 3-week windows of traceroute measurements gathered from CAIDA's active measurement infrastructure Archipelago (Ark). For each ITDK, CAIDA conducts alias resolution using the best available tools [24, 25] to estimate which IPv4 addresses belong to the same physical routers (called "nodes" in the ITDK).

Even with state-of-the-art tools, alias resolution at Internet scale is a complex and challenging process; the resulting router-level graph is incomplete and has analytic artifacts that are difficult to quantify and filter [26]. For example, the number of inferred nodes depends not only on the comprehensiveness and accuracy of alias resolution, but also on the conditions of data collection: how many non-responding intermediate hops and responding destinations observed. (Any responding destination could be a router.) Furthermore, the ITDK router-level graph is a *hypergraph*, i.e., we infer many routers to be interconnected by the same link, which may be due to MPLS, inadequate alias resolution, topology changes during data collection, or non-responding hops distorting the graph.

The ITDK curation process replaces non-responding hops (denoted by * in traceroute data) between two known IPv4 addresses with provisional *placeholder* nodes in order to maintain knowledge of connectivity between these two addresses. If a known interface has N placeholders as immediate neighbors, they could all be the same interface, or N different interfaces (or anything in between). These placeholder hops propagate into the router-level graph, inflating its size and creating "shadowy" nodes: inferred routers with all interfaces unknown. In the July 2013 ITDK data, 66.5% of inferred nodes (31.8M) have at least one known IPv4 interface, the remaining 33.5% (16M) contain only placeholder interfaces.

The ITDK also includes a file listing DNS hostnames (if they exist) for intermediate nodes; we augment this list with available hostnames for destination nodes using CAIDA's internal database that stores reverse DNS lookups for every traceroute-observed IP address. We then construct a file that lists the hostnames for the ITDK nodes (i.e., routers) for which at least one interface had a hostname mapping. We use this subset of 18.9M nodes with 18.5M associated hostnames to test our geolocation methodology. Most (99.9%) hostnames map to one interface, but some hostnames map to multiple IPv4 addresses, for a variety of reasons, including hostnames changing during the measurement window, or DNS-based load balancing and server selection methods.

	airport				telco		United Nation		population	
	IATA		**ICAO**		**CLLI**		**LOCODE**		**major city**	
number of hints	7,622	**1.1%**	6,402	**0.9%**	121	**0.0%**	134,106	**19.5%**	540,223	**78.5%**

Table 1: Number and percentage of entries in our dictionary of geographic hints contributed by different sources.

2.3 Latency and TTL measurements

Geographic information encoded in a router's hostname can appear conflicting, ambiguous, or invalid. For example, the hostname *ccr21.par01.atlas.cogentco.com* contains three potentially location-related strings: *ccr* (airport code for Concord, CA), *par* (Paris, France or one of the 20 towns named Paris in the USA), and *atlas* (part of the city name Salas Atlas, Spain). We use latency and TTL measurements gathered operationally by Archipelago to help resolve such ambiguities. (One motivation for this study was our intent to include the locations of ITDK routers as an additional component of this data kit.)

Each IP address observed in Ark traces has associated Round Trip Time (RTT) and Time-To-Live (TTL) values from reply packets arriving at Ark monitors that sent probes to this address. From the TTL values in those replies, we can infer the hop count, i.e., path length or number of routers between a given monitor and the router it probes. We use RTTs and hop counts as location resolvers because for a given Ark monitor, we assume that it should take roughly the same amount of time and the same number of hops to travel to routers located in the same place.

For each {Ark monitor, router} pair, we computed: (1) the minimum RTT measured between this monitor and any interface on this router it probed, and (2) the average TTL in the reply packets from any interface on this router arriving at this monitor. The RTT primarily depends on the geographic distance between the monitor and the router, as well as congestion conditions along the path; we use the minimum RTT to reduce the impact of congestion on our analysis. The observed TTL value depends on the initial TTL value set by the router when it created the response packet (different routers use different values, see Figure 1); because most routers decrement the TTL field, the ending TTL value reflects the approximate number of routers between the probed router and the Ark monitor. We stored the pairs of values $\{min\_RTT, av\_TTL\}$ in a database and used them to construct a distance vector between each router in the ITDK and the Ark monitors that probed any interface(s) on that router.

3. METHODOLOGY

We designed a four-step methodology to infer DNS hostname heuristics that use common geographic naming conventions. First, we divide hostnames into groups sharing a common Public Suffix (as defined in Mozilla's Public Suffix List [27]), assuming such hostnames belong to the same organization and utilize a common naming convention. We derive a *geohint* for each geographic hint we find at the same place in hostnames with a common suffix (§3.1). Second, for each geohint, we construct a 4-dimensional vector of values (§ 3.2) computed from minimum RTTs and average TTLs between CAIDA's Ark monitors and the routers associated with this geohint. Third, we (use ground truth to) train a classifier and use it to determine whether the remaining geohints are likely valid (§3.3). Finally, we combine valid geohints into more general rules describing the position and type of geographic hint found in each public suffix's hostnames (§3.4).

3.1 Creating router sets and geohints

For hostnames with the same public suffix, we search for strings

hostname	ccr21.	par01.	atlas.	cogentco.com
position	2	1	0	
hint	ccr	par	atlas	
location	|	|	Salas Atlas, ES	
	|	Paris, FR		
	Concord, CA			

Table 2: Extracting geographic hints from a hostname.

	elements		
geohint	public suffix	hint+position	geographic coordinates

Table 3: A *geohint* is a unique combination of a geographic hint, a public suffix, a hint's position in a hostname, and a lat/long pair.

that could be geographic hints located in the same position in the hostname. Only 3.6M of nearly 19M nodes with DNS mappings in the July 2013 ITDK data have apparent geographic hints in their DNS names.

Table 2 illustrates the concept of *position*. We strip the common public suffix part of the name, split the remainder into substrings (truncating non-alphabetic characters), and count the distance from each substring to the stripped suffix part of the hostname. For example, in Table 2 the substring *ccr* is in position 2 and the substring *par* is in position 1. We then search our dictionary (§2.1) for geographic hints we extracted from hostnames, similar to the technique used in [11]. If an observed substring is in our dictionary, then we include the router containing this hostname into a corresponding *router set*, defined as a group of routers that share the same public suffix and have the same geographic hint encoded in the same position in at least one of the hostnames of their interfaces. In other words, a router set is defined by three parameters: [public suffix, substring, substring position]. Thus, the router with the interface name shown in Table 2 will be a member of at least three router sets (possibly more, depending on the composition of hostnames of its other interfaces): [cogentco.com, *ccr*, position 2], [cogentco.com, *par*, position 1], and [cogentco.com, *atlas*, position 0].

Each substring found in our dictionary may map to multiple physical locations, e.g., *paris* may be Paris, France, or one of the 20 towns named Paris in the USA. For each router set we create as many geohints as there are possible geographical locations for the substring defining this router set (Table 3). In other words, a geohint is a unique combination of a geographic hint (potentially location-related string of characters), a particular public suffix, specific position in hostname, and specific geographic coordinates (latitude/longitude pair). For example, the hint *diego* found in the third position in the hostnames with public suffix *as2116.net* would produce two geohints: [*as2116.net*, *diego*, 3, San Diego USA] and [*as2116.net*, *diego*, 3, Diego Martin Tobago].

3.2 Testing geohints with RTT/TTL data

We use available active measurement data to test whether a given geohint is likely to accurately map its router set to the correct location. We create a 4-dimensional vector derived from Ark probes sent to the interfaces of all routers in the set as follows.

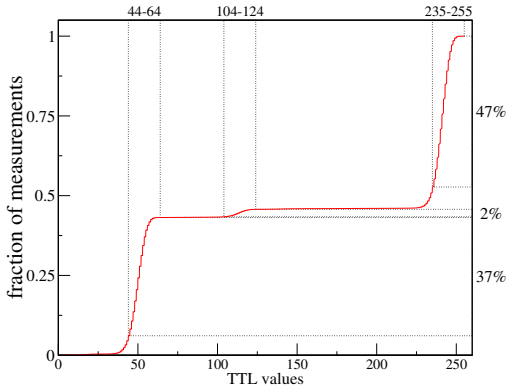

Figure 1: CDF of TTL values found in reply packets. Y-axis shows fraction of TTL values within 20 hops of each initial value of 64 (37%), 128 (2%), or 255 (47%). The union of these ranges covers 86% of TTL values.

First, we compute the mean and standard deviation of minimum RTTs observed between a given Ark monitor and all interfaces this monitor probed in this router set. To remove possible bias due to the RTT long tail distribution [28], we filter out values larger than the mean plus three standard deviations, and recalculate the new mean $\overline{min\_RTT}$ and its standard deviation. The latter is one component of the classifying 4-D vector.

Next, we use $\overline{min\_RTT}$ to estimate the expected signal propagation speed between the known location of the Ark monitor and the coordinates of each possible geohint associated with this router set. Given the dynamic nature of the underlying network, we do not attempt to estimate the correct current speed in the network, but rather use the measurements to teach our classifier the correct range of values. Therefore, we compute the average propagation speed from Ark monitors that observed any router in this router set, and its standard deviation, making these two values components of our classifying 4-D vector. We automatically reject geohints when the estimated speed exceeds the speed of light.

Finally, we analyze the hop count of paths from each Ark monitor to routers in a router set. A given router sets the same initial TTL value in all outgoing packets, and we can calculate the hop count by subtracting the TTL value observed in the collected reply packet from the initial TTL. Although different routers in the same geographic location may have different initial TTLs, the distance in hops between that router and a given Ark monitor is relatively stable. In the July 2013 ITDK, 86% of the observed TTL values are within 20 hops of 64, 128, or 255 (Figure 1). Since more than 75% of paths observed by Ark monitors are fewer than 20 hops [29], we infer the smallest possible hop count given one of these three initial TTL values.

Although 86% of probed routers are within 20 hops of 64, 128, and 255 values, nearly 14% are not. To minimize distortions of estimated hop distances due to incorrectly inferred initial TTL values, we use a similar method that we used to remove outliers in latency values: we calculate the mean and the standard deviation of hop counts observed by a given Ark monitor in packets returning from all interfaces in the router set that it probed, discard values larger than the mean plus three standard deviations, and recalculate the new mean and standard deviation from the remaining values. The average standard deviation of hop count distance across the monitors is the last component of our classifying 4-D vector.

To summarize, the resulting 4-dimensional classifying vector for each geohint associated with a given router set consists of: the mean and standard deviation of estimated signal propagation speeds for all routers in this set across all Ark monitors that probed them; average standard deviation of RTTs to all routers in the set from all Ark monitors; and average standard deviation of hop count distance to all routers in the set from all Ark monitors.

3.3 Classifying geohint validity

Our next step is to evaluate which geohints are valid, i.e., statistically likely to point to a correct location. We used hostnames from six public suffixes for which we have ground truth data on hostname construction heuristics (see §4) to train Weka 3's REPTree [30] classifier with 10-fold cross-validation. We implemented the resulting classifier decision tree and classified each geohint as *likely valid* or not. We found that the two components of the 4-D vectors predominantly used for classification were the standard deviation of RTT and the standard deviation of estimated signal propagation speed.

3.4 Building general geolocation rules

Rather than reporting the result of the classifier directly (e.g., the geohint [*above.net*, SAN, 2] indicates *above.net* routers in San Diego, CA), we combined related *geohint*s to infer more general rules. For example, we surmise that if the following geohints are valid: [*above.net*, *SAN*, 2] ⇒ San Diego, CA; [*above.net*, *PAR*, 2] ⇒ Paris, France; and [*above.net*, *LAX*, 2] ⇒ Los Angeles, CA, then there is likely a general rule: hostnames used on *above.net* routers encode location using the IATA airport codes in the second position of the hostname: [*above.net*, <IATA>, 2].

To derive general rules, for each public suffix, we calculate the fraction of likely valid geohints that share the same type and position. If more than 60% of such geohints are likely valid, then we create a rule for that public suffix with the given type and position.

We then check for conflicts between generated rules, i.e., two rules that both match at least one hostname but infer different geographic locations, and reject the rule matching fewer hostnames. We discarded 6.47% of rules because of conflicts.

In total, we generated 1,711 general inference rules covering 1,398 domains. Of those, 1,126 domains matched one rule, 235 domains matched two rules, 33 matched three rules, and 4 domains matched four rules.

General rules allow us not only to match specific hints (*SAN, PAR, LAX* in the example above), but also to geolocate other routers similarly named (e.g., using IATA codes in the example above), even if we only encounter their names once and thus lack sufficient probe data for meaningful RTT and TTL statistics. General rules also incorporate measurements across geohints, increasing our confidence in the resulting inferences. Perhaps most importantly, general rules are less likely to become stale over time.

4. RESULTS AND VALIDATION

We obtained ground truth from operators responsible for 6 public suffixes who shared their router naming schemes: *akamai.com, belwue.de, cogentco.com, digitalwest.net, ntt.net, and peak10.net*. We used these data to train our classifier (see §3.3), to optimize constraints on active measurement data used for resolving geohint ambiguities, and to validate our results.

4.1 Constraining active measurement data

In order to optimize the use of measurement data and filter out statistically unreliable cases from analysis, we tested all possible combination of constraints on our measurement data. We ran classifiers for all sets of 4-D vectors obtained by varying between 1 and 5: (a) the minimum number of probes from a given Ark monitor to

	minimum number			inferences		
rank	measurements	monitors	boundary	correct	total	accuracy
1	2	2	3	16,142	16,270	99.2%
2	4	4	3	16,113	16,270	99.0%
3	4	4	4	16,113	16,270	99.0%
46	4	1	2	4,377	16,270	26.9%
47	4	2	4	4,334	16,100	26.9%
48	4	2	2	4,330	16,100	26.9%

Table 4: Three best and three worst sets of constraints on active measurement data used for geolocation, ranked by the number of routers with correctly inferred geographic hints. or correctly inferred to contain no geographic hint.

source	number of hints		used hints		matched hostnames	
major city	540,223	**78.5%**	3,392	**59.1%**	745,052	**20.6%**
IATA	7,622	**1.1%**	2,189	**38.2%**	698,741	**19.3%**
CLLI	121	**0.0%**	94	**1.6%**	632,810	**17.5%**
LOCODE	134,106	**19.5%**	42	**0.7%**	4,623	**0.1%**
ICAO	6,402	**0.9%**	18	**0.3%**	264	**0.0%**
none*					1,535,503	**42.5%**
total	688,474	**100%**	5,735	**100%**	3,616,993	**100%**

* none is the hostnames with a geographic hint that matched no rule

Table 5: For each source of possibly meaningful geographically strings (hints), we provide the number of entries in our dictionary, the number of entries used by at least one geolocating rule, and the number of hostnames that matched one of the rules produced for this source. The "none" row shows the number of hostnames that appeared to contain geographic hint(s), but did not match any geolocation rule.

constraint	nodes	fraction
number of nodes	31,790K	100.0%
with hostnames	18,956K	59.6%
with one hostname	18,890K	99.6%
with geographic hint	3,617K	19.1%
with 2+ probes	1,225K	3.9%
with 2+ monitors	920K	75.1%
with geographic hint	232K	25.2%

Table 6: ITDK (July 2013) nodes matching a given constraint. 59.6% of the nodes have at least one hostname, 20.6% of which contain a geographic hint. 3.9% of nodes were probed at least two times by at least one Ark monitor, of which 52.4% had at least two probes from at least two monitors; of this latter subset of 920K nodes, 25.2% (232K) had at least one geographic hint in one of their hostnames.

a given router (used to compute min_RTT and av_TTL); and (b) the minimum number of Ark monitors probing routers in a given router set. We also experimented with different boundaries for rejecting outliers in RTT and hop counts: 2σ, 3σ, and 4σ. Table 4 shows the three best and three worst sets of constraints ranked by the number of correct inferences they created for our ground truth data set. Over 27% of combinations had an accuracy around 99%. Thus, we selected a minimum threshold of 2 Ark monitors which each received 2 replies from a given router set[2] and the 3σ rule to discard the outliers. Table 6 shows the number of nodes in our data set matching various constraints.

4.2 Hostnames matching the rules

Table 5 illustrates the representativeness of our dictionary of geographic hints in the pool of analyzed hostnames. For the five sources of geographic strings in our dictionary, we show the numbers of: entries in the dictionary; entries used by at least one geolocation rule; and hostnames that matched a rule using this source. The character strings of major cities constitute the bulk of entries in the dictionary (540,223, or 78.4%). Although only 3,392 entries of this type were used by any rules, they produced the majority of useful hints (59.1%) geolocating 745,052 hostnames (20.5% of hostnames in the July 2013 ITDK data that appeared to have geographic hints). The dictionary contains 7,622 hints from IATA airport codes (1.1% of its entries). Of these, 2,189 were used by at least one rule, comprising 38.1% of all hints used by a generated geolocation rule. In the final classification, we geolocated 698,741 hostnames (19.3%) with IATA-based rules. Despite having only a miniscule number of CLLI codes in the dictionary (121 or 0.0175%), 94 of them were used in classifying rules, and helped geolocate nearly 632,810 hostnames (17.4% of hostnames with hints)[3]. Finally, both UN/LOCODE and ICAO hints are rarely used, helping to classify only 0.127% and 0.00729% of hostnames. Out of 3,616,616 hostnames in the July 2013 ITDK that appeared to have a geographic hint and therefore were potentially suitable for geolocation by our method, we could not find geolocation rules for 1,535,535 (42.4%, designated as "none" in Table 5). Of the 3,244 corresponding domains, we classified 1,398 (43.1%) using 1,711 rules. It is possible that some of the unclassifiable hostnames/domains belong to edge organizations who do not encode geographic information into their hostnames (the apparent hints are false), some belong to organizations who use in-house naming conventions, and we might have missed or misinterpreted some correct geographic inferences.

[2]For a data set containing only two values, the standard deviation is just the difference between either number and the average.

[3]Our dictionary might benefit from Telcordia's proprietary CLLI code database, but then we could not share the dictionary.

domain	type	positive		negative		number of hostnames
		true	false	true	false	
akamai.com		0%		100%		170
		0%	0%	100%	0%	
belwue.de	city name	52%		48%		161
		86%	14%	99%	1%	
cogentco.com	IATA	90%		10%		13,129
		99%	1%	100%	0%	
digitalwest.net	IATA	49%		51%		111
		100%	0%	98%	2%	
ntt.net	CLLI	96%		4%		2,584
		100%	0%	100%	0%	
peak10.net	IATA	100%		0%		115
		100%	0%	0%	0%	
–total–		90%		10%		16,270
		99%	1%	100%	0%	

Table 7: Comparing DRoP's results with ground truth data. "positive" means that we found a geographic hint, "true positive" means that we correctly placed this router within 10 km of its actual location, "false positive" means that we mapped the hint to a wrong location. "negative" means that no geographic hint was found, "true negative" means that operators told us these hostnames has no geographic hint; "false negative" means we failed to recognize a geographic hint.

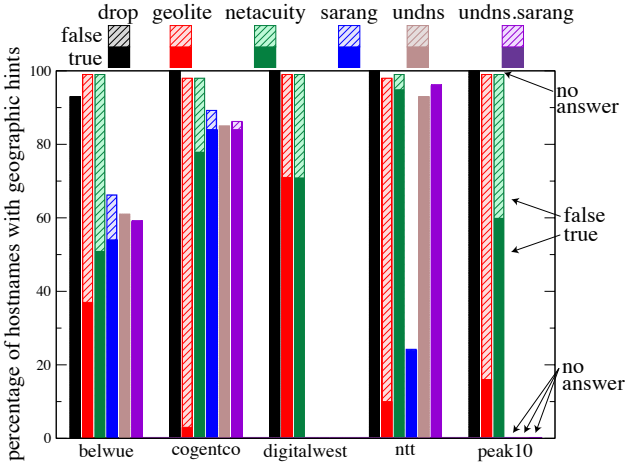

Figure 2: Validation results. To compare against Netacuity and Geolite, we limit this plot to hostnames that encode validated geographic locations, and to allow databases to agree on the location of cities with the same name we relaxed the agreement threshold from 10 km to 40 km. For these hostnames, our algorithm provided the right answer in 99% of cases. Excluding belwue.de, we got 100%. Netacuity was the most successful existing solution with 81%. Geolite correctly geolocated 12% of the hostnames.

4.3 Validating inferences for specific domains

Table 7 and Figure 2 present the results of validation. In Table 7 the top line of each domain's row shows what fraction of this domain's hostnames we inferred to contain ("positive") or not contain ("negative") geographic hints. The bottom line of each row shows: the fraction of those hostnames with recognized hints that matched ("true positive") or did not match ("false positive") the actual location of the hostnames according to our ground truth data, and the fraction of those hostnames where our algorithm did not recognize hints and operators confirmed there was no hint ("true negative"), or where we missed an actual hint ("false negative").

Not all hostnames in the July 2013 ITDK matched the naming schemas provided by the operators. First, some non-matching hostnames belonged to end hosts rather than routers. Second, our descriptions of naming conventions may be incomplete. Since most non-matching names do not contain any geographic hints, in Table 7 they contribute to "false positives": our rules suggest a geographic location for those hostnames, but we do not really know.

Akamai operators informed us how they use IATA codes in their router names, but our data contained no hostnames that matched their naming convention. Our algorithm correctly did not infer any rules for *akamai.com*, resulting in 100 true negative answers.

For all domains where our code inferred a geographic location, it was correct 94% of the time. The largest ground truth set from *cogentco.com*, containing 80% of the hostnames in our ground truth data, had a success rate of 99%. Routers of *digitalwest.net*, *ntt.net*, and *peak10.net* we geolocated with over 100% success rates when their hostnames contained geographic hints.

Figure 2 compares the success rate of our algorithm (black bars, labeled **drop** in the legend) with geolocation results from Maxmind's freely available Geolite database, Digital Envoy's Netacuity database, and iPlane's *undns* and *sarang* tools [31][4]. Geolite and Netacuity directly map IP addresses to geographic coordinates, but *undns* (and sometimes *sarang*) maps to geographic names. For uniform comparison, we mapped these names to coordinates and considered anything within 40 km to map to the same location. (We expanded the threshold defining the same location from 10 km to 40 km in order for the databases to agree on locations with the same name.) Figure 2 shows that no geolocation method provided answers for all hostnames; the height of each colored bar indicates the fraction of queried hostnames for which this method provided an answer. The lower, solid, part of each bar represents the fraction of hostnames correctly geolocated by this method; the upper, lighter, area of each bar represents the fraction of queried hostnames that the method geolocated more than 40 km away from the correct location, i.e., it failed. Figure 2 does not include hostnames that did not match the naming schemas provided by the operators ("false positives" in Table 7), since their true location is unknown.

Our algorithm correctly geolocated almost 99% of the hostnames for which we have ground truth (almost 100% if we exclude *bel-*

[4]Geolocation providers Akamai and Quova refused to sell us their services for evaluation.

wue.de which uses German strings in hostnames). Netacuity performed surprisingly well for all seven ground truth domains: an 81% success rate. In contrast, Geolite correctly geolocated only 12% of hostnames; this geolocation provider focuses on end hosts rather than routers. (Its commercial version of the database may perform better at geolocating routers.) *sarang* alone was worse than *undns* alone, with 65% and 83% success rates, respectively; combining these related tools as in [9] (and discarding conflicting cases) yielded a success rate of 84%.

We also tested our ability to use ground truth from four of these domains to accurately train the classifier for the remaining domain not used for training. When the excluded domain was *belwue.de*, the rate of true positive answers for this domain increased from 86% to 92%. In all other cases, the accuracy values shown in Table 7 dropped by less then 1% on average, providing additional confidence in our results.

5. SUMMARY OF ALGORITHM STEPS

To summarize, our algorithm involves creating two baseline data sets and four analysis processes. The first baseline data set is a dictionary of geographic hints culled from a variety of public sources (§2.1). The second is a collection of hostnames, and RTT and TTL values compiled from ITDK and raw Ark data (§2.2, §2.3). The four analysis processes are:

1. Construct *geohints* for hostnames that share public suffix, hint, and hint's position (§3.1)
   (a) Group hostnames based on a common public suffix.
   (b) Remove the public suffix, and break the remainder of the hostname into substrings of consecutive letters. Assign a position to each substring counting leftward.
   (c) Search these substrings for geographic hints.
   (d) Group routers with the same public suffix and the same geographic hint encoded in the same position in at least one of the hostnames of their interfaces into *router sets*.
   (e) For each router set create as many geohints as there are possible geographical locations for the substring defining this router set.
2. Create a 4-dimensional vector of active measurement data for each geohint (§3.2)
   (a) Find the mean and the standard deviation of min_RTTs observed between a given Ark monitor and all interfaces that this monitor probed in this router set: $\overline{min\_RTT}$ and σ_{min_RTT}, discard 3σ-outliers, recalculate $\overline{min\_RTT}$ and σ_{min_RTT}.
   (b) Using $\overline{min\_RTT}$, calculate the expected signal propagation speed s between the known location of the Ark monitor and the supposed location of the geohint.
   (c) Find the average speed $\bar{s}$ across all Ark monitors and its standard deviation σ_s.
   (d) Using the observed av_TTL and inferred initial TTL value, infer the smallest possible positive hop counts between a given Ark monitor and all interfaces that this monitor probed in this router set.
   (e) Find the mean and standard deviation of hop counts to all routers in this router set from all Ark monitors: $\bar{h}$ and σ_h, discard 3σ-outliers, recalculate $\bar{h}$ and σ_h.
   (f) For each geohint create a vector containing the following values: $\{\bar{s}, \sigma_s, \sigma_{min\_RTT}, \sigma_h\}$.
3. Classify geohint validity (§3.3)
   (a) Use the ground truth available from the operators of eight public suffixes and 20% of hostnames in those suffixes to create a training set.
   (b) With training set, create classifier of geohint validity likelihood.
   (c) Use classifier to evaluate validity of each geohint.
4. Derive general geolocation rules (§3.4)
   (a) For each public suffix, combine classified geohints into a single rule under the following conditions: they share the same geographic hint type, its position in hostnames, and more than 60% of them are classified as likely valid.
   (b) If two rules provide conflicting locations for the same hostname, reject the rule matching fewer hostnames.

6. CONCLUSIONS

We have advanced the state-of-the-art in scalable methods for accurately inferring the geographic location of Internet resources. We developed an automated process of finding geography-related strings in DNS hostnames and creating domain-specific rules for mapping such strings to actual locations.

First, we built an extensive dictionary of geographic strings and mapped them to all likely geographic coordinates. Next, we searched for those strings in observed router hostnames, and corroborated our inferences using active measurement data collected by Ark monitors. Specifically, we checked whether routers that apparently shared the same geographic string in their DNS mappings also had similar RTT values, hop counts, and estimated signal propagation speed from the same set of Ark monitors. Small variations in RTT and hop counts provided additional corroboration that routers were co-located, while small variations in propagation speed provided evidence that routers were in their inferred location.

Our method automatically generalizes inferences that yield mappings of hostnames to physical locations consistent with observed network performance characteristics into domain-specific rule sets. We resolve conflicting rules, i.e., different rules that infer different geographic locations for the same hostname, in favor of the rule that correctly places the most routers.

We generated a total of 1,711 rules covering 1,398 different domains and validated them using domain-specific ground truth about DNS naming policy we gathered for six domains. For the one public suffix that used both standard and non-standard hints, we correctly inferred the locations of only 86% of its hostnames, reducing our average overall accuracy from nearly 100% to 99% (Table 7).

Since our automated process relies on standard names compiled into our geographic string dictionary, we cannot infer the locations of routers with non-standard names by this method. We would like to explore the possibility of reverse engineering non-standard names used by some operators by integrating other geolocation approaches [5, 1], to search hostnames that share both a domain and a PoP for geographic hints they might also share. Another improvement would integrate techniques for detecting stale geographic information using inconsistencies in traceroutes [14].

In future ITDK data collections, we may conduct additional measurements aimed to increase the number of nodes that can be included in the training and classification sets for geolocating ITDK routers. We also hope to use DNS-based location inference techniques to cross-validate our IP alias resolution methods.

Acknowledgments

We would like to thank the operators who provided us with the ground truth, without which this work would not have been possible. The work was supported by U.S. NSF grant CNS-0958547, DHS S&T Cyber Security Division contract N66001-12-C-0130, and by Defence Research and Development Canada (DRDC) pursuant to an Agreement between the U.S. and Canadian governments for Cooperation in Science and Technology for Critical Infrastructure Protection and Border Security. This material represents the position of the authors and not of NSF, DHS, or DRDC.

7. REFERENCES

[1] D. Feldman and Y. Shavitt and N. Zilberman, "A structural approach for PoP geo-location," in *Computer Networks*, 2012.

[2] H. Madhyastha and T. Isdal and M. Piatek and C. Dixon and T. Anderson and A. Krishnamurthy and A. Venkataramani., "iPlane: An Information Plane for Distributed Services," in *OSDI 2006*, 2006.

[3] K. Yoshida and Y. Kickuchi and M. Yamamoto and Y. Fujii and K. Nagami and I. Nakagawa and H. Esaki, "Inferring POP-Level ISP Topology through End-to-End Delay Measurement," in *PAM 2009*, 2009.

[4] M. Arif, S. Karunasekera, S. Kulkarni, A. Gunatilaka, and B. Ristic, "Internet Host Geolocation Using Maximum Likelihood Estimation Technique," in *AINA '10: IEEE International Conference on Advanced Information Networking and Applications*, 2010.

[5] B. Eriksson and P. Barford and B. Maggs and R. Nowak, "Posit: A Lightweight Approach for IP Geolocation," in *ACM SIGMETRICS Performance Evaluation Review*, Sept 2012.

[6] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang, "Towards street-level client-independent ip geolocation," in *USENIX NSDI*, March 2011.

[7] C. Guo, Y. Liu, W. Shen, H. Wang, Q. Yu, and Y. Zhang, "Mining the Web and the Internet for Accurate IP Address Geolocations," in *IEEE INFOCOM*, 2009.

[8] D. Moore, R. Periakaruppan, J. Donohoe, and K. Claffy, "Where in the World is netgeo.caida.org?," in *INET'00: Annual Internet Society Conference*, 2000.

[9] E. Katz-Bassett, J. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards IP geolocation using delay and topology measurements," in *ACM SIGCOMM Internet Measurement Conference*, 2006.

[10] S. Laki, P. Mátray, P. Hága, I. Csabai, and G. Vattay, "A Model Based Approach for Improving Router Geolocation," *Computer Networks*, vol. 54, no. 9, 2010.

[11] J. Chabarek and P. Barford, "What's in a Name? Decoding Router Interface Names," in *ACM HotPlanet*, Auugust 2013.

[12] N. Spring and R. Mahajan and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *ACM SIGCOMM*, 2002.

[13] "**dataset**:Internet Topology Data Kit (ITDK)," July 2013. `http://www.caida.org/data/active/internet-topology-data-kit/`.

[14] M. Zhang and Y. Ruan, "How dns misnaming distorts internet topology mapping," in *In USENIX Annual Technical Conference*, 2006.

[15] "**dataset**:Archipelago Measurement Infrastructure (ark)," July 2013. `http://www.caida.org/projects/ark/`.

[16] "A Practical Guide to (Correctly) Troubleshooting with Traceroute," August 2013. `http://www.nanog.org/meetings/nanog45/presentations/Sunday/RAS_traceroute_N45.pdf`.

[17] "**dataset**:Wikipedia's page on airport codes," August 2013. `http://wikipedia.org/wiki/Airport_code/`.

[18] "Common Language Location Identifier (CLLI) code," `http://en.wikipedia.org/wiki/CLLI_code`.

[19] "**dataset**:UN/LOCODE Code List," August 2013. `http://www.unece.org/cefact/locode/service/location.html`.

[20] "**dataset**:GeoNames," August 2013. `http://www.geonames.org/`.

[21] "**dataset**:OpenFlights," August 2013. `http://openflights.org/data.html`.

[22] "**dataset**:Google Maps," August 2013. `http://maps.googleapis.com/maps/api/geocode/json?sensor=false&address=`.

[23] "Ark IPv4 Routed /24 Topology Dataset." `http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml`.

[24] "iffinder Alias Resolution Tool," 2012. `http://www.caida.org/tools/measurement/iffinder/`.

[25] K. Keys, Y. Hyun, M. Luckie, and k. claffy, "Internet-Scale IPv4 Alias Resolution with MIDAR," *IEEE/ACM Transactions on Networking*, vol. 21, Apr 2013.

[26] B. Huffaker and M. Fomenkov and k. claffy, "Internet Topology Data Comparison ," tech. rep., Cooperative Association for Internet Data Analysis (CAIDA), 2012.

[27] "Mozilla's public suffic list." `http://publicsuffix.org`.

[28] A. Acharya and J. Saltz, "A study of Internet round-trip delay," in *Technical report, University of Maryland*, 1997.

[29] "Summary statistics for all archipelago monitors." `http://www.caida.org/projects/ark/statistics/all_monitors.xml#pathlens`.

[30] "Weka 3." `http://www.cs.waikato.ac.nz/ml/weka/`.

[31] "sarang." `http://iplane.cs.washington.edu/data/sarang.tgz`.