What's in a Name? Decoding Router Interface Names

Joseph Chabarek
University of Wisconsin Madison
jpchaba@cs.wisc.edu

Paul Barford
University of Wisconsin Madison
pb@cs.wisc.edu

ABSTRACT

DNS names assigned to interfaces of network devices along an end-to-end path are an important source of information for both operations and research. Our study focuses on the interface DNS names that encode detailed information about the device *e.g.,* interface type, bandwidth, manufacturer. In this paper we describe a methodology for discovering and characterizing the structure of diverse interface DNS names. We extract, organize and assess the details of the encoding used in different networks. The results of our analysis show that many different encodings are used, and that meaningful encodings are common in the core of the Internet. To enable interface DNS name decoding to be used in practice, we incorporate our information extraction library into a new version of traceroute that we call *PathAudit*.

Categories and Subject Descriptors: C.2.1 [Network Architecture and Design]: C.4 [Performance of Systems]: Measurement Techniques
General Terms: Design, Experimentation, Measurement
Keywords: Active probing, Network measurement

1. INTRODUCTION

Network operators and researchers commonly use measurements from active probe-based tools as the basis for understanding key characteristics of Internet infrastructure. This approach is attractive because it allows tests to be performed in a targeted fashion and across infrastructure that may not be owned by the tester. Probes are used to measure dynamic properties of paths (*e.g.,* available bandwidth [7] or SLA compliance [14]), or details of application performance (*e.g.,* [3]) or service availability (*e.g.,* [8]). Probes are also used to identify structural and connectivity properties of the network *e.g.,* by interpreting the IP addresses returned by tools such as `traceroute` (*e.g.,* [16]). The characterizations that result from these measurements serve as the starting point for network planning, day-to-day network management, and for the design and implementation of new protocols and systems.

There are a number of challenges in probe-based measurements and in using them to infer Internet properties. First, systems used for probing must be carefully calibrated in order to return accurate measurements [13, 15]. Next, probing is inherently a sampling process and very little information may be returned by individual probes (typically a delay value or an IP address). This challenge can be addressed by using multiple measurements (*e.g.,* [7, 16]) or non-obvious network mechanisms [11] to infer network properties. Despite the large body of work on active probe-based network measurement and characterization, there would appear to be many Internet properties that are beyond the reach of this measurement methodology.

The objective of our work is to enhance the utility of active probe-based measurements to enable the properties of individual devices in the Internet to be identified. We seek the ability to identify properties such as device manufacturer, device type, line card type and link type among others. The ability to identify these properties is of intrinsic interest from an Internet characterization perspective, but also has important implications for inferring more detailed properties of Point of Presence (PoP) configurations as well as the possibility of inferring related characteristics such as power consumption. The challenge is that most Internet service providers consider device configuration information proprietary and actively block probes from tools such as `nmap` [4] that might reveal details of a target device.

The starting point for our study is the well-known practice of embedding location identifiers (*e.g.,* full names or airport codes) in the domain names associated with IP addresses of interfaces on network-based devices. These location identifiers have been used for many years to enhance network topology measurements [16] and IP geolocation estimates [19]. Our observation is that additional information related to device characteristics is sometimes embedded in domain names of interfaces. While these interface labeling conventions are embedded in the device's operating system and therefore only available to the network operator via the command line, they are often reflected in the domain name assigned to the interface as a matter of practice in order to assist in real time network configuration tuning and troubleshooting.

In this paper we describe a methodology for decoding domain names associated with IP addresses of network devices. Our approach seeks to identify the naming conventions used by individual service providers and to interpret the details of the name features. This generalizes and complements prior work that was focused solely on extracting location hints from domain names (*e.g.,* [16]). Our approach is based on analyzing a set of domain names from reverse DNS lookups on IP addresses collected from `traceroute`. The challenge in this work is in making sense out of the vast range of naming conventions that could be used. We use clustering to identify tag structures that have similar characteristics. We then inspect exemplars of the clusters to interpret the features and characteristics of the naming conventions.

HotPlanet '13 Hong Kong, China

We apply our domain name decoding methodology to a large set of `traceroute` measurements and associated domain names collected by the Archipelago project (Ark) [6]. The results of our analysis highlight the prevalence of structured naming conventions and the diversity of features used in naming from both a service provider and path perspective. As a partial validation of our method, we present results of a survey of network operators on the naming conventions that they use in their infrastructures. We find that large service providers routinely use identifiable naming conventions. To put our methodology into practice, we developed an end-to-end probing tool that we call *PathAudit*. PathAudit is an extension to `traceroute` that uses our custom information extraction library to report the identifiable characteristics in each interface name.

2. METHODOLOGICAL OVERVIEW

At the highest level, the domain names assigned to IP interfaces on network-based communications equipment are defined and constrained by DNS specifications [1]. To quickly review, domain names are read from right to left and consist of a series of alphanumeric strings (labels) separated by dots (“.”). The right-most label (*e.g.,* `com`) is the top-level domain (TLD) and specifies the starting point in the global, hierarchical name space. As you read from right to left, labels become more specific. For TLDs such as `.com` or `.net` that are commonly associated with domain names for network-based communications equipment, a service provider name is typically the second label to the left of the TLD, *e.g.,* `att.com`. Labels to the left of the service provider name depend on the conventions of individual organizations.

We posit that Internet service providers who assign domain names to their device interfaces use an identifiable naming convention (although some service providers may not assign names at all, as we show in Section 4). The convention may be as simple as an opaque string such as `1.foo.com`, `2.foo.com`, etc. Or, the naming convention may embed meaningful information about the interface or the device that is automatically generated by a management script and can help network operators in their day-to-day configuration and maintenance activities.

Consider `ae-5-5.ebr2.Washington1.Level3.net` an example gathered from the Level3 traceroute looking glass server. The rightmost portion `Level3.net` identifies the naming organization and the left portion identifies a network element according to that organization’s naming convention. The left part of our example clearly has structure and includes potentially valuable pieces of information about the interface and device. Note, that we include country code TLD’s in the naming organization where appropriate. Multiple measurements show that Level3 uses a structured convention for internal IP interfaces. The information specific to the device is spread across the leftmost three labels. The location (`Washington1`) is clearly evident in the third label from the right. The fourth label from the right (`ebr2`) can be interpreted as identifying the device as a core router. The leftmost label (`ae-5-5`) is specific to the interface and is interpreted as belonging to an aggregated Ethernet bundle.

While we will show that domain names can provide key insights into network infrastructure, there are a number of limitations to our method. Our approach is based on gathering interface IP addresses from TTL-limited probing tools like `traceroute`. It is possible that TTL-limited probes are blocked by networks thus limiting the scope of data gathering. While this in and of itself is not a limitation of our methodology, it does limit the scope of applicability. It is also possible that the IP addresses that are returned by TTL-limited probes may not reflect the specific ingress interface along an end-to-end path. Thus, care must be taken in drawing conclusions about path characteristics based on domain names. Furthermore, `traceroute` measurements are known to exhibit anomalous characteristics such as loops, thus care must be taken to use tools such as Paris traceroute that address these issues [5].

Our approach is based on reverse DNS lookup to recover domain names from IP addresses. Prior work has identified the operational problems associated with managing and maintaining domain names for IP interfaces such as the fact that devices and line cards may be moved or replaced without updating names. This can lead to erroneous interpretations of path characteristics, which may be able to be overcome using certain heuristics [20].

3. EXAMINING THE NETWORK INTERFACE NAMESPACE

The first part of our work is focused on extracting information from domain names that are assigned to network device interfaces. Across naming domains, (and potentially within a domain) there is a huge variation in the structure and content of an interface name. In an attempt to tame the diversity of names we use a tagging process to dynamically discover device details which can inform an inference over interface details and a domain’s naming schema. While substrings in network interface names have been used previously for tasks such as estimating the geographic location of network routers [2] and finding boundaries between networks [16], to the best of our knowledge, there have been no prior efforts to quantify or fully interpret the amount of information available in network interface names. To investigate the interface namespace, we developed a set of methods and tools that analyze the naming structure and naming content.

3.1 Information Extraction Methodology

To facilitate information extraction from network device interface names, we use a variety of information sources. These include specification details of network devices and configuration parameters [10, 18], operator observations [17], publicly available naming conventions, our operator survey, and private correspondences. We have converted these specifications and observations into regular expressions and domain dictionaries used to extract device details.

Our method begins by considering a set of end-to-end path measurements that report an IPv4 address for each hop on an end-to-end path and a record of IPv4 to DNS mappings. Such measurements are easily collected with tools such as `traceroute`. We disregard any hops that do not include a DNS name. Our focus is on understanding the details of the names of *intermediate hops* between the endpoints. Therefore, we also disregard the names of the source and destination hosts for each end-to-end probe. In this study, we focus exclusively on IPv4. However, our methods can easily be extended to interface names from IPv6-enabled devices.

After collecting a list of interface names for the taxonomy study, we order the strings by provider and parse each DNS name. Note that in the standalone PathAudit tool, each name from the running *traceroute* is individually parsed. The goal of the parsing step is to identify substrings within the name which contain extractable network information. We store the parse results in a set of tagging data structures that record the the matching substring’s beginning, end, and the type of information identified.

The tagging process is done in a single pass where each name is analysed using a series of parsing objects including regular expressions as seen in Table 1 and a dictionary mechanism to identify substrings of interest. Table 1 is a partial listing. We use regular expressions for identifying configuration parameters such as media type, interface slot, router identifier, *etc.* that have been included

Table 1: Sampling of regular expression examples with the matching tag class and the context inferred from a match. Additional regular expressions are used and will be available to the public after publication of the paper. We use \d to represent the decimal class in regular expressions, C to indicate that the link name corresponds to the Cisco IOS naming convention and J to indicate the Juniper naming convention when vendor matching class is available.

Regular expression	Matching Class	Description
fa\d+-	speed, vendor	C:Fast Ethernet
fe-\d+	speed, vendor	J:Fast Ethernet
t1-\d+	speed, vendor	J:T1
t3-\d+	speed, vendor	J:T3
gi\d+-	speed, vendor	C:Gigabit Ethernet
ge-\d+	speed, vendor	J:Gigabit Ethernet
gig\d+	speed	gigabit
te\d+	speed, vendor	C:10 Gig Ethernet
xe\d*-\d+	speed, vendor	J:10 Gig
tenge\d	speed	10 Gig Ethernet
tengigabitethernet	speed	10 Gig Ethernet
pos\d+-	vendor	C:SONET
se\d+-	vendor	C:TI
posch\d+	vendor	C:SONET
tu\d+	vendor	C:Tunnel
crs\d+	funtion,vendor	C:Core
ae-\d+	vendor	J:Ethernet Bundle
cr\d+	function	Core
Core	function	Core
ccr	function	Core
ebr	function	Core
border	function	Peering
edge	function	Peering
igr	position	Peering
br\d+	function	Peering
aggr	function	Customer
cust	function	Customer
gw\d+	position	Customer

in a name and also to identify common string patterns that indicate that there might be structure within the name. These configuration parameters often have slightly different delimiters (*e.g.,* "." or "-") or formats and require the generality of regular expressions over simple direct string matching.

We develop dictionaries to extract city locations and state names that are embedded in interface names. We compress all of the dictionaries of interest into a single *trie* implementation for fast lookup during the parsing step. If done in an unstructured manner, dictionaries can provide many false tags. For example, a field containing `Fibernet` also would contain a city tag with the value of `bern`. We mitigate this by ordering the regular expressions and dictionaries. If there is overlap between dictionary or regular expression matches, the parsing object with the lower priority is ignored. Additionally, we carefully groomed the cities dictionary and blacklisted short city names that caused significant numbers of obvious false positives.

3.2 Interface DNS Name Corpus

We use data available from the CAIDA Archipelago Project [6] to assemble a large number of end-to-end path probes and corresponding router interface names. We use daily snapshots, which report results from `traceroute`-based probes from the monitoring infrastructure. To examine the network namespace in detail, we focus on a specific 7 day measurement cycle that started on July 15, 2011 and was conducted by the hosts in Ark's team one.

There were roughly 9.5 million probes in the test cycle. These paths contained over 115 million non-unique hops encountered by the probes sent from the team one monitors and do not include the host and destination hops. Roughly 74% or 85.4 million of these hops are resolved with the IP to DNS mappings provided by the Ark bulk DNS resolution service. The resolved hops result in over 435,000 unique interface names from roughly 26,000 naming organizations, which were the starting point for our analysis.

Our objective is to assess the structure and details of interface names broadly, across a large set of networks. While the Ark project gives us a good starting point in terms of broad reach across the Internet, the ability to resolve an interface IP address to a DNS name is dependent on the policies and configurations of individual service providers networks.

Figure 1 depicts the cumulative distribution of the percentage of interface names on end-to-end paths in our data corpus that can be resolved (*i.e.,* the number of potential names that can be analyzed by our tool because the interface IP address reverse-resolves to a meaningful name). The figure shows that roughly 20% of the paths resolve all interfaces, while nearly 10% of the paths resolve fewer than 40% of the interfaces on end-to-end paths. Further investigation reveals that that interfaces with resolvable DNS names are much more likely to be encountered by probes that traverse the core of the Internet (*i.e.,* are associated with large service providers). These results demonstrate that interface naming is a common practice. Our challenge is to extract the details of the current naming conventions using our tagging process.

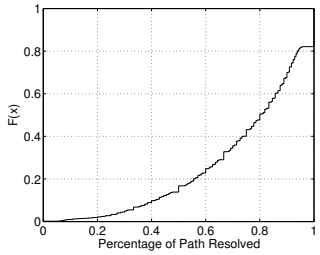
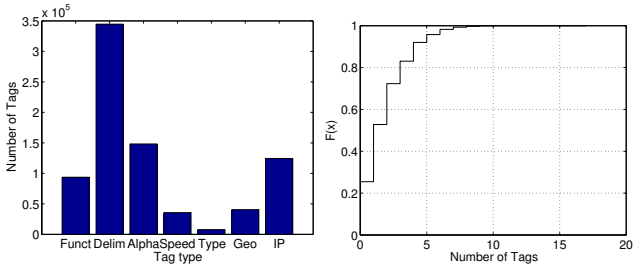


Figure 1: Cumulative distribution of the percentage of interface names on end-to-end paths in the July 15, 2011 Ark data set that resolve to a DNS name.

3.3 Tagging Results

Our tagging tool assigned roughly 800,000 tags to the 435,000 unique interface names in our data. Figure 2a shows the break down of the types of tags that were assigned. The tags are described as follows: *(i)* **Function** indicates that the interface is identified with a device that is located in the core, access, or border of the network *(ii)* **Delimiter** indicates that the name uses structured delimiters in the left-most field of the name *(iii)* **Alphanumeric** indicates that a pattern consistent with [A-Za-z][A-Za-z]+[0-9] patterns was identified, which is a common format for abbreviations *(iv)* **Speed** tag indicates a substring that hints at the interface speed such as gigabit or ten gigabit was identified *(v)* **IP** tag indicates a substring contains an IP address delineated by dashes *(vi)* **Type** indicates identification of a naming convention associated with a vendor such as Juniper or Cisco. For example, one manufacturer uses `gi-0-0-1` while another uses `ge0-0-1`. This is not enough to guarantee that a device is from a particular manufacturer, just a hint.

Figure 2b shows a cumulative distribution over the number of tags per DNS name. For named interfaces, roughly 47% have at least 2 tags. In Figures 3a, 3b , and 3c we show the top 5 tag values for the *geographic*, *speed*, *function* tags respectively. The *speed*



(a) Tags per Type (b) Tags per Name

Figure 2: Number of tags per type and CDF of tags for each unique resolved name for July 15, 2011 Ark data set.

Table 2: Field compressibility ratio for names from the July 15, 2011 Ark data set. Lower values indicate more commonality between names.

Name fields	Compression ratio
All fields	0.21
Left most	0.26
Second to left	0.24
Second to right	0.16
Right most	0.11

and *function* tags have significant concentrations in the top 5 values while the *geographic* and *alphanumeric* tags are not dominated by any particular values. Given the geographical distribution and facilities constraints of network service providers, it is not surprising that there is no dominant set of geographic tag values. The *speed* tags clearly indicate the prevalence of gigabit and ten gigabit Ethernet. They also indicated other router operating system labels for interfaces such as "`ae`" for bundled Ethernet links and "`pos`" for Packet Over SONET links, which do not indicate a specific link speed, but do provide hints to physical connectivity. In Figure 3c, the tags align with variations of the common roles of network interfaces an the edge/border/gateway of a network, in the core/carrier function, and peering interfaces. Other tags such as the *alphanumeric* tags and *delimiter* tags are used as catch-alls that attempt to find hyphen delineated subfields within the larger dotted elements.

The results above show that service providers use a variety of naming practices for assigning names to network device interfaces. To examine the commonality of the vocabulary used across naming domains, we compute the compression ratio (original file size to compressed file size) for the entire corpus and also for dotted fields of interest. A larger compression ratio indicates that the file has a higher variability and larger resulting compressed file generated by the common linux "gzip" utility. The results show that within a name, as we move from rightmost dotted field to leftmost, on average the variability of the content within the field increases. This represents the variation in naming conventions that providers use in labeling their interfaces.

4. VALIDATION

Operating systems for network devices have explicit naming conventions for the line cards in multi-card chassis systems and for ports on individual cards or fixed chassis systems. For example, in Juniper's JUNOS [10, 18], the internal interface name contains a media abbreviation, as well as the location of the port in the device. Similar naming conventions are used by Cisco and other manufacturers. For example, interfaces with the internal names of gigabitethernet1/3 and GigabitEthernet1/3 are shortened in Junos and IOS to be Ge1/3 and Gi1/3 respectively and in the leftmost field in the dns name to be *ge-1-3* or *gi-1-3*. Similarly FastEthernet2/0/5 is transposed to *fe-2-0-5*, and TenGigabitEthernet3/4 is *te-3-4*. While the exact transposition varies between providers, the intended relationship between DNS interface name and internal operating system name is clear. With automated tools that scrape interface names from router configurations, an operator can create a DNS PTR to the interface that is unique, memorable, and can be identified via `traceroute` without looking up a name-to-device mapping in a management database. This facilitates the processes of configuration management and network troubleshooting.

Table 3: Summary responses from a network operator survey on interface naming conventions

Operator practice	Number of operators
Responded	22
Automatic name generation	5
Manual name generation	15
Scripted reverse DNS	2
Manual reverse DNS	2
Geographic encoding	20
Interface function clue	16
VLAN ID	16
Media type	12
Use OS interface label	14

Additionally, we conducted an informal survey of network operators to ascertain their approach to assigning names to network interfaces. The questions included in our survey were as follows: *i)* Describe the naming convention you use for your router interfaces in detail, giving example fields and values *ii)* What networks have you been involved in naming using the aforementioned convention? *iii)* Is reverse DNS naming done in a formal or ad hoc manner?

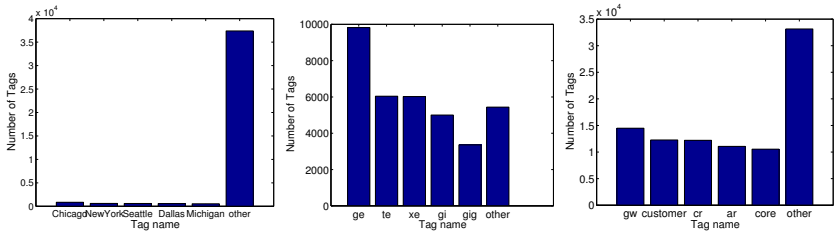
The survey was distributed to the North American Network Operators Group (NANOG). There were 22 operators who responded to our survey. Not all responders filled in every question. We summarize the results in Table 3. The table shows that operators use both scripts and data entry to name interfaces. The majority of respondents chose to add useful data such as location, function, or media type to the interface name, with 14 out of 22 responding that they used some form of the router operating system interface label in the name. This is an encouraging sign that we can find structure and meaningful device information from interface names.

The responses indicate that meaningful names are assigned to network device interfaces and that a diversity of naming methods and conventions are used. In terms of details, there are some operators that used structured names that included the city names, a router designation, an interface designation, and VLAN identifiers, geographic code, or device function. Others viewed the inclusion of device specific details with suspicion citing security concerns and resorted to names with little information content.

5. FINDING COMMON NAMING CONVENTIONS

In order to develop a deeper understanding of the naming conventions that are used for interfaces, we apply a suite of unsupervised machine learning tools along with expert knowledge. Our goal is to answer two questions: *(i)* What is the naming schema used by a particular provider? and *(ii)* What common naming conventions between providers can be identified?

We use hierarchical clustering to answer these questions. We use the interface names from selected naming organizations that



(a) Top geographic tags (b) Top speed tags (c) Top function tags

Figure 3: Top 5 occurring tags for the the geographic, speed, and function tag types from the July 15, 2011 Ark data set.

are well known service providers and build clusters out of names with similar tag structure. We choose clustering over simple matching since it is more flexible and our classification problem relies partly on tags that can be potentially misleading (in the case of a false positive), and we must consider data sets with potentially non-standard abbreviations. We would like to tune our clustering algorithms such that we are forgiving of missing tags that might not be in our parsing tool, but want to avoid clustering different naming schemes together.

We create a feature vector for each name for use in the clustering algorithm. We use binary features that indicate the presence of a tag or delimiter in each dotted subfield. Other features include: *(i)* number of dotted fields *(ii)* short string terminated by a '-' *(iii)* number of '-' delineated fields in label 1 *(iv)* geo tag in label 1 *(v)* speed tag in label 1 *(vi)* function tag in label 1 *(vii)* VLAN keyword appears.

To perform the task of intra-provider clustering we use hierarchical agglomerative clustering, a greedy-merge algorithm. With this algorithm one can either explicitly select the maximum number of resulting clusters or set a cutoff known as an inconsistency coefficient. To tune the hierarchical clustering process we can use a dendrogram visualization along with expert knowledge over a number of providers to find the number of clusters that provide representative groupings.

We choose 8 large providers from the July-2011 dataset and cluster their interface names. We found that each provider used multiple naming schemas. There were clear naming differences in all 8 of the sample networks for internal facing interfaces and customer facing interfaces. Customer facing interfaces commonly had a clear organization identifier, some naming organizations such as `alter.net` used the word "customer" in the interface name. Others, such as `easynet.net` have an organization name and gateway tag separated by a hyphen. Internal facing interfaces make extensive use of the speed, vendor, and function tags, though the tags can vary in position. The speed and function tags increase confidence that these are in fact router interfaces. Six of the eight providers have a naming schema that incorporates VLANs.

For brevity we give a breakdown of the structure of one provider, Level3. Based on the inter-cluster distance observed in the dendrogram we stop the merge algorithm when there are 6 clusters to avoid merging clusters that are significantly different. To summarize the 6 clusters, two of the clusters contain domain names that note customer names. The difference between the two clusters is hyphenation. Another cluster represents dialup interfaces and was differentiated by the dialup keyword and the presence of an dashed-delimited IP address. There is a cluster for interfaces with VLANs, and a fifth cluster represents internal interfaces with `speed` tags followed by a delineated sequence. The final cluster, has one element and appears to be an anomaly in that the function tag is in a different position than all the other tags (*i.e,* ...`te-3-1-dallas1`... as compared to ...`te-3-2.car2.dallas1`...) leading to a unique number of dotted fields for this name compared to its peers.

6. NAME ANALYSIS IN PATHAUDIT

An important objective of our work is to make our interface name analysis techniques available to the community. We believe that this capability will be useful in both measurement-based research of Internet structure and in day-to-day operations, where `traceroute` continues to be widely used for troubleshooting.

We implemented a library that we call *PathAudit*, which performs name extraction and analysis on domain names. The current version of PathAudit is implemented in roughly 1500 lines of Python. The tool is comprised of a library that implements the parsing functionality, a database containing the dictionaries for the parser, a front end utility that calls `traceroute`, analyzes the result and displays interface device information in addition to the standard address and name.

PathAudit provides analysis of interface names on an end-to-end path between a server running PathAudit and any remote client. Similar to looking glass servers that are commonly available in service provider networks, the PathAudit server initiates a `traceroute` measurement to a target host. The tool operates on the domain names that are resolved from IP addresses on hops between the source and destination hosts. A quantitative link report is produced, which includes a breakdown of each interface name following the tag types described in Section 3.1.

We describe the tool output when a probe is sent from a test end-host to the remote host `www.weather.com` (a snapshot of the tool is omitted for brevity). There are 15 hops between the workstation running PathAudit at our site and the target host. In total, 14 of the IP addresses associated with hops were resolved to domain names by `traceroute`. Examination of the details of the naming analysis reveals the following: *(i)* **Geo** tags show that the path goes from Madison to Kansas City to Dallas. *(ii)* **Speed** tags show the link speeds for five of the hops (a mixture of gigabit Ethernet and ten gigabit Ethernet) and that an additional three hops are bundled Ethernet links. *(iii)* **Function** tags show there are three occurrences of *border* and *edge* interfaces, a *peering* link is encountered at least once, and at least 3 interfaces as part of the *core* of a provider.

7. RELATED WORK

The work that is most similar to ours are studies that use information embedded in domain names to infer certain properties of the Internet. The best examples of these are studies that use location information such as city name abbreviations or airport codes to assist in identifying the geographic location of networking equipment. Paxson was one of the first to use this kind of location in-

formation in his landmark routing dynamics studies in the mid-1990's [12]. Similarly, the Rocketfuel project developed *Undns*, a location-to-node mapping tool that aids in identifying the geographic positions of routers. Our work is also informed by Zhang *et al.*, which highlights the potential pitfalls of location information in domain names [20]. Our methodology generalizes the notion of deriving meaning from domain names using all labels. Our framework also enables efficient, on-going discovery and interpretation of naming conventions using large data archives such as Ark [6].

Beyond location hints, there is little mention of standards for naming conventions in the research or network operations literature. Short articles (*e.g., Naming Conventions* by Morris [9]) and presentations (*e.g., How to Accurately Interpret Traceroute Results* by Steenbergen [17]) can be found that suggest certain methods for naming and interpretation of names, but we are unaware of any published standards. Device equipment such as Cisco Systems and Juniper publish the port naming conventions embedded in their operating systems [10, 18]. We used these to bootstrap our naming interpretation efforts.

8. CONCLUSIONS AND FUTURE WORK

The objective of our work is to gain deeper insights into the structure and behavior of the Internet. In this paper, we describe an general analytic framework for decoding domain names associated with IP interfaces on network elements. Active probes are used to gather device interface IP addresses, which are translated into domain names via reverse lookup. We parse and tag the substrings in the names and then cluster and interpret the strings to identify naming conventions.

We analyze an archive of path probes from the Ark project [6]. Our results highlight the details of the naming conventions. Typical features include device role, device type, link type, and interface slot number. These results are validated through self-consistency checks with device manufacturers and through an on-line survey of service providers. We also assess the prevalence of device-specific naming conventions among service providers and on end-to-end paths. Our analysis shows that identifiable naming conventions are prevalent in large service providers whose equipment tends to appear on many paths and these providers have largely adopted standards for naming that reveal important device details. To put our methodology into practice, we develop an active measurement tool called PathAudit, which is built on top of `traceroute`.

Our on-going work is focused in three areas. First, we continue to expand and enhance the capabilities of PathAudit so that it can identify the broadest set of device details, to this end we are working to automate the process of adding new tag names as they emerge in interface labels. Second, we are enhancing our analysis methodology to consider ensembles of measurements toward the goal of understanding rack and PoP configurations. Finally, we continue to analyze and evaluate Ark data toward the goal of more broadly understanding the Internet characteristics.

Acknowledgements

This work was supported in part by NSF grants CNS-0831427, CNS-0905186, ARL/ARO grant W911NF1110227 and the DHS PREDICT Project. Any opinions, findings, conclusions or other recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, ARO or DHS.

9. REFERENCES

[1] Domain Names Implementation and Specification. `http://www.ietf.org/rfc/rfc1035.txt`, 1987.

[2] Undns. `www.scriptroute.org/source/`, 2002.

[3] Keynote Systems Web Performance Testing. `http://www.keynote.com`, 2012.

[4] Nmap Free Security Scanner. `http://nmap.org`, 2012.

[5] B. Augustin et. al. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proceedings of IMC '06*, October 2006.

[6] Y. Hyun, B. Huffaker, D. Andersen, E. Aben, M. Luckie, kc claffy, and C. Shannon. The Archipelago Measurement Infrastructure. `http://www.caida.org/projects/ark`.

[7] M. Jain and C. Dovrolis. End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, August 2002.

[8] W. Jiang and H. Schulzrinne. Assessment of VoIP Service Availability in the Current Internet. In *Proceedings of Passive and Active Measurement Conference '03*, San Diego, CA, March 2003.

[9] M. Morris. Naming Conventions. *NetworkWorld*, January 2008.

[10] Juniper Networks. Interface Naming Conventions Used in the JUNOS Software Operational Commands. `http://www.juniper.net`, 2012.

[11] JJ. Pansiot, P. Mindol, B. Donnet, and O. Bonaventure. Intra-Domain Topology from mrinfo Probing. In *Proceedings of Passive and Active Measurement Conference '10*, Zurich, Switzerland, March 2010.

[12] V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5), October 1997.

[13] V. Paxson. Strategies for Sound Internet Measurement. In *Proceedings of the ACM Internet Measurement Conference '04*, Taormina, Italy, March 2004.

[14] J. Sommers, P. Barford, N. Duffield, and A. Ron. Multi-objective Monitoring for SLA Compliance. *IEEE/ACM Transactions on Networking*, 18(2), April 2010.

[15] J. Sommers, P. Barford, and W. Willinger. Laboratory-based Calibration of Available Bandwidth Estimation Tools. *Elsevier Microprocessors and Microsystems Journal*, 31(4), 2007.

[16] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP Topologies with Rocketfuel. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, August 2002.

[17] R. Steenbergen. How to Accurately Interpret Traceroute Results. North American Network Operators Group Meeting 45, 2009.

[18] Cisco Systems. Interface and Line Numbers in Cisco 1800, 2800 and 3800 Series Routers. `http://www.cisco.com/en/US/products/hw/routers/ps282/products\_tech\_note09186a008035b051.shtml`, 2012.

[19] B. Wong, I. Stoyanov, and E. Sirer. Octant: A Comprehensive Framework for the Geolocation of Internet Hosts. In *Proceedings of USENIX Symposium on Network Systems Design and Impelmentation*, Cambridge, MA, April 2007.

[20] M. Zhang, Y. Ruan, and J. Rexford. How DNS Misnaming Distorts Internet Topology Mapping. In *Proceedings of USENIX Annual Technical Conference '06*, Boston, MA, May 2006.