Learning to Extract and Use ASNs in Hostnames

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

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

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

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

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

ABSTRACT

We present the design, implementation, evaluation, and validation of a system that learns regular expressions (regexes) to extract Autonomous System Numbers (ASNs) from hostnames associated with router interfaces. We train our system with ASNs inferred by RouterToAsAssignment and bdrmapIT using topological constraints from traceroute paths, as well as ASNs recorded by operators in PeeringDB, to learn regexes for 206 different suffixes. Because these methods for inferring router ownership can infer the wrong ASN, we modify bdrmapIT to integrate this new capability to extract ASNs from hostnames. Evaluating against ground truth, our modification correctly distinguished stale from correct hostnames for 92.5% of hostnames with an ASN different from bdrmapIT's initial inference. This modification allowed bdrmapIT to increase the agreement between extracted and inferred ASNs for these routers in the January 2020 ITDK from 87.4% to 97.1% and reduce the error rate from 1/7.9 to 1/34.5. This work opens a broader horizon of opportunity for evidence-based router ownership inference.

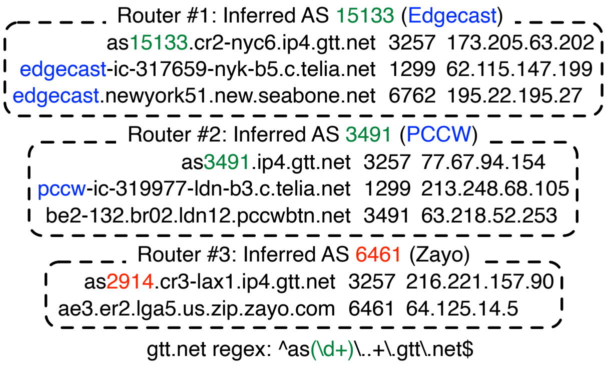
CCS CONCEPTS

• **Networks → Naming and addressing**.

KEYWORDS

Regular expression learning, Internet topology

ACM Reference Format:
Matthew Luckie, Alexander Marder, Marianne Fletcher, Bradley Huffaker, and k claffy. 2020. Learning to Extract and Use ASNs in Hostnames. In *ACM Internet Measurement Conference (IMC '20), October 27–29, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3419394.3423639


Router #1: Inferred AS 15133 (Edgecast)
Router #2: Inferred AS 3491 (PCCW)
Router #3: Inferred AS 6461 (Zayo)
gtt.net regex: ^as(\d+)\..+\.gtt\.net$

```
          as15133.cr2-nyc6.ip4.gtt.net  3257  173.205.63.202
 edgecast-ic-317659-nyk-b5.c.telia.net  1299  62.115.147.199
   edgecast.newyork51.new.seabone.net  6762  195.22.195.27

             as3491.ip4.gtt.net  3257  77.67.94.154
 pccw-ic-319977-ldn-b3.c.telia.net  1299  213.248.68.105
  be2-132.br02.ldn12.pccwbtn.net  3491  63.218.52.253

   as2914.cr3-lax1.ip4.gtt.net  3257  216.221.157.90
 ae3.er2.lga5.us.zip.zayo.com  6461  64.125.14.5
```

Figure 1: Border routers with interface IP addresses routed and named by their supplying AS. These hostnames allow researchers and operators to reason about router-level interdomain connectivity.

1 INTRODUCTION

Identifying the Autonomous System (AS) that operates a router is critical to understand connectivity within and between organizations. For example, recent work has examined patterns of congestion between ASes [5, 9, 34], connectivity of cloud providers [35, 36], macroscopic impacts of submarine cable deployments [8], and load balanced paths within and between ASes [27, 29]. Because there is no database that stores the AS that operates every router, researchers have developed and refined heuristic-based methods to infer the AS that operates a router using topological constraints from traceroute and BGP paths [12, 18, 21, 22, 24–26]. However, router-level Internet connectivity is complex, techniques for inferring router-level connectivity have limited capability, and obtaining ground truth at scale to inform heuristic development is difficult, restricting the accuracy of these methods.

To aid network management, some network operators encode the AS that operates a router in DNS hostname strings. This AS annotation is important for attribution, as when two ASes interconnect, one AS supplies an IP address to its neighbor to facilitate connectivity. This IP address is registered to the supplying AS in WHOIS, and typically originated in BGP by the supplying AS, so naïve interpretations of who operates a router that use these sources of data can mislead operators and researchers. Figure 1 provides examples of border routers with interface IP addresses routed and named by their supplying AS, as well as the ASNs that one heuristic-based method, bdrmapIT [24], inferred for the routers. The operators supplying IP addresses for these router interfaces have different conventions for assigning hostnames to these IP addresses; gtt.net embeds the ASN, while telia.net and seabone.net embed the *AS name*. In this paper, we implement a method to automatically learn regular expressions (regexes) that extract these ASN annotations. This paper makes four contributions:

IMC '20, October 27–29, 2020, Virtual Event, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8138-3/20/10…$15.00
https://doi.org/10.1145/3419394.3423639

(1) We introduce and validate a method that infers regexes that extract AS numbers from hostnames. Our method finds patterns in hostnames that embed an ASN, including necessary literals and character classes in regexes that capture these patterns. We validated our inferred regexes with five operators, who reported that we captured their naming intent, though in one case our regex was too specific because of stale hostnames. We implemented our method in the Hoiho tool [19], and publicly release our source code implementation as part of scamper [16].

(2) We demonstrate the utility of our algorithm by applying it to 19 sets of training data across 10 years. We used the 17 Internet Topology Data Kit (ITDK [1]) snapshots built by CAIDA between July 2010 and January 2020 that include routers annotated with AS inferences using the RouterToAsAssignment [12] (2010–2017) or bdrmapIT [24] (2017–2020) heuristic-based router ownership inference methods – as well as two PeeringDB snapshots – to build regexes for 206 suffixes. We publicly release the training data and inferred regexes on a website that shows how these regexes applied to training data [20].

(3) We modify bdrmapIT to evaluate extracted ASNs when inferring router ownership. We consider the possibilities that the hostname is stale, or the bdrmapIT-inferred AS was wrong (consider the discrepancy for the third router in figure 1). We extend bdrmapIT to consider the extracted ASNs in the context of the topological constraints it gathers when inferring the AS that operates a router. For the January 2020 ITDK, our modification distinguished stale from correct hostnames for 92.5% of hostnames incongruent with bdrmapIT's initial inference, increasing agreement between the extracted and inferred ASNs from 87.4% to 97.1% and reduce the error rate from 1/7.9 to 1/34.5 for these routers. We publicly release our modifications to bdrmapIT [23].

(4) We establish a new source of validation data for router-level interconnection mapping. Obtaining ground truth at scale is fundamentally intractable; prior work (§2.1) validated methods on at most 7 networks, mostly Tier-1 and R&E networks, triggering concerns about generalizability of methods. We inferred 90 regexes from the January 2020 ITDK for networks of diverse geography, size, and class, offering future methods a promising approach to validation data that can be shared.

2 BACKGROUND AND RELATED WORK

2.1 Inferring Router Ownership

In 2003, Mao *et al.* developed heuristics to improve IP-to-AS mappings for router interfaces [22], changing IP-to-AS mappings derived from BGP routing so that traceroute-inferred AS paths were more congruent with BGP AS paths for corresponding prefixes. They used hostnames to infer the AS for named but unrouted interfaces, assigning the same AS mapping as a neighboring routed interface with the same suffix. However, this heuristic is unreliable because the supplying AS assigns hostnames to the IPs they provide to their neighbor for interconnection (figure 1). In 2004, Mao *et al.* used a dynamic programming technique to change IP-to-AS mappings at a /24 prefix granularity using co-located BGP and traceroute views [21]. However, operators typically use /30 or /31 prefixes (rather than /24s) for private interconnection between networks to use address space efficiently.

In 2010, Huffaker *et al.* evaluated router ownership heuristics that used router alias resolution, AS relationships, and degrees [12]. The best-performing heuristic they evaluated was to choose the AS that announced the longest matching prefix for the most interfaces on the router in BGP (election), breaking ties by choosing the smaller of the ASes with interfaces on the router (degree). They validated using data from a Tier-1 network, a Tier-2 network, as well as five research and education networks, and reported that 80% of the AS inferences were correct when the inferred routers had interfaces from multiple ASes, but reduced to 71% when including routers with interfaces in a single AS, likely because traceroute usually only observes the provider-supplied (and BGP-announced) address for border routers of stub ASes. They publicly released their source code implementation, which they called RouterToAsAssignment. Twelve ITDKs built between July 2010 and February 2017 used this technique to infer router ownership.

In 2016, Luckie *et al.* and Marder *et al.* built the bdrmap [18] and MAP-IT [26] techniques, respectively. The bdrmap technique focused on finding all neighbor routers forming an interdomain link with an AS hosting a traceroute vantage point (VP), that are observable from that VP. From a single VP, bdrmap conducts traceroute to every routed prefix, and uses a set of heuristics in conjunction with AS relationships and BGP routing information to infer ownership of the neighbor routers. Luckie *et al.* validated bdrmap's router ownership inferences using ground truth provided by a Tier-1 network, one large and one small access network, and a research and education (R&E) network, reporting that ≈97.1% of inferred ASes corresponded to the organization operating the router. MAP-IT used a graph refinement technique to infer owners for all routers observed in the *middle* of a traceroute path, in a collection of traceroutes. In validating MAP-IT, Marder *et al.* focused on IP addresses they inferred were used for AS interconnection for two Tier-1 networks. They manually interpreted hostnames those operators assigned, reporting that 95.0% of their interconnection inferences were correct. They also obtained ground truth from a R&E network, reporting that all interconnection inferences were correct.

In 2018, Marder *et al.* built bdrmapIT [24], incorporating bdrmap's heuristics, such as those for inferring ownership of routers only observed at the edge of traceroutes where there are no adjacent routers from which to reason about ownership, with MAP-IT's graph refinement, to extend bdrmap's router ownership inference heuristics outside the VP network. They validated using ground truth from a Tier-1 network, a large access network, and two R&E networks, reporting that ≈95.3% of their AS interconnection IP address inferences correctly identified the operating ASes on each side of the interconnections. Five ITDKs built between August 2017 and January 2020 used bdrmapIT to infer router ownership.

These existing heuristic-based methods are not suitable for inferring ownership of routers observed in one-off or limited sets of traceroutes, as they rely on accumulating constraints from a large set of traceroutes for their accuracy. Our method allows researchers to benefit from these previous methods without requiring a large set of traceroutes. In addition, our method enables corroboration and validation of inferences made using these heuristic-based methods. Importantly, since validation of previous methods has used proprietary ground truth data, our method yields validation data (hostname annotations) that can be shared.

2.2 Topology Information in Hostnames

Researchers and operators use hostnames in DNS pointer (PTR) records to understand router-level topology. Prior work has used these hostnames to infer and validate router ownership [12, 22, 26], geolocation [11, 13, 31, 33], physical properties [2, 6, 7, 10], and which interfaces belong to the same router [14, 17, 19, 32]. Using hostnames to infer router-level properties of networks is challenging, as each operator independently decides on a naming convention for their suffix, leading researchers to manually build regexes to capture structure unique to each suffix [10, 12, 14, 17, 33]. Further, hostnames can have typos and become stale [37].

More recently, researchers have applied machine learning approaches to automatically infer regexes that extract information from hostnames. In 2014, Huffaker *et al.* built DRoP [13] to infer regexes that extract apparent geolocation strings from router hostnames. In 2019, Luckie *et al.* built the Holistic Orthography of Internet Hostname Observations (Hoiho) tool [19] to infer the portion of a hostname that embeds a *router name* shared among interfaces on the same router, but unique across routers in the suffix, to infer router aliases. That work learned regexes that increased in specificity and completeness over the course of eight stages, and showed that it is possible to infer likely aliases using hostnames and regexes that prior techniques missed [19]. In this work, we modify Hoiho to learn to infer the portion of a hostname that embeds the ASN of the network that operates the router, and introduce techniques to detect hostnames that contain stale ASN annotations, so that heuristic-based router ownership inference methods can automatically evaluate correct ASN annotations. More broadly, the measurement community will be able to more confidently characterize router-level Internet structure if the community is able to automatically use information encoded in hostnames.

3 BUILDING CONVENTIONS

Our algorithm learns if an operator uses a naming convention (NC) that embeds an ASN in a hostname, by evaluating automatically generated candidate regexes against sets of router hostnames with the same suffix. We determine suffixes using the Mozilla public suffix list [28], which lists effective top-level domains (e.g. .com, .org.nz) under which operators can register their own domain suffixes (e.g. example.com, luckie.org.nz). We annotate each router with a *training ASN* inferred heuristically (§2.1) or recorded by an operator in PeeringDB. If a regex extracts different ASNs for different hostnames in the same suffix, and the extracted ASNs are congruent with training ASNs, we infer that each hostname embeds the ASN that operates the router, as we do for the example in figure 1. A regex must extract multiple different ASNs congruent with training data, because some operators embed the ASN that supplies the address, even for addresses they supply to neighbor routers, illustrated by the example in figure 2.

3.1 Evaluating and Ranking Regexes

A hostname contains an *apparent ASN* if it contains a numeric string (number) congruent with the training ASN for the router with that hostname. Hoiho evaluates regexes using the following per-hostname classifications. Hoiho assigns a true positive (TP) when a regex extracts a number congruent with the training ASN

training ASN	hostname (PTR record)
15576	ge0-2.01.p.ost.ch.as15576.nts.ch
15576	lo1000.01.lns.czh.ch.as15576.nts.ch
15576	te0-0-24.01.p.bre.ch.as15576.nts.ch
44879	01.r.cba.ch.bl.cust.as15576.nts.ch
51768	02.r.czh.ch.sda.cust.as15576.nts.ch
206616	01.r.cbs.ch.wwc.cust.as15576.nts.ch

as(\d+)\.nts\.ch$

Figure 2: Example suffix that labels the AS that supplies the address, not the AS that operates the router.

training ASN	hostname (PTR record)
701	201.atm2-0.vr1.tor2.alter.net
855	te-4-0-0-85.53w.ba07.mctn.nb.aliant.net
6057	mlg4bras1-be127-605.antel.net.uy
20940	as24940.akl-ix.nz
205073	as202073.swissix.ch
207032	gw-as20732.init7.net

(a) Apparent ASNs can have an edit distance of one from their training ASN, by coincidence or typo.

122	50-236-216-122-static.hfc.comcastbusiness.net
209	209-201-58-109.dia.stat.centurylink.net
209	209-206-252-105.stat.centurytel.net

(b) Hostnames can embed an IP address, with portions the same as the training ASN, by coincidence.

Figure 3: Example hostnames with apparent ASNs, which differ from the training ASN by a single digit (a), or were derived from an IP address (b).

for that hostname. Some hostnames contain an apparent ASN that differs from the training ASN with a Damerau-Levenshtein edit distance of one [4, 15] possibly suggesting a typo. Figure 3a shows sample hostnames with apparent ASNs that have an edit distance of one from the training ASNs, some of which are typos, and others by coincidence. Hoiho therefore also assigns a TP when the first and last characters of the training ASN and extracted number are the same, and both numbers are at least three digits in length, to avoid inferring a TP where the training ASN and extracted number have an edit distance of one by coincidence; Hoiho found seven hostnames with typos in the January 2020 ITDK. Hoiho otherwise assigns a false positive (FP) when a regex extracts a different number than the training ASN, or when the extracted number is part of an IP address embedded in the hostname, illustrated in figure 3b. Finally, Hoiho assigns a false negative (FN) when a regex does not extract a number from a hostname and there is an apparent ASN in the hostname congruent with the training ASN.

Our metric for ranking regexes, which we call Absolute True Positives (ATP), is TP-(FP+FN). Our ATP metric includes FNs because our goal is to find a regex that matches as many hostnames as possible, rather than find a regex that has high positive predictive value (PPV, TP/(TP+FP)) for a subset of interfaces. Hoiho used a different definition of ATP when it evaluated regexes to infer router names [19], penalizing regexes that separated hostnames from routers (one class of FN), but did not penalize regexes that did not match routers with aliases (a second class of FN) because not all operators embed a router name in their hostnames.

training ASN	hostname (PTR record)	
109	109.sgw.equinix.com	(a)
714	714.os.equinix.com	(b)
714	714.me1.equinix.com	(c)
714	p714.sgw.equinix.com	(d)
714	s714.sgw.equinix.com	(e)
24115	p24115.mel.equinix.com	(f)
24115	s24115.tyo.equinix.com	(g)
22282	22822-2.tyo.equinix.com	(h)
24482	24482-fr5-ix.equinix.com	(i)
54827	54827-dc5-ix2.equinix.com	(j)
55247	55247-ch3-ix.equinix.com	(k)
2906	netflix.zh2.corp.eu.equinix.com	(l)
19324	ipv4.dosarrest.eqix.equinix.com	(m)
8075	8069.tyo.equinix.com	(n)
8075	8074.hkg.equinix.com	(o)
55923	45437-sy1-ix.equinix.com	(p)

		TP	FP	FN	ATP
	Phase 1: Generate Base Regexes				(§3.2)
#1	^(\d+)\.[^\.]+\.equinix\.com$	a, b, c	n, o	d, e, f, g, h, i, j, k	-7
#2	^p(\d+)\.[^\.]+\.equinix\.com$	d, f		a, b, c, e, g, h, i, j, k	-7
#3	^s(\d+)\.[^\.]+\.equinix\.com$	e, g		a, b, c, d, f, h, i, j, k	-7
#4	^(\d+)-.+\.equinix\.com$	h, i, j, k	p	a, b, c, d, e, f, g	-4
	Phase 2: Merge Regexes				(§3.3)
#5	^(?:pls)?(\d+)\.[^\.]+\.equinix\.com$	a, b, c, d, e, f, g	n, o	h, i, j, k	1
	Phase 3: Embed Character Classes				(§3.4)
#6	^(?:pls)?(\d+)\.[a-z\d]+\.equinix\.com$	a, b, c, d, e, f, g	n, o	h, i, j, k	1
	Phase 4: Build Regex Sets				(§3.5)
#7	^(?:pls)?(\d+)\.[a-z\d]+\.equinix\.com$ ^(\d+)-.+\.equinix\.com$	a, b, c, d, e, f, g, h, i, j, k	n, o, p		8

Figure 4: Inferring a naming convention (NC) for the Equinix suffix (equinix.com) across four phases. The regexes that form the NC increase in specificity and coverage through each phase.

3.2 Generate Base Regexes

Our algorithm consists of four stages, illustrated in figure 4, which increase specificity and coverage as the algorithm proceeds. In the first phase, Hoiho builds base regular expressions that extract ASNs focusing on structure encoded in the hostname using punctuation characters. This phase recursively builds base regexes that capture the apparent ASN with (\d+) and use regex components that exclude specific punctuation depending on the punctuation at the beginning and end of each portion (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, similar to prior work [19]. For example, Hoiho builds ^(\d+)-[^-]+-[^\.]+\.equinix\.com$, ^(\d+)-[^-]+-[^-]+\.equinix\.com$, and ^(\d+)-.+\.equinix\.com$ for hostname i in figure 4. For the portion that contains the apparent ASN, we extended Hoiho to embed sequences of alphanumeric characters around the ASN delimited by punctuation, in accordance with common operational practice, in the base regexes. For example, regex #2 in figure 4 Hoiho embeds "p" in ^p(\d+)\.[^\.]+\.equinix\.com$ as "p" appears in the same punctuation-delimited portion as the ASN in hostnames d and f.

3.3 Merge Regexes

In the second phase, Hoiho examines the set of regexes for each suffix that are very similar, i.e., they differ by a single simple string. For example, the first three regexes in figure 4 all have ^ and (\d+)\.[^\.]+\.equinix\.com$ in common, but regexes #2 and #3 additionally contain the strings "p" and "s". Hoiho merges these three regexes by embedding an *or* statement into the portion of the regex that differs – (?:p|s), so that a single regex will match hostnames with a "p" or "s" prior to the ASN. In this case, regex #1 does not embed any characters between the portion of the regex that is in common, indicating these characters are optional. Hoiho represents this optionality by appending a question mark at the end of the *or* statement – (?:p|s)? – to compose ^(?:p|s)?(\d+)\.[^\.]+\.equinix\.com$. Prior work in Hoiho [19] did not merge similar regexes. This improvement generalizes when building regexes for alias resolution.

3.4 Embed Character Classes

In the third phase, Hoiho identifies character class sequences in common for matched hostnames, replacing the components that exclude specific punctuation built in §3.2 (e.g., [^\.]+ and [^-]+) with components that specify character classes. For example, the component [^\.]+ for regex #5 in figure 4 matches alphabetical characters for hostnames a-g, although hostname c also contains a digit. Therefore, Hoiho replaces [^\.]+ with [a-z\d]+, which matches a sequence of alphanumeric characters, to build regex #6. This phase is the same as the approach in prior work [19].

3.5 Build Regex Sets

In the final phase, Hoiho builds sets of regexes to form a NC in order to increase coverage where the operator uses multiple formats. Hoiho ranks regexes by ATP (descending) and evaluates the outcome of combining a regex with each of the regexes below it in the rank order. Hoiho includes an expanded regex in its working set if the ATP is greater than the regex it started with. For example, Hoiho combines regexes #4 and #6 to build NC #7 in figure 4 because regexes #4 and #6 match different hostname formats. Unlike prior work [19] Hoiho does not require the PPV of the new inferences to be similar or better than the PPV of the regex it started with, as our goal is to identify discrepancies between training and embedded ASNs, where the training ASN might be wrong. Appendix A discusses the distinction between merging regexes (§3.3) and building regex sets (§3.5).

3.6 Select Best Convention

Hoiho ranks NCs by ATP, and initially select the highest ranked NC as the best, even if a regex below had a higher PPV. Then, Hoiho considers NCs with a lower ATP but expressed in fewer regexes. Hoiho selects a lower-ranked NC if it matches at least as many hostnames as the current best NC, has at least as many TPs, and no more than one additional FP. Because these NCs are made of fewer regexes, there is less opportunity to choose a regex that is the result of over-fitting to the training data.

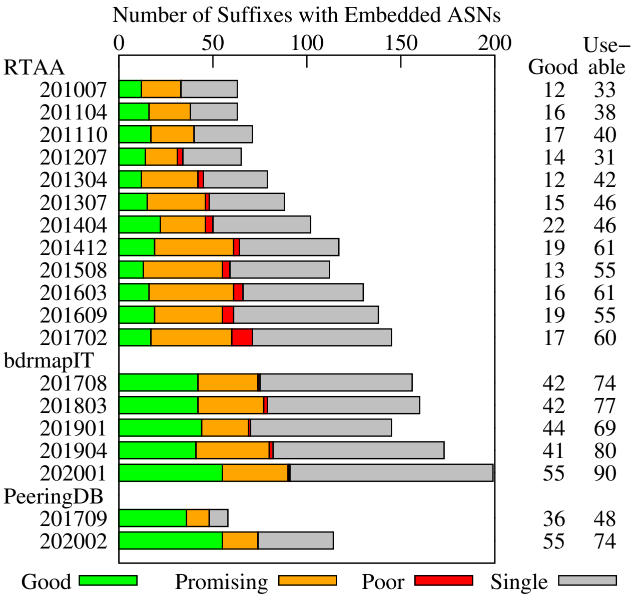


Figure 5: Classification of NCs. The number of NCs embedding ASNs is growing over time.

4 EVALUATING CONVENTIONS

We applied our method to 17 ITDKs assembled by CAIDA between July 2010 and January 2020 that contain IPv4 routers annotated with an ASN inferred with RouterToAsAssignment (RTAA) or bdrmapIT (§2). We also applied our method to two sets of training data that used ASNs recorded by operators in PeeringDB. Hoiho classified a NC as *good* if it extracted at least three unique ASNs congruent with training ASNs with a PPV ≥ 80%, and a NC as *promising* if it extracted at least two unique congruent ASNs with a PPV ≥ 50%. The good and promising NCs are *usable* because they usually extract a congruent ASN. Hoiho classified a NC as *single* if it extracted congruent ASNs belonging to a single organization, as in figure 2. The remaining NCs with a PPV < 50% are *poor*.

Figure 5 shows that Hoiho classified 12 – 55 NCs per ITDK as good, with the number of good NCs growing over time. The growth of good NCs is a combination of three factors: improvement in heuristic methods, increasing numbers of suffixes that embed ASNs, and an increasing number of VPs that provide visibility of interfaces labeled with ASNs. Hoiho also inferred 55 good NCs for the February 2020 PeeringDB snapshot. Hoiho inferred usable NCs for 206 suffixes across all 19 sets of training data. We obtained ground truth from operators for five NCs we inferred for the January 2020 ITDK. The operators confirmed the regexes reflected their NC, though one (poor) NC was too specific because of stale hostnames.

We manually investigated the relationship between the extracted ASNs and the suffixes for the 108 single NCs for the January 2020 ITDK, establishing that 79.5% of the suffixes belonged to the organization with the ASN (e.g. nts.ch in figure 2). 82.5% of the 40 single NCs for the February 2020 PeeringDB snapshot are for IXPs that allow the member to assign the hostname for the IXP address, using their own suffix. ≈12.5% of the suffixes matched a provider, peer, or IXP used by the extracted ASN; for ≈7.5% the training ASN appeared to coincidentally be the hostname.

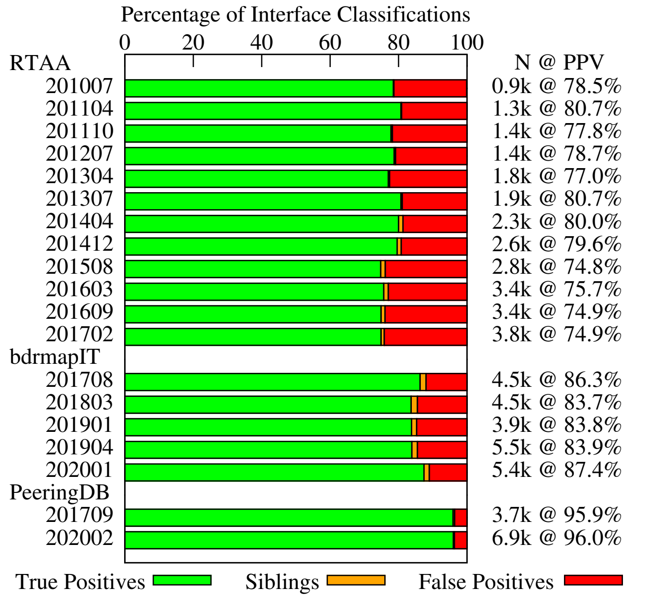


Figure 6: Evaluation of *usable* NCs on training data. The agreement between training and extracted ASNs increases as methods to infer operators improve.

	Usable	Single
Simple – ^as(\d+)\.example\.com$	17.7%	4.6%
Start – ^as(\d+)\.[a-z]+\.example\.com$	50.8%	23.1%
End – ^[a-z\d]+\.as(\d+)\.example\.com$	10.8%	43.1%
Bare – (\d+)\.[a-z]+\d+\.example\.com$	5.4%	7.7%
Complex	15.4%	21.5%

Table 1: Taxonomy of how and where operators embedded ASNs in hostnames. Operators that labeled the neighbor ASN usually placed it at the start of the hostname.

Figure 6 shows the growing congruity between the training ASNs that were heuristically inferred in the ITDK, and ASNs extracted by the usable regexes; the PPV for ASNs inferred using RouterToAsAssignment is 74.8% – 80.7% and 83.7% – 87.4% for bdrmapIT. Some extracted ASNs differ from their training ASN because the training ASN is a sibling AS of the extracted ASN: including these siblings increased the PPV for RouterToAsAssignment inferences by ≈1%, and bdrmapIT inferences by ≈2%. The PPV for the ASNs stored by operators in PeeringDB was 96.0%; we therefore hypothesize that while some hostnames were stale, more were correct, and these heuristic methods inferred the wrong ASN. In §5, we describe modifications we made to bdrmapIT to incorporate extracted ASNs.

The January 2020 ITDK and February 2020 PeeringDB training sets were complementary: in total, we observed 130 usable NCs for different suffixes, with 34 in common – i.e., IXPs observed in both ITDK and PeeringDB, as well as 56 ISPs unique to the ITDK and 40 IXPs unique to PeeringDB. Of the 34 common suffixes, 24 had exactly the same regexes; the 10 that differed varied in specificity, with our method inferring a less specific regex for larger training sets that captured the diversity present in the hostnames.

We characterized how and where operators embedded ASNs in the 130 usable NCs, summarizing our results in table 1. An operator with a *simple* NC embedded only a neighbor ASN prefaced with "as" in the hostname. The NCs we classified as *start* and *end* embedded

	Correct ASN		Incorrect ASN	
	Used (TP)	Not Used (FN)	Used (FP)	Not Used (TN)
Transit Provider	6	0	0	0
European ISP	4	0	0	0
Large ISP	4	0	2	36
Regional US IXP	6	0	0	1
Asia-Pacific IXP	3	1	0	1
PeeringDB (23)	322	26	6	49
Total:	345	27	8	87
	372		95	

Table 2: Summary of validation. Our modifications to bdrmapIT made the correct decision for 432 of 467 (92.5%) of extracted ASNs that were incongruent with bdrmapIT's initial heuristic inference.

the neighbor ASN prefaced with "as" at the start or end of the hostname, and embedded additional information in the hostname, such as the bandwidth at which the neighbor is connected, or the name of the neighbor. The NCs we classified as *bare* embedded the neighbor ASN but did not preface the ASN with any alphabetic characters. Finally, the NCs we classified as *complex* embedded the neighbor ASNs in the middle of the hostname, or used a different annotation other than "as", or required multiple regexes to capture the convention. Most operators (68.5%) placed the neighbor ASN at the start of the hostname; in contrast, 43.1% of operators that only embedded their own ASN placed it at the end of the hostname.

5 USING CONVENTIONS IN bdrmapIT

When processing a router-level graph, bdrmapIT [24] builds topological state for router nodes, annotating each router node with *subsequent ASNs* for adjacent routers in traceroute paths, and *destination ASNs* for which the router was in a traceroute path. bdrmapIT usually uses the subsequent ASNs to reason about which ASN operates a given router. For routers where there were no subsequent routers, bdrmapIT usually uses the destination ASNs to reason about which ASN operates the router. We modified bdrmapIT to evaluate ASNs extracted from hostnames as part of bdrmapIT's inference approach, to determine if the ASNs appeared reasonable; i.e., were not typos or stale. Our modified bdrmapIT inferred an extracted ASN was reasonable if it matched, or was a sibling of, an ASN in either the subsequent or destination ASN sets, or the extracted ASN is a provider of one of the ASes in these sets.

We supplied all of the good, promising, and poor NCs Hoiho inferred from the January 2020 ITDK to our modified bdrmapIT, and then re-processed that ITDK. Compared to the initial ITDK, the agreement between the inferred and extracted ASNs for the routers with ASN annotations increased from 87.4% to 97.1%. and the error rate reduced from 1/7.9 to 1/34.5. Of the 723 router interfaces with ASN mappings different from the ASNs extracted from hostnames, our modified bdrmapIT used the extracted ASN for 526 (72.8%) and was most likely to use ASNs extracted using good NCs than other classes; it used 82.5% of 570 extracted ASNs from good NCs, 44.0% of 109 from promising NCs, and 18.2% of 44 from poor NCs.

We obtained ground truth for which ASN operated a given router from five operators, which we used to determine if the hostname contained the correct ASN. We also cross-validated with ASNs recorded by operators in PeeringDB for interface IP addresses with hostnames in 23 different suffixes; we excluded nine router interfaces where the training ASN, extracted ASN, and ASN in PeeringDB were all different, because it was not clear if any of the ASNs were correct. In total, this validation data covered 467 of 723 (64.6%) hostnames where the extracted ASN was different from the ASN initially inferred by bdrmapIT. Our modified bdrmapIT made the correct decision to use the ASN in 92.5% of these hostnames. Table 2 shows that 372 hostnames in our validation data contained the correct ASN, and our modified bdrmapIT used 345 (92.7%) of those hostnames. There were also 95 interfaces with incorrect hostnames in our validation data, and our modified bdrmapIT used eight (8.4%). Five FPs were because the operator recorded their main ASN in PeeringDB (e.g., Microsoft AS8075) but the IXP operator embedded a sibling ASN in the hostname (e.g., Microsoft ASes 8069 or 12076). The extracted ASNs were coincidentally providers of the actual ASN for the remaining three.

6 LIMITATIONS

Zhang et al. established in 2006 that because operators do not necessarily maintain hostnames in DNS, Internet topology mapping efforts using hostnames can be distorted [37]. Errors in hostnames can impact the accuracy of router ownership inferences using our regexes. Therefore, we recommend that researchers use our regexes in conjunction with topological checks as we did in §5.

Our method relies on training data that is usually correct for routers with interfaces in a given suffix – i.e., the training ASNs are usually correct and the hostnames are not stale. Outside of IXPs, where some operators record their public peering interface addresses in PeeringDB, we rely on heuristic-based methods to infer training ASNs. Our method may not infer a usable NC if the heuristic-based methods perform poorly for the routers covered by a suffix, or the hostnames in a suffix are stale.

7 FUTURE DIRECTIONS

We have shown that it is possible to automatically learn regexes that extract ASNs from hostnames, and that we can use these ASNs to improve router ownership inferences that are critical for attribution. A challenging direction for further work is to learn regexes that extract *AS names*, such as those found in figure 1, without relying on a dictionary of AS names. In our preliminary investigation, at least 3x more suffixes embed AS names than AS numbers in hostnames. Obtaining this capability could result in a large, heterogeneous source of validation data, showing exciting promise for evidence-based router ownership inference techniques.

Our regexes have also value in identifying router and AS-level interconnections that are not revealed by current measurement infrastructure. In our preliminary investigation using the OpenINTEL PTR lookup of all delegated IP address space [30], we increased the number of hostnames matching a regex inferred from the January 2020 ITDK from 5.4K to 22.5K. While some of these hostnames could be stale [37], these hints could inform deployment of new measurement infrastructure to determine which interdomain links exist [3], improving our capability to measure the evolution of the Internet interconnection ecosystem.

```
#7    ^(?:p|s)?(\d+)\.[a-z\d]+\.equinix\.com$
      ^(\d+)-.+\.equinix\.com$

#7a   ^(?:p|s)?(\d+)(?:\.[a-z\d]+|-.+)\.equinix\.com$

      ^(\d+)\.[a-z\d]+\.equinix\.com$
#7b   ^p(\d+)\.[a-z]+\.equinix\.com$
      ^s(\d+)\.[a-z]+\.equinix\.com$
      ^(\d+)-.+\.equinix\.com$
```

Figure 7: Equivalent naming conventions (NCs) using the example routers in figure 4. We argue that NC #7 captures the diversity in the hostnames without unnecessary complexity.

A MERGING REGEXES VS. REGEX SETS

The method described in §3 contains two stages that are functionally similar: merging individual regexes that differ by a single simple string (§3.3) and building regex sets so that a NC contains multiple regexes each of which match a subset of the hostnames (§3.5). For example, NC #7 in figure 4, duplicated in figure 7, could have been presented differently (e.g. #7a and #7b in figure 7). Our overall goal with Hoiho is to infer regexes that a human might have built, rather than regexes that are either overfitted to the training data, or so complex that a human would have used multiple simpler regexes.

At one extreme, we might merge the two regexes that make up NC #7, structuring the functionally different portion in a second or statement, as in NC #7a in figure 7. It is our judgment that NC #7a is one that an operator or user might find difficult to reason about, and that a human building a regex would probably avoid using multiple "or" statements. We suggest that this regex is over-fitted, and in fact the NC is better expressed with two crisp regexes (NC #7), rather than one. At the other extreme, we might present the NC using four individual regexes, as in NC #7b in figure 7. The first three regexes each individually make minor contributions to NC #7b: using the hostnames in figure 4, the first regex extracts three ASNs congruent with training data, while the second and third extract two each. We suggest that the first three regexes in NC #7b are so similar that a human building a regex would probably merge these three regexes using a single "or" statement.

ACKNOWLEDGMENTS

We thank Young Hyun and Ken Keys for assistance with the ITDK, and the anonymous reviewers for their helpful comments. This work was supported by NSF OAC-1724853, and by the Department of Homeland Security (DHS) Science and Technology Directorate, Cyber Security Division via contract 70RSAT18CB0000013, but this paper represents only the position of the authors.

REFERENCES

[1] CAIDA. 2020. Macroscopic Internet Topology Data Kit (ITDK). https://www.caida.org/data/internet-topology-data-kit/.

[2] Joseph Chabarek and Paul Barford. 2013. What's in a Name? Decoding Router Interface Names. In *HotPlanet*. 3–8.

[3] Kai Chen, David R. Choffnes, Rahul Potharaju, Yan Chen, Fabian E. Bustamante, Dan Pei, and Yao Zhao. 2009. Where the Sidewalk Ends: Extending the Internet AS Graph Using Traceroutes From P2P Users. In *CoNEXT*. 217–228.

[4] Fred J. Damerau. 1964. A Technique for Computer Detection and Correction of Spelling Errors. *Commun. ACM* 7, 3 (March 1964), 171–176.

[5] Amogh Dhamdhere, David D. Clark, Alexander Gamero-Garrido, Matthew Luckie, Ricky K.P. Mok, Gautam Akiwate, Kabir Gogia, Vaibhav Bajpai, Alex C. Snoeren, and k claffy. 2018. Inferring Persistent Interdomain Congestion. In *SIGCOMM*. 1–15.

[6] Ramakrishnan Durairajan, Paul Barford, Joel Sommers, and Walter Willinger. 2015. InterTubes: A Study of the US Long-haul Fiber-optic Infrastructure. In *SIGCOMM*. 565–578.

[7] Ramakrishnan Durairajan, Joel Sommers, and Paul Barford. 2014. Layer 1-Informed Internet Topology Measurement. In *IMC*. 381–394.

[8] Rodérick Fanou, Bradley Huffaker, Ricky Mok, and k claffy. 2020. Unintended Consequences: Effects of Submarine Cable Deployment on Internet Routing. In *PAM*. 211–227.

[9] Rodérick Fanou, Francisco Valera, and Amogh Dhamdhere. 2017. Investigating the causes of congestion on the african IXP substrate. In *IMC*. 57–63.

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

[11] 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.

[12] Bradley Huffaker, Amogh Dhamdhere, Marina Fomenkov, and kc claffy. 2010. Toward Topology Dualism: Improving the Accuracy of AS Annotations for Routers. In *PAM*.

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

[14] 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.

[15] Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10, 8 (Feb. 1966), 707–710.

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

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

[18] Matthew Luckie, Amogh Dhamdhere, Bradley Huffaker, David Clark, and k claffy. 2016. bdrmap: Inference of Borders Between IP Networks. In *IMC*. 381–396.

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

[20] Matthew Luckie, Alexander Marder, Marianne Fletcher, Bradley Huffaker, and k claffy. 2020. Data supplement for "Learning to Extract and Use ASNs in Hostnames". https://www.caida.org/publications/papers/2020/hoiho/.

[21] Z. Morley Mao, David Johnson, Jennifer Rexford, Jia Wang, and Randy Katz. 2004. Scalable and Accurate Identification of AS-Level Forwarding Paths. In *INFOCOM*.

[22] Z. Morley Mao, Jennifer Rexford, Jia Wang, and Randy H. Katz. 2003. Towards an accurate AS-level traceroute tool. In *SIGCOMM*. 365–378.

[23] Alexander Marder. 2020. bdrmapIT source code repository. https://github.com/alexmarder/bdrmapit.

[24] Alexander Marder, Matthew Luckie, Amogh Dhamdhere, Bradley Huffaker, Jonathan M. Smith, and kc claffy. 2018. Pushing the Boundaries with bdrmapIT: Mapping Router Ownership at Internet Scale. In *IMC*. 56–69.

[25] Alexander Marder, Matthew Luckie, Bradley Huffaker, and kc claffy. 2020. vrfinder: Finding Outbound Addresses in Traceroute. In *SIGMETRICS*.

[26] Alexander Marder and Jonathan M. Smith. 2016. MAP-IT: Multipass Accurate Passive Inferences from Traceroute. In *IMC*. 397–411.

[27] Ricky K.P. Mok, Vaibhav Bajpai, Amogh Dhamdhere, and k claffy. 2018. Revealing the Load-Balancing Behavior of YouTube Traffic on Interdomain Links. In *PAM*. 228–240.

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

[29] Yibo Pi, Sugih Jamin, Peter Danzig, and Feng Qian. 2020. Latency Imbalance Among Internet Load-Balanced Paths: A Cloud-Centric View. In *SIGMETRICS*.

[30] Roland Van Rijswijk-Deij, Mattijs Jonker, Anna Sperotto, and Aiko Pras. 2016. A High-Performance, Scalable Infrastructure for Large-Scale Active DNS Measurements. *IEEE Journal on Selected Areas in Communication* 34, 6 (June 2016), 1877–1888.

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

[32] Neil Spring, Mira Dontcheva, Maya Rodrig, and David Wetherall. 2004. *How to Resolve IP Aliases*. UW-CSE-TR 04-05-04.

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

[34] Srikanth Sundaresan, Danny Lee, Xiaohong Deng, Yun Feng, and Amogh Dhamdhere. 2017. Challenges in Inferring Internet Congestion Using Throughput Measurements. In *IMC*. 43–56.

[35] Bahador Yeganeh, Ramakrishnan Durairajan, Reza Rejaie, and Walter Willinger. 2019. How Cloud Traffic Goes Hiding: A Study of Amazon's Peering Fabric. In *IMC*. 202–216.

[36] Bahador Yeganeh, Ramakrishnan Durairajan, Reza Rejaie, and Walter Willinger. 2020. A First Comparative Characterization of Multi-cloud Connectivity in Today's Internet. In *PAM*. 193–210.

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