Context-aware Clustering of DNS Query Traffic

David Plonka
University of Wisconsin-Madison
plonka@cs.wisc.edu

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

ABSTRACT

The Domain Name System (DNS) is a one of the most widely used services in the Internet. In this paper, we consider the question of how DNS traffic monitoring can provide an important and useful perspective on network traffic in an enterprise. We approach this problem by considering three classes of DNS traffic: canonical (*i.e.,* RFC-intended behaviors), overloaded (*e.g.,* black-list services), and unwanted (*i.e.,* queries that will never succeed). We describe a context-aware clustering methodology that is applied to DNS query-responses to generate the desired aggregates. Our method enables the analysis to be scaled to expose the desired level of detail of each traffic type, and to expose their time varying characteristics. We implement our method in a tool we call *TreeTop*, which can be used to analyze and visualize DNS traffic in real-time. We demonstrate the capabilities of our methodology and the utility of TreeTop using a set of DNS traces that we collected from our campus network over a period of three months. Our evaluation highlights both the coarse and fine level of detail that can be revealed by our method. Finally, we show preliminary results on how DNS analysis can be coupled with general network traffic monitoring to provide a useful perspective for network management and operations.

Categories and Subject Descriptors: C.2.3 [Network Operations]: Network management, Network monitoring, C.4 [Performance of Systems]: Measurement Techniques

General Terms: Design, Experimentation, Measurement, Performance

1. INTRODUCTION

Methods for classifying and identifying key characteristics of network traffic have important implications in network management, traffic engineering and network security. For example, the popularity and large sizes of the files distributed through peer-to-peer (P2P) applications can consume a significant percentage of the bandwidth in a network. The ability to accurately identify P2P traffic can enable it to be throttled at the network border to the benefit of other more critical traffic types. Similarly, the ability to identify malicious traffic accurately and in a timely fashion in the best case can enable an attack to be blocked before it is completed or at least can enable the effects to be mitigated quickly.

The key challenge in accurately identifying different traffic types and their characteristics is that there is no inherent mechanism for this task. In years past, port numbers could be used to classify a large percentage of network traffic, primarily due to the limited diversity of applications. However, there is a wide variety of applications in use today, and many of these use ephemeral ports or standard protocols such as HTTP for communication, which defeat simple classification via port numbers. In the case of malicious traffic, there is strong incentive to actively obfuscate payloads (*e.g.,* via packing and morphing methods), which makes the identification problem even more challenging. Finally, encrypted traffic transmitted via standard protocols represents perhaps the most significant classification challenge since it would seem that almost no details could be discerned.

Prior work on non-port based approaches to identifying network traffic include payload-based analysis, behavioral analysis and clustering analysis. Payload-based approaches (*e.g.,* [28, 12]) are standard *e.g.,* in network intrusion detection systems (NIDS) and in some commercially available traffic shaping systems. This approach tries to match packet payloads to a library of signatures composed of unique byte sequences associated with particular attacks or applications. A disadvantage of the payload-based approach is that byte sequences are often not unique to a particular traffic type, which leads to the well-known false alarm problem in NIDS. Classification methods based on behavioral characteristics such as [19, 29, 20] focus on building statistical models of transport layer metrics such as connection duration and packet size to distinguish applications. Cluster-based approaches such as [13, 22] take the next logical step by using standard machine learning methods to divide traffic into groups based on similarity of transport layer characteristics. We believe that these methods have merit but are ultimately limited by the diversity of information available to them from the protocols that are being used. While traffic classification using methods such as the aforementioned can be useful, they often omit key details that are required to diagnose and remedy problems and are likely to never be able to fully distinguish all traffic types accurately. We argue that a broader perspective is necessary.

In this paper, we investigate the question of how Domain Name System queries could be used to provide important and unique insights on network traffic. Our motivation for this work is the observation that DNS is used by almost all applications in the Internet, and the conjecture that the plain-text DNS query/response traffic is a rich source of information on network traffic that might otherwise be difficult to understand. For example, while prior classification methods might accurately identify application traffic as HTTP, information from DNS queries that precedes this traffic could be used to further label the traffic with prominent domain names. (Throughout the remainder of this paper, we refer to standard or expected DNS traffic as "canonical".) DNS is also now routinely used for black-listing services (throughout the remainder of this paper, we refer to this type of DNS traffic as "overloaded"), which are critical for spam checking, but increasingly used for other purposes (see Section 3 for details). Understanding the nature of this traffic could be useful in network operations. Finally, there are many queries that never succeed, but still require DNS resources. So, any improvement in understanding this category of DNS traffic (throughout the remainder of this paper, we refer to this type of traffic as "unwanted") will be important to network operations and security administrators.

The starting point for our work was a set of traces of DNS query/response traffic continuously gathered from our campus network from January through April, 2008. This data set comprised over 11 billion total query responses for tens of thousands of clients. With a data set this large and diverse, a principled analysis method is required in order to extract, visualize and evaluate the desired information.

Our approach to analyzing the DNS traces is data-driven and context-aware. In particular, we apply a clustering methodology that is guided by DNS syntax and semantics to decompose the query/response traces into the three major categories described above. We also employ IP prefix and domain name search trees to divide clusters into more detailed subclusters and aggregates. Rather than relying on single fields, we distinguish additional unwanted and overloaded traffic types by identifying combinations of query names, response codes, and answer values. Additionally, we employ a "reflexive clustering" method that uses these multiple dimensions for creating groups where the interpretation of one group is based on the context of the other.

We implemented our context-aware clustering method in a tool we call *TreeTop*. This tool enables both off-line and real-time analysis and visualization of DNS query/response traffic. Specifically, TreeTop analyzes query/response traffic with a variety of filters and summarizes in tabular or graphical reports. TreeTop is currently in operational use in our campus network and is also available to the community [25].

When applied to our DNS query/response traces, TreeTop highlighted a number of interesting characteristics that demonstrate the utility of our approach. First, we found a diurnal cycle consistent with standard packet traffic. The profile for this traffic is relatively smooth and clearly highlights a wide variety of popular applications such as Facebook, Google, etc. Next, we automatically identified approximately 200 black-lists and found black-list traffic to be of significant volume continually while also marked by high magnitude spikes. Finally, we defined and measured a new high-volume category of unwanted, avoidable queries due to incorrect use of resolver search lists. While the details of these results are derived from our local dataset, our approach and TreeTop can be used to investigate similar activity in other networks.

The remainder of this paper is organized as follows. In Section 2, we discuss prior studies that are related to our own. In Section 3, we provide an overview of DNS including details that are pertinent to this paper. In Section 4, we describe the measurement infrastructure used to gather our DNS query traces, and details on the traces themselves. In Section 5, we describe our context-aware clustering method, and in Section 6, we describe the implementation of the method in our TreeTop tool. The results of the analysis of our dataset are provided in Section 7. We outline future work, summarize, and conclude in Section 8.

2. RELATED WORK

Methods for analyzing the characteristics of network traffic behavior have been described in a large number of prior studies. Of particular relevance to our work are prior studies that describe techniques for classifying network traffic including [19, 20, 29, 13, 22]. These methods have been shown to be highly accurate, and we consider the information that they produce to be complementary to what is produced by our DNS query analysis. Our approach to clustering is informed by the work of Cho, *et al.* in [7] and Estan, *et al.* in [14]. Both employ hierarchical aggregation based on IP packet header information and the latter describes a dynamic method for creating minimal, multidimensional clusters of interest. Our work diverges from those techniques by utilizing the DNS traffic payload to create new clusters and by introducing hierarchical aggregation by domain names. Clustering methods have been applied to network traffic in several other studies. For example, Estan and Varghese describe a method for efficient identification of heavy hitter flows that is based on a fixed cluster definition [15]. Zhang, *et al.* describe a method for detecting anomalous BGP route advertisements based on clustering update behavior [39]. Finally, Yegneswaran *et al.* use cluster analysis as the key component of an algorithm for intrusion signature generation in [37]. Our work differs from these in that our clustering techniques are customized to the unique semantics of the DNS.

There is a growing literature on the empirical characteristics of DNS behavior and performance (*e.g.,* [3, 18, 21]). These studies have focused on volume and diversity of query types from both the client and server perspectives, and shed light on the impact of specific mechanisms such as client-side caching. More recent studies have focused on how the DNS can provide insights on unwanted activity. For example, Whyte, *et al.* describe a method for identifying scanning behavior associated with worm infections based on maintaining whitelists of known DNS records [36]. Other works propose new tools for passive monitoring of DNS traffic. Wessels, *et al.* [34, 35] introduce a tool to identify high volume types of DNS traffic. While we build upon this tool (*dnstop*), our analysis differs in that we perform clustering based on the DNS response answer values rather than just queries and response codes and then we *apply* DNS measurements to classify IP traffic in general. In [33], Weimer introduced a tool that populates a database from passive DNS traces and effectively identified abusive behavior including botnet activity.

Likewise, Zdrnja, *et al.* [38] record DNS trace information to a database and subsequently identify DNS anomalies including fast flux domains typically associated with botnet activity [10, 26]. While that work mentions DNS traffic due to anti-spam tools, our work differs in that we isolate and measure this black-list traffic, and we monitor all DNS query responses from clients, not just authoritative answers. Several of the methods described in these papers are incorporated into our DNS monitoring framework.

Finally, Ren, *et al.* [27] propose visualization techniques for DNS data. Our motivation is similar to this study in some respects and we also utilize time series data as input, but our work differs in analysis method and in the tree-based visualizations that we produce.

3. DNS MECHANICS

We are primarily concerned with analysis of DNS packets sent in response to queries from end-hosts, *i.e.,* those at the periphery of the Internet. As in [38], we analyze just the responses (replies) because the details of the query are repeated in the corresponding DNS response packet. Here we present a partial overview of the DNS service as it is used by these hosts and provide definitions of the terms we use in this paper.

DNS query packets and response packets have a similar form, and are typically exchanged between clients and name servers using the UDP "domain" service port 53. The packet contains a header, a question section, and an answer section.

Generally, queries are performed with query names that are Internet domain names. The Internet domain space is hierarchical [1], with a well-known set of top-level domains, such as "com," "net," and "org." Institutions have sub-domains, such as "example.com" and "example.org," in which they can arbitrarily create sub-domains and entries such as "www.example.com."

DNS client hosts typically perform queries by using a resolver that is supplied with the operating system. The most common queries are for the IP addresses associated with domain names. These queries have a type IPv4 Address (A) or IPv6 Address (AAAA, known as "quad A") and contain a string-based query name such as "www.example.org," to which a DNS name server typically responds with either "No error" (NOERROR) or "Nonexistent Domain" (NXDOMAIN). In the NOERROR case, one or more IP addresses, such as 192.0.2.2, are returned in the response packet's answer section. Other common query types include those for Mail eXchanger (MX) records used to route e-mail, Pointer (PTR) records used to translate IP addresses to names, Service Location (SRV) records used for automatic discovery of services, and Text (TXT) records used for various purposes. Each query type may have its own corresponding answer type.

We refer the reader to either [32] or [23], [24] for a thorough introduction to DNS packet structure and service semantics.

[1]See Figure 5 for a graphical example of a portion of the DNS hierarchy.

4. EMPIRICAL DATASETS

In this work, we are interested in DNS traffic, *i.e.,* queries and corresponding replies, exchanged between Internet hosts and trustworthy recursive name servers. To assure the legitimacy of the servers, we monitor only the traffic involving those servers under the campus' administrative control. This avoids us having to question the validity of responses because the campus DNS servers perform recursive queries, on their clients behalf, only to zone-authoritative name servers (based on referrals from the Internet's trusted root servers). Thus, we avoid rogue DNS servers such as those investigated in [9].

For off-line analysis, we capture DNS traffic exchanged between campus client hosts and the campus' recursive anycast [2] DNS service. Our university operates a recursive name service consisting of four geographically dispersed server machines that answer queries received at one of the service's two IP addresses, which are in different campus network prefixes. As such, this recursive anycast DNS service exemplifies current best practice for a large, highly-reliable lookup service that serves tens of thousands of clients. The complication introduced by anycast is that any of the servers could handle a specific client's request, so we monitor all servers simultaneously, and combine the traces at synchronized points in time to get a complete view.

In this paper, we consider a traffic trace from January 8, 2008 through April 21, 2008. Tables 1 [2] and 2 show the query types and response codes as percentages of total DNS traffic observed during this time. The active client numbers are based on the count of clients observed performing queries in a five minute interval. Figure 1 presents the traffic as a time series. While the details have been omitted for space, note the rich set of characteristics involving multiple dimensions in the measurement data. (The weeks labeled 2, 3, and 12 are during the January inter-semester and spring recesses, thus had lower traffic volume due to fewer active clients.)

Query Type	Queries/Sec	Active Clients
A	671 (54%)	4521 (87%)
PTR	310 (25%)	1386 (26%)
AAAA	120 (10%)	906 (17%)
MX	99 (8%)	197 (3%)
TXT	25 (2%)	112 (2%)
SRV	5 (0%)	145 (2%)
any	1236 (100%)	5183 (100%)

Table 1: DNS query distribution: average rates and average numbers of active clients by query type. An "active client" is one that has performed a DNS query within a given five minute interval.

For online analysis in real-time, we also monitor traffic at individual DNS servers and on an individual workstation. That is, the traffic is observed within the end host, either the DNS server or client host, at its network interface.

[2]The percentages of active clients in Table 1 are not expected to add to 100% because any given active client can issue multiple types of queries in a measurement interval.

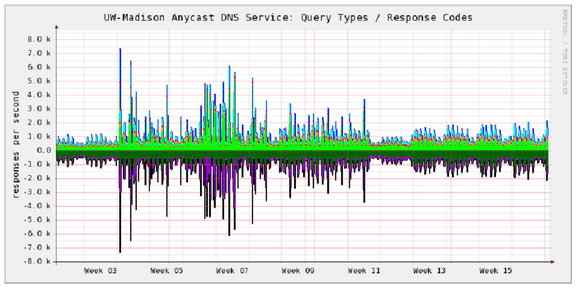

Figure 1: DNS query and response rates, January 8, 2008 through April 21, 2008. Query rates by type are plotted above the horizontal axis and the corresponding response rates by code are plotted below. See Tables 1 and 2 for the rate values.

Response Code	Responses/Sec
NOERROR	729 (59%)
NXDOMAIN	480 (39%)
SERVFAIL	27 (2%)
any	1236 (100%)

Table 2: DNS response distribution: average rates by response code.

5. ANALYSIS METHOD

Our initial observation about the measurement data, presented in time series in Figure 1, is that the DNS query responses have a rich set of characteristics not unlike those seen when measuring *all* Internet traffic (*i.e.,* not just DNS) involving a similar number of hosts. This observation motivated our analysis goals and the methods we developed to achieve them.

5.1 Goals

We have two primary goals for off-line and real-time DNS traffic analysis:

1. **Distill Useful DNS Traffic Types.**

   The number of combinations of DNS packet field values is large, similar to that of TCP and UDP IP headers in general IP traffic. This suggests applying analysis techniques successful in prior work, *i.e.,* aggregation-based clustering techniques inspired by [7] and [14], both of which use hierarchical, volume-based clustering to more succinctly store and represent an otherwise overwhelming number of measures. Thus, our foremost goal is to distill the measurement data so that we can present essential, concentrated clusters that will be useful in both research and operations.

2. **Enable Flexible Analysis.**

   Our second goal is a flexible analysis of DNS traffic such that we can answer new questions and conveniently apply the knowledge gleaned from our analysis to broader Internet traffic applications.

   For example, we wish to use the knowledge of the domain names by which clients refer to Internet hosts for the measurement and analysis of IP traffic in general. That is, we want to classify traffic by familiar domain name identifiers.

5.2 Methods

We use two methods to achieve our goals:

1. **Context-awareness.**

   Our first method is to form clusters by leveraging the knowledge of DNS syntax and semantics. Instead of attempting to apply general clustering methods (*e.g.,* simple K-means), we use knowledge of the protocol itself and knowledge gleaned from prior work to assemble DNS-specific clusters.

   Our starting point for context-aware clustering is based on our specification of three general types of DNS queries. While other high-level taxonomies of DNS traffic are certainly possible, we argue that the following three classes support the goal of making the resulting analysis useful in both research and operations.

   Unwanted Traffic.
   Many of the prior empirical studies of DNS traffic discuss high-volume anomalies observed in the data, and are driven by concern of their potential impact on local and Internet-wide DNS operations. These anomalies are within an important class of *unwanted* DNS traffic including all sorts of misdirected and malformed queries, such as those with IP addresses as query names, unknown Top Level Domains (TLDs), RFC-1918 addresses for PTR, and for names containing invalid characters.

   Overloaded Traffic.
   The DNS has come to be both extended and reused for new purposes in both foreseen and creative ways, *i.e.,* it has become *overloaded.* By this we mean that an earlier function of the DNS is overloaded with new meaning (rather than meaning that the DNS service is experiencing excessive load due to these new purposes). In light of these new uses, there is the danger of misinterpreting this "overloaded" traffic as either unwanted or typical DNS traffic, thus we wish to identify and isolate it in analyses.
   The primary examples of applications that *overloads* the DNS are "black-lists." The most common intent and use of these lists is to limit spam or network abuse by providing a mechanism for determining whether or not a given IP address or domain name is currently a member of a list that is maintained by some "listing service" (both community-based and commercial services are available). These lists exist in many varieties including Real-time Blackhole Lists (RBLs), DNS Black-Lists (DNSBLs), DNS White-Lists (DNSWLs), Uniform Resource Identifier Black-Lists (URIBLs), Spam URI Real-time Black-Lists (SURBLs), and Right-Hand-Side Black-Lists (RHSBLs, for testing the domain name portion of an email address).
   Black-lists employ an informal protocol [1] atop DNS and, in doing so, they overload the meanings of the DNS A query type and its response codes. For instance, a given IP address or fully qualified domain

name (FQDN) is tested by prepending it to the black-list's domain name and then performing a DNS lookup, and testing for "magic numbers" in the returned answer. While the meaning of these numbers is defined by the particular black-listing service, black-lists clearly overload the DNS query types, response codes, and answers, thus requiring special context-aware treatment in our clustering method to isolate this traffic from the canonical. In Section 6, we explain in detail how we cluster this traffic using a technique we call "reflexive clustering."

Canonical Traffic.
This class of traffic is the expected, well-behaved DNS traffic. Essentially, it is what is likely to be left over once the unwanted and overloaded traffic is removed, and is most often used to identify hosts and services, such as converting domain names to IP addresses or the reverse (A, AAAA, or PTR queries), routing electronic mail (MX queries), etc. Canonical traffic uses the RFC-defined query classes, types, and response codes in a well-defined fashion.
We have significant interest in the canonical traffic and the clients involved in it since our intent is to apply the information gleaned to improve identification and analysis of the subsequent IP traffic involving those clients. The DNS query/response traffic is a compelling, transparent source of additional information about Internet traffic beyond what is available in packet headers. DNS traffic is of relatively low volume (compared with all IP traffic involving a given population of clients), making it practical to process in real-time. Lastly, it is not obscured by encryption mechanisms that thwart general payload analysis.
With these categories, our method improves the analysis of DNS traffic by using clusters involving multiple fields of the response packets (such as query name, response code, and answer values) and reflexive clusters prepared from other clusters in a DNS-specific way. That is, we form clusters using the contextual knowledge of DNS traffic and its idiosyncrasies for unwanted, overloaded, and canonical traffic.

2. **Utilize Purpose-built Data Structures.**
Our method to achieve the goal of flexible clustering and analysis in *real-time* is to utilize efficient, high-performance data structures to handle IP addresses and domain names. (In contrast, a relational database as the data store is a good choice for off-line analysis as in [33] and [38].) The ability to store, lookup, and report IP address and domain names are key functions to identify and measure the unwanted, overloaded, and canonical types of traffic. Furthermore, an implementation will benefit if these data structures can be combined and nested arbitrarily. This is the online equivalent of the flexibility achieved by joins in relational databases.

We continue in the next section by describing our implementation of these methods to cluster DNS traffic.

6. DESIGN AND IMPLEMENTATION

To implement and apply our clustering methodology, we developed two high-performance data structures and an analysis tool that employs them. Both naturally have a hierarchical structure like the IP address and domain name systems whose elements they store. These data structures are described below.

6.1 Data structures

6.1.1 *The Address Tree*

Our *address tree* is a binary prefix search tree or trie similar to a Patricia trie, as used in BSD UNIX to perform efficient longest-prefix matching for IP routing lookup tables [30], but with additional features.

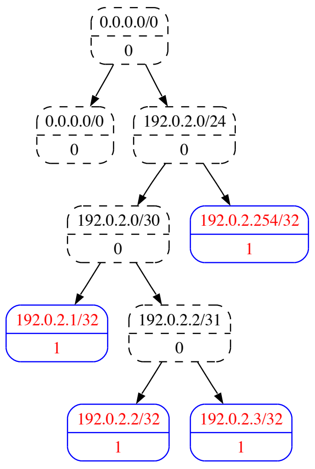

Figure 2: An address tree containing four IPv4 addresses, each with a count of 1. Internal nodes are shown with dashed lines and occupied nodes with solid lines.

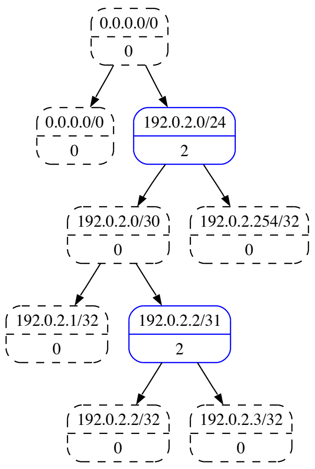

Figure 3: An address tree containing two IPv4 prefixes, each with "rolled-up" counts of 2. This is the result of aggregating the tree shown in Figure 2 with a 40% threshold.

An example address tree is shown in Figure 2. The address tree is based on the tree implementation in Aguri that is thoroughly described in [7], and has the following characteristics:

- The trie's alphabet consists of only binary digits 1 and 0. Thus, the internal node out-degree is 2.
- Level-compression is employed to reduce node count and thus increase storage efficiency. This can also benefit performance by eliminating the traversal of a long list that terminates in just one entry.
- Aggregation is performed by configurable threshold tests of a counter stored in the node. This aggregation "rolls up" entries from more-specific to less-specific. Figure 3 shows a 40% threshold aggregation of the tree shown in Figure 2; this means that nodes with values representing less than 40% of the total (in this case, 4) are aggregated to the parent node. The affected leaf nodes are available for reclamation.
- A Least-Recently-Used (LRU) node allocation scheme is employed. This allows total size of the tree to be bounded and to automatically aggregate reclaimed leaf node counts to their parents whenever the list of free nodes is exhausted.

In addition to the functionality of Aguri's tree, we added the following:

- The option to dynamically allocate nodes on demand rather than a fixed pool of nodes reclaimed by LRU (with automatic aggregation). When not memory-constrained, this allows us to retain all host IP addresses in the tree, so that detail is not lost. This enables exact set representation (for instance, to store interesting IP prefixes), accurate counting of entries inserted into the tree, and thus additional analyses. [3]
- Testing for exact match and longest-prefix match without modification of the tree. Essentially, this provides general set-membership testing in the IP address space. Since Aguri's tree was fine-tuned to its sole purpose, it didn't provide an API to test for matches in the tree. (It could add entries, increment counters, optionally aggregate, and report the tree contents.)

The address tree data structure is used as the basis for clustering. It gives us the ability to aggregate by IPv4 and IPv6 addresses [4] and to test for set membership in prefix sets.

6.1.2 *The Domain Tree*

Inspired by the effective use of prefix tries within the IP number space in both Aguri and AutoFocus [14], we employ a similar technique to the domain name space. Thus, we introduce the *domain tree*: an n-ary prefix search trie for fully-qualified-domain names.

Figure 4 is an example of a domain tree structure. Domain trees differ from address trees in the following ways:

[3]See [7] for an analysis of the accuracy of counting when aggregation is applied.

[4]For brevity, we've shown just IPv4 addresses in the figures, however we actually use address trees as a unified store of IPv4 and IPv6 addresses, by representing an IPv4 address as a 128-bit "IPv4-Mapped IPv6 Address." [17]

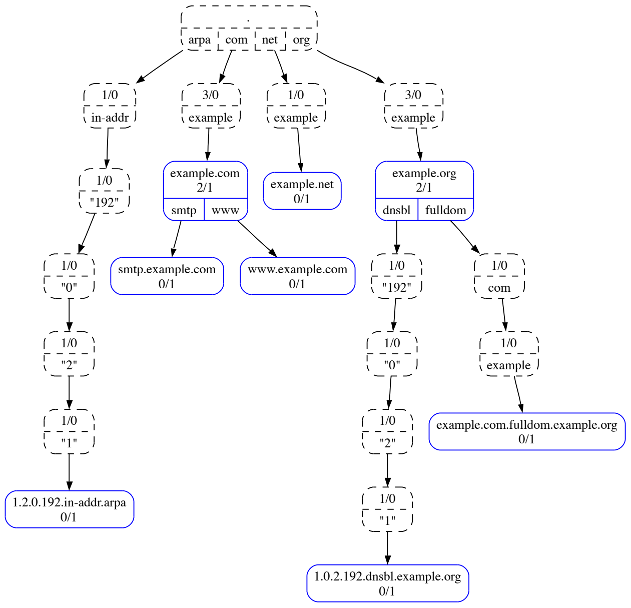

Figure 4: A domain tree counting references to 8 fully-qualified domain names (FQDNs). Various prefix and exact counts for those entries are shown, slash-separated, in the nodes as $prefix\_count/exact\_count$.

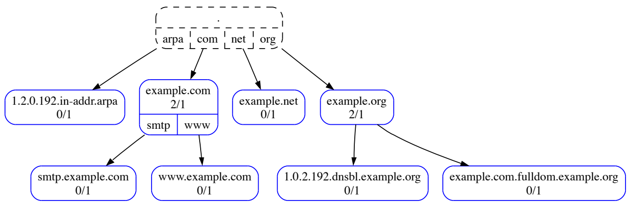

Figure 5: A domain tree containing 8 FQDNs; this is a level-compressed presentation of the tree shown in Figure 4.

- Since the presentation of a domain-name is a series of labels separated by "." characters (*e.g.,* "www.example.com") with the most-specific label first rather than last, domain trees use reversed FQDNs, *e.g.,* "com.example.www." Thus, the prefixes matching the FQDN "www.example.com" include "com" and "com.example," but not "www."
- The alphabet representable by a domain node consists of all possible case-inspecific domain name labels. RFC-1035 [24] specifies a 63 character maximum length. Thus, the maximum domain node out-degree is very large. In our implementation we store references to child nodes in a red-black tree [8], so that it is both space efficient (versus a hash) and exhibits predictable performance. [5]

[5]The red-black tree does not play a major role in determin-

- Terminal domain tree entries, *i.e.,* FQDNs, are of variable distance from the root and are not always leaves in the tree. For instance, "www.example.com" and "example.com" could both be valid domain names resolving to an IP address. In contrast, full IP address entries in an address tree are always leaves, and thus never both a terminal entry and a prefix.

- Aggregation is implemented as a reporting feature rather than a data restructuring feature. Therefore, domain tree nodes must contain multiple counters: one counting exact matches and one counting prefix matches. This is also necessary because, unlike fixed-length IP address entries, FQDNs can contain other FQDNs, thus internal nodes that are also terminal entries need exact counters.

- Level-compression is a reporting feature rather than a structural feature. This retains the advantage of compacting the presentation but without introducing the need to store label sequences, *i.e.,* varying sizes of arrays of labels, within a node.

Figure 5 shows a level-compressed report of the same domain tree as depicted in Figure 4. Like address trees, domain trees are employed in clustering. For instance, we can aggregate by domain names queried or test query names for prefix matches in sets of FQDN suffixes such as known TLDs and dynamically discovered DNS black-lists. (Recall that the FQDNs are represented in reverse, so a prefix match in the tree means that the suffix matches in the canonical FQDN presentation format.)

6.2 The TreeTop Analysis Tool

We developed a DNS and general traffic analysis tool called *TreeTop*. TreeTop is implemented as a patch to dnstop [35], and is about 8000 lines of C code including approximately 3000 lines originally from dnstop. Thus, TreeTop has all dnstop's functions combined with our additional features (including the ability to identify additional unwanted traffic). We've run TreeTop on Intel and PowerPC-based Linux and Mac OS X machines; it should be portable to other UNIX-like systems.

TreeTop has two forms of output, tabular and hierarchical text reports and graphical reports in which hierarchies are represented as directed graphs.

TreeTop sets the aggregation threshold in one of two ways. When run interactively, TreeTop sets the threshold as a function of terminal window size. It chooses a threshold with the goal of representing 100% of the observed traffic as a level-compressed tree in the user's window. When run non-interactively for off-line analysis, TreeTop's aggregation levels, and the size of the resulting reports and tree graphs, are configurable via a command-line option.

In contrast to prior tools that employ single-level hashes with domain names as keys (such as dnstop and nscd), TreeTop's functionality is based on the aforementioned data structures: address trees, domain trees, and combinations thereof. We employ them to quantify unwanted traffic in new dimensions (such as hierarchical counting of both the total number of clients and number of domains queried), to identify overloaded DNS traffic (such as DNS black-list queries), and, ultimately, to classify general IP traffic based on the domain names by which the participating hosts know that IP traffic's source and destination IP addresses.

ing the domain tree's functionality; other structures may be substituted based on performance objectives.

6.2.1 *Clustering DNS Black-list Traffic*

Here we describe how TreeTop clusters black-list traffic; other clustering is done similarly, but sometimes using filters that were already present in dnstop. In Section 5, we explained that DNS black-lists overload the meaning of particular fields of the request and response packets. To identify this type of traffic, we look for high-confidence evidence of it, then save some state information, and interpret subsequent packets using that state, where otherwise the interpretation would be ambiguous. We call this "reflexive clustering" and describe it below using black-list traffic to illustrate. The term "reflexive cluster" is analogous to "reflexive ACL": an access control list (ACL) with entries that are created dynamically based on the prior matching of a packet to a corresponding ACL.

Consider an example involving the domain entries shown in Figures 4 and 5. To query a black-list, a candidate IP address or FQDN is prepended to a black-list's domain name and then a DNS lookup for an A (IPv4 Address) record is performed. Suppose there exists a DNS black-list named "dnsbl.example.com" that black-lists IP addresses that are known sources of spam e-mail. Suppose further that "smtp.example.com" is a Mail-eXchange (MX) host that receives an email message from host 192.2.0.1. Wishing to limit the propagation of spam, this MX host queries for "1.0.2.192.dnsbl.example.org." If it results in an NXDOMAIN response, it means 192.2.0.1 is *not* a member of the black-list. If it results in a NOERROR response, an IPv4 address within the reserved 127.0.0.0/8 local network is returned, *e.g.,* 127.0.0.2. In the context of black-lists, the NOERROR response means that the given IP address is listed. Furthermore, IPv4 addresses in the answer section are overloaded; 127.0.0.2 commonly means this is a general entry in the list. Thus, black-list domain queries are distinguished, at least in part, by the fact that that they return bogon addresses. Bogon addresses are addresses that should never be routed in the Internet because they lie within either reserved address spaces (like 127.0.0.0/8) or within prefixes that have yet to be allocated by the Internet Assigned Numbers Authority (IANA) [11].

To identify and cluster black-list traffic, TreeTop first maintains a read-only *bogon address tree* because black-listing uses bogons as answers. Next, as response packets are processed, if the response code is NOERROR, TreeTop performs a prefix match in the bogon address tree for any addresses present in the packet's answer section. If a match is found, then the answer is a bogon (*i.e.,* within 127.0.0.0/8) and TreeTop adds the address to a *bogon seen address tree.* TreeTop next examines the packet's query name. If the name appears to begin with either (1) an embedded IP address (as a reversed dotted-quad, *i.e.,* "1.0.2.192") or (2) a nested FQDN ending in a known TLD, it adds the trailing domain (*i.e.,* "dnsbl.example.org") to a *list domain tree* and increments a counter of query references to that black-list domain name. At this point TreeTop has likely discovered a black-list and has a cluster counting references for true positive hits in the black-list.

To count the black-list negative response misses, TreeTop performs a prefix match in the list domain tree if a packet's response code is NXDOMAIN. If a match is found, TreeTop examines the packet's query name as above; if it begins with either an embedded IP address or nested FQDN, the count for the matching entry in the list domain tree is incremented.

In this way, the total DNS black-list traffic is accumulated in a reflexive cluster; the counts of NOERROR responses (that identified the black-lists) and NXDOMAIN responses associated with all black-lists are summed in the list domain tree. This tree is subsequently used to quantify the black-list traffic and to save the black-list domains to a file for initialization of the list domain tree on subsequent TreeTop analysis runs.

6.2.2 *General Traffic Analysis*

The clustering technique described for black-list traffic that uses an assemblage of carefully linked address trees and domain trees generalizes to other purposes. Here we describe how we use the addresses observed in IP traffic headers to cluster IP traffic by its domain.

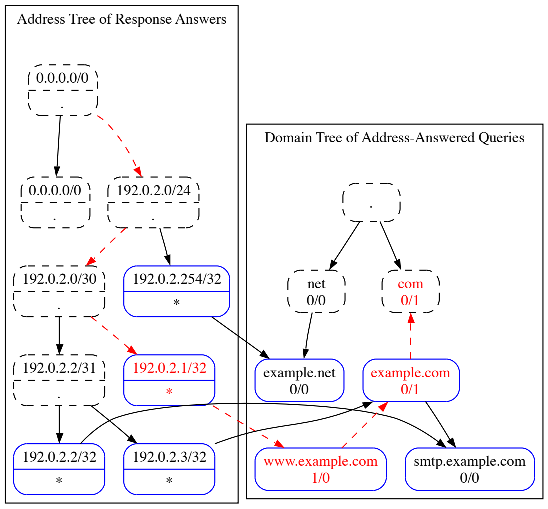

Figure 6: An example of TreeTop's combined use of address and domain trees to measure traffic by domain name. Here, traffic from 192.0.2.1 is observed; the dashed edges are traversed to locate the domain node counters to be incremented.

Figure 6 is an example of a combined arrangement of address and domain trees that TreeTop employs to measure IP traffic by domain names. [6] To initialize and maintain the trees and the requisite links between them, TreeTop observes DNS queries with valid A or AAAA answers, then adds (or updates) entries to both this address tree and this domain tree. Then, on observing subsequent IP traffic, an IP address (*e.g.,* from the packet header's source address field) is looked-up and a series of links are traversed to maintain exact and prefix counters in the nodes. In this way, traffic measurements by domain name are achieved with performance similar to that of standard IP exact match lookups.

In the DNS as it is commonly used today, it is certainly the case that multiple IP addresses are sometimes associated with a single domain name and that multiple domain names are sometimes associated with a single IP address. [7] TreeTop accommodates the former implicitly and accommodates the latter by finding a common prefix (*i.e.,* FQDN suffix) of the domain names, with the default case being "." (the root of the domain name hierarchy). For instance in Figure 6, if both "example.com" and "www.example.com" happened to resolve to address 192.0.2.1, we would link the node 192.0.2.1 to "example.com" and upon traversal, increment roll-up counters (not shown) rather than exact or prefix counters. This allows the measures of traffic involving IP addresses with multiple names to be represented in aggregated domain tree reports, even when there isn't an exact match to a single domain name.

We've described two instances of how TreeTop uses combinations of address and domain trees to maintain counts of DNS and general IP traffic in many dimensions. Other such arrangements, including domain trees in nodes of address trees and vice-versa, enable counting and tracking known domain names per client and the ability to determine the number of clients that know a given domain name. The former, *i.e.,* address trees of domain trees, is useful in general traffic analysis. When domain names are tracked on a per client basis, the measurement system is aware of which names are legitimately known (bounded by TTL) by each client. Also, the clients' domain name caching behavior can be used to determine whether those clients' applications are likely utilizing stale name to IP address translations. [8] The latter, *i.e.,* domain trees of address trees, may provide a more useful measure of a domain name's popularity than query rate, given that the query rate is increased for lower TTLs.

These analyses demonstrate the utility of these data structures when analyzing DNS traffic.

7. RESULTS

In this section we report our experiences using TreeTop to perform both off-line and real-time analyses. Our intent is to highlight the utility of our approach in terms of the scope and detail of information, rather than to show all possible reports. For clarity of presentation in time series plots, we focus on just the week of March 8, 2008. (Results for other weeks not during student recesses are similar.) Figure 7 shows an overview of all DNS traffic by type. In Figures 7, 8, and 9, note that the traffic types are shown "stacked" on top of each other so that none is obscured by another and so that the the highest values on the vertical axis are totals. Each type is examined further below.

[6] Some details have been omitted from Figure 6 including the Time-To-Live (TTL) of the name to IP relationship. TreeTop can use TTL information to report IP to name mappings at past points in time and to cull expired mappings from the data structures.

[7] A common case of multiple domain names being associated with a given IP address is a web server configured with many "virtual hosts", perhaps thousands, that use a single IP address.

[8] A common cause of stale name to IP address translations is that some applications use `inet_addr`, `inet_aton`, or `gethostbyname` APIs to resolve a name only at initialization and thus neither collect TTL information, nor resolve names and reestablish connections upon expiry of that domain name to an IP address translation.

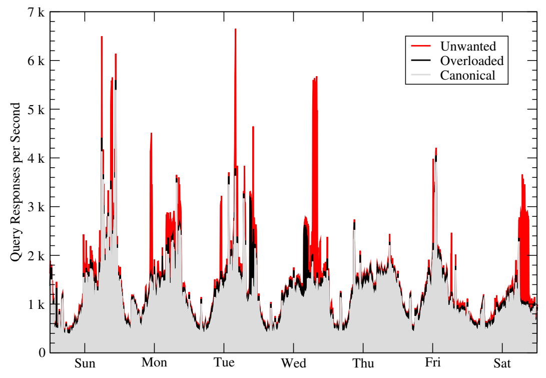

Figure 7: Clusters of DNS traffic during the week of March 3, 2008. Note that many spikes have been identified as unwanted and overloaded types.

7.1 Unwanted Traffic

We begin by focusing on unwanted traffic identified during the sample week. It is decomposed as four sub-clusters in Figure 8.

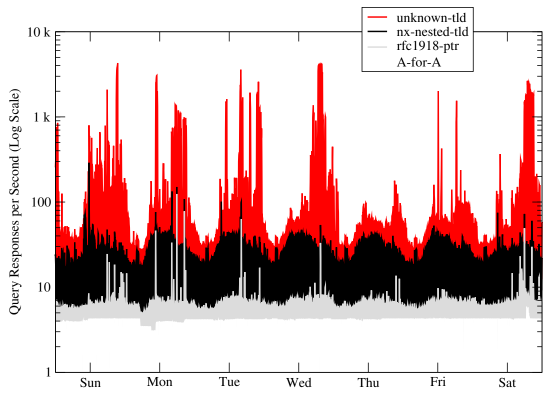

Figure 8: Unwanted DNS traffic during the week of March 3, 2008.

7.1.1 unknown-tld, rfc1918-ptr, and A-for-A Traffic

The traffic categories unknown-tld, rfc1918-ptr, and A-for-A are identified by existing filters in dnstop and are described thoroughly in [35] as well as briefly below. We include them here since they significanlty represent one of our three primary traffic types (*i.e.,* unwanted) and serve as a point for comparison to results published in earlier studies.

The unknown-tld queries are those for TLDs that are not

Type	Queries/Sec	Active Clients
unknown-tld	197 (87.3%)	530
nx-nested-tld	22 (9.8%)	310
rfc-1918-ptr	2 (1.1%)	78
A-for-A	4 (1.8%)	15
any	226 (100%)	n/a

Table 3: Distribution of unwanted DNS traffic types during the week of March 3, 2008. Values are averages shown with their respective percentages of the total unwanted traffic.

officially recognized by Internet governance organizations.

The rfc1918-ptr queries are PTR queries requesting names for private IPv4 addresses that exist in one of the private IP address ranges specified by RFC 1918. These are misdirected queries except within the private network using that IP address range. (The campus network in which our DNS servers reside does not use these addresses.)

A-for-A queries are queries for addresses with a query name string that already contains an address and are typically due to a bug in one resolver implementation. In Figure 8, note the lack of diurnal fluctuations in the level of A-for-A traffic. This indicates that this traffic's sources are "always on" which agrees with the likely source being a buggy resolver in a server's operating system.

7.1.2 nx-nested-tlds Traffic

The unknown traffic category of nx-nested-tlds, meaning nonexistent nested Top Level Domains, are queries resulting in NXDOMAIN response codes that have a query string that appears to contain a domain name ending in a known TLD embedded with an FQDN. For instance, TreeTop would count a query for "www.example.com.example.com" that results in NXDOMAIN as an nx-nested-tld.

These queries typically stem from resolver's search list feature that, for convenience, allows users to enter names that are not FQDNs, *i.e.,* a either a single label or a partially qualified domain name. For instance, in the aforementioned example, a user within "example.com" might configure their machine to search "example.com", then they could connect to "www.example.com" simply by referring to it as "www".

An issue arises when querying for names like "www.example.com" when there is a search list configured. Technically, since it does not end in a "." (which would make it "rooted" and thus an FQDN), "www.example.com" is considered a partially qualified domain name. The negative impact of this is that a resolver might first a lookup for 'www.example.com.example.com", and upon failure, lookup "www.example.com" which will succeed. From the user's point of view, everything works, however two queries were performed, one unnecessarily.

RFC 1535 [16] addresses this issue by prescribing that when a "." exists in a specified name, it should be assumed to be a FQDN and should be tried as a rooted name first. From our measurements however, it is clear that not every name server or resolver does this: much of the nx-nested-tld traffic was due to queries for resolvable FQDNs in our university's domain that are incorrectly being nested by appending our university's domain again (presumably from a resolver search list). A second significant source of such traffic was a campus mail server performing black-list queries with a slight misconfiguration. Since most black-list queries result in a negative response (NXDOMAIN), a single missing "." at the end of a black-list's domains name (thus making it only a partially qualified domain name and therefore a candidate to apply the search list) can cause the NXDOMAIN query to be retried after appending a domain in the local machine's resolver search list.

The high volume of nx-nested-tlds we observe are unnecessary repeated queries (that always fail with NXDOMAIN) and are most often due to persistent misconfigurations or ill-behaved resolvers rather than simply typos in FQDNs.

7.2 Overloaded Traffic

Next, we present an analysis of overloaded DNS traffic during a typical week. We've identified two kinds of overloaded DNS traffic: black-list and dnsbugtest traffic summarized in Figure 9 and Table 4 and described in more detail below.

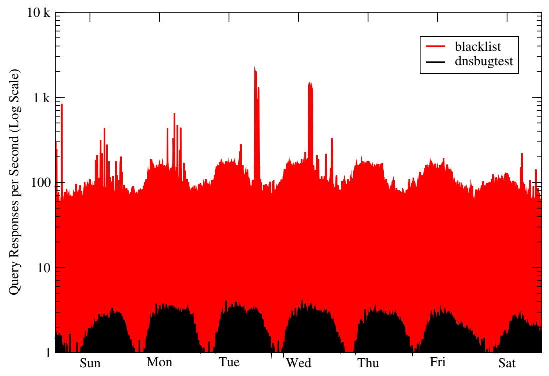

Figure 9: Overloaded DNS traffic during the week of March 3, 2008.

Type	Queries/Sec
blacklist	122 (98.4%)
dnsbugtest	2 (1.6%)
any	124 (100%)

Table 4: Distribution of overloaded DNS traffic types during the week of March 3, 2008. Values are averages and their respective percentages of the total overloaded traffic.

7.2.1 *Black-list Traffic*

Through the period of this week, TreeTop identified 220 domains as black-list domains. Consideration of the names showed about 10% of these seem likely to be false positives. [9] False negatives arise for black-lists that are in use, but that never answer in the affirmative, *i.e.,* always result in NXDOMAIN. The accumulated NOERROR and NXDOMAIN query reply rates are shown in Figure 9 and Table 4.

7.2.2 *dnsbugtest Traffic*

A second type of overloaded DNS traffic that we identified is what we call "dnsbugtest" traffic. Described in [6], a system intentionally sends a malformed DNS query to a server, in an attempt to determine the quality of service that server is providing. Based upon the response, an assessment is made, and an appropriate action may be taken to either rely upon or avoid that service.

As seen in Figure 9, dnsbugtest traffic is strongly diurnal. This suggests that it is directly linked with user behavior. We believe the dnsbugtest technique is used by a commercial implementation of Zero Configuration Networking (Zeroconf [31]) and is thus tied to mobile computing.

7.3 Canonical Traffic

One of our main interests in the canonical DNS traffic is in the query names and the resulting A or AAAA answers of successful queries. It is this traffic that is likely the precursor to IP traffic to and from the host IP addresses in the answers. We present decompositions of this portion of the canonical traffic as hierarchical graphs.

7.3.1 *DNS Queries for Addresses*

In Figure 10, we show the domain tree hierarchy of query names which were accompanied by address answers in DNS response traffic. We can discern the following from this graph depicting ten minutes of DNS traffic:

- 492,586 address queries were answered and are represented in graph. (This is the $prefix\_count$ from the "." root node.)
- Popular web services including Facebook, Google, and Weather.com, represented approximately 15%, 5%, and 4% of those queries.
- Most of the answered queries for those services were for sub-domains. (This can indicate how the services content is distributed or how the service is load-balanced using the DNS.)
- 47% of the answered queries for "com" sub-domains were rolled-up because those sub-domain's query counts did not exceed the 3% aggregation threshold. (This is the middle percentage, under the *roll_up_count* value in the "com" node.)
- Within the campus, only queries in the IT department's domain, "doit.wisc.edu," rivalled the quantities of queries to those commercial services.
- No TLDs other than "com," "edu," and "net" had subdomains with 3% of the answered queries.

[9]One exemplary false positive was similar to "10.mx.example.com" and resolved to 127.0.0.1. This "example.com" domain was mistaken for a black-list because it does contain a nested TLD: "mx" is Mexico's country-code TLD (ccTLD). In actuality, this was a Mail-eXchanger (MX) for "example.com," inexplicably configured with a localnet IP address. False positives can be reduced by considering only those domains with a sufficient proportion of NXDOMAIN responses as black-list candidates.

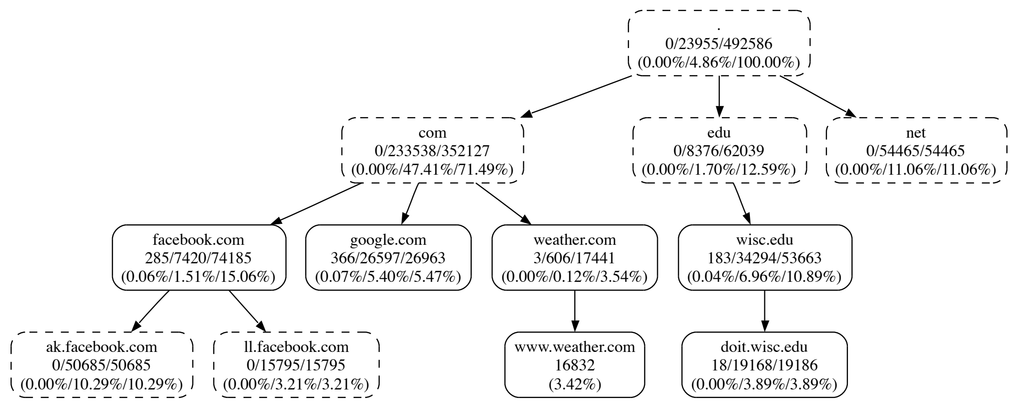

Figure 10: A TreeTop graph of query domains answered with Addresses during 10 minutes beginning at 1900 hours, Wednesday, March 5, 2008. The slash-separated query counts and corresponding percentages are $exact\_count/roll\_up\_count/prefix\_count$. The aggregation threshold was 3%.

7.3.2 IP Traffic by Domain

Lastly, we present sample general IP traffic measurements by domain, as implemented in TreeTop by the method described in Section 6.

In Figure 11, we show a TreeTop graph prepared in *real-time* by running TreeTop on a single workstation to monitor that workstation's own incoming traffic. Note that this graph has the same hierarchical structure as that in Figure 10, but instead of counting answered queries, the graph in Figure 11 counts bytes received from source hosts known to the workstation by each node's given domain name.

We see, in Figure 11, that inbound traffic for this web browsing session was received primarily from "facebook.com," "collegehumor.com," and "youtube.com." Note that this includes both HTTP and HTTPS traffic; the latter demonstrating the capability of our technique to identify the host names associated with the sources of encrypted traffic with payloads that can not be externally examined to determine the URL host name. [10]

Lastly, in Figure 11, a small portion of the traffic (0.15%) was from IP sources addresses for which no domain name was know. This traffic is counted in the "unnamed" node and includes the DNS requests themselves (since a host's DNS server is necessarily identified by IP address), and would also include, for instance, any web traffic from URLs specifying hosts by IP address rather than DNS-resolvable host name.

This example demonstrates how the combination of just transport layer information and the associated DNS traffic can be used to measure and classify IP traffic in general. Thus our technique avoids payload dependencies in traffic classification in situations when the payload is simply unavailable, *i.e.,* when the traffic was either encrypted or was recorded without payload (as in IP flow data).

[10]For HTTP traffic, host names can often be identified using the "Host" field; this information can not be externally observed in HTTPS traffic, but can be inferred by our processing of the corresponding DNS query responses.

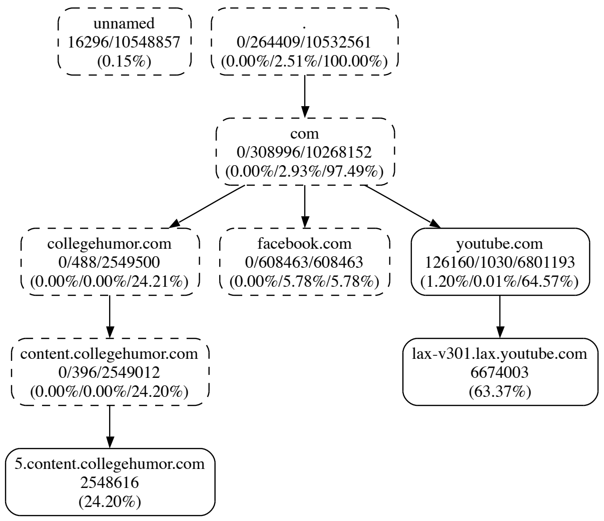

Figure 11: A TreeTop graph of traffic destined for a single workstation during a 5-minute web browsing session. The values shown are volume of *bytes received* from the domain specified in each node; the aggregation threshold was 5.5%.

8. FUTURE WORK

Our on-going activities are focused on enhancing our analysis methods to expose further details of network behavior, and on expanding our understanding of the relationship between DNS query analysis and information provided by other standard traffic monitoring methods. Based on TreeTop's ability to monitor general IP traffic by domain, we will quantify the amount of IP traffic that does and does not appear to use DNS names as service identifiers. This will allow us to evaluate the efficacy of our methods to classify IP traffic continually on a large scale, for instance by monitoring links that aggregate traffic for an entire enterprise or organization.

We have not yet specifically isolated zero configuration networking and service discovery traffic. Some of these techniques are RFC-defined extensions to DNS and others are experimental. These include DNS-Based Services Discovery (DNS-SD [4]) and Multicast DNS (mDNS [5]). Because of its unique characteristics, such traffic warrants identification by clustering; we have yet to determine whether it is best considered a sub-cluster of overloaded or canonical traffic.

We are evaluating the use of our methods in middleboxes applications including firewalls, NIDS, and Intrusion Protection Systems (IPS). In addition to the benefit of using the user-visible identifiers (domains rather than IP addresses), there are advantages to this for combined IPv4 and IPv6 environments and transition efforts. This is because the same DNS name can be used as a service identifier with both IP versions, whereas this is not the case for IP addresses.

Finally, the authenticity of domain name to IP mappings is of paramount importance in some applications of our methods. In this work, we restricted ourselves to monitoring DNS traffic involving our own trusted servers. However, we plan to apply our methods to determine the trustworthiness of other DNS servers. In part, this can be done by identifying discrepancies amongst the passively observed answers returned to client hosts, such as those resulting from cache poisoning. There are other opportunities to discover both undesirable and malicious network activities.

Conclusion

In this paper we present a set of novel techniques for analyzing DNS query traffic. The goal of our work is a deeper understanding of general network traffic and of unusual or unwanted traffic that can have a negative impact on networks. Unlike prior efforts that have focused on using DNS traffic to identify specific behavior (*e.g.,* bots that use fast flux), we take a general approach to query analysis by using data-driven cluster analysis to expose coarse-to-fine characteristics of network traffic associated with the queries. The specific classes of DNS queries that we focus on in this paper are canonical queries consistent RFC-intended behaviors, overloaded queries commonly associated with black-listing services, and unwanted queries that will never succeed and are thus superfluous. Our clustering methods are context-aware and they are oriented around the hierarchy inherent in both IP addresses and domain names, and enable users to specify the desired level of analysis detail. We implement our clustering methods in the TreeTop tool which can be applied to DNS query traces off-line, near real time, or in real time to streams of DNS queries. We use a set of DNS queries collected in our campus network over a period of three months to demonstrate the capabilities of our methods. Our analysis shows how TreeTop can expose the rich diversity of general network traffic, the significant use of black-listing services and the characteristics of unwanted traffic. We believe that these tests highlight the novelty and utility of our methods.

Acknowledgments

We thank Mike Blodgett, Michael Hare, and Jesse Thompson for their assistance in understanding the campus DNS traffic. We also thank Cristian Estan for his counsel. Kenjiro Cho, Damian Ivereigh, Paul Vixie, and Duane Wessels, each authored freely-available software that was useful in our implementation. This work was performed in cooperation with the University of Wisconsin - Madison's Division of Information Technology; we thank Hideko Mills for her support. Lastly, we are thankful to the program committee and anonymous reviewers for their suggestions that helped to improve this paper.

This work was supported in part by NSF grants CNS-0347252, CNS-0627102 and CNS-0716460. Any opinions, findings, conclusions or other recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the NSF.

9. REFERENCES

[1] Spam and Open Relay Blocking System. `http://www.sorbs.net`.

[2] J. Abley and K. Lindqvist. Operation of Anycast Services. IETF RFC 4786, December 2006.

[3] N. Brownlee, K. Claffy, and E. Nemeth. DNS Measurements at a Root Server. In *Proceedings of IEEE Global Telecommunications Conference (Globecom '01)*, San Antonio, TX, December 2001.

[4] S. Cheshire. DNS Service Discovery. `http://dns-sd.org`.

[5] S. Cheshire. Multicast DNS. `http://multicastdns.org`.

[6] S. Cheshire. Method and Apparatus for Detecting Incorrect Responses to Network Queries. United States Patent 20060253612, 2006.

[7] K. Cho, R. Kaizaki, and A. Kato. Aguri: An Aggregation-Based Traffic Profiler. In *Proceedings of the Workshop on Quality of Future Internet Services (QofIS '01)*, Coimbra, Portugal, 2001.

[8] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[9] D. Dagon, N. Provos, C. Lee, and W. Lee. Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority. In *Proceedings of Network and Distributed System Security Symposium (NDSS '08)*, San Diego, CA, February 2008.

[10] D. Dagon, C. Zou, and W. Lee. Modeling Botnet Propagation Using Time Zones. In *Proceedings of The Network and Distributed Systems Security Symposium (NDSS '06)*, San Diego, CA, February 2006.

[11] D. Deitrich, S. Gill, B. Greene, N. Long, and R. Thomas. Bogon Reference. `http://www.team-cymru.org/?sec=8&opt=25`, 2001.

[12] C. Dews, A. Wichmann, and A. Feldmann. An Analysis of Internet Chat Systems. In *Proceedings of ACM Internet Measurement Conference (IMC '03)*, Miami Beach, FL, October 2003.

[13] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification Using Clustering Algorithms. In *Proceedings of ACM SIGCOMM Workshop on Mining Network Data (MineNet '06)*, Pisa, Italy, September 2006.

[14] C. Estan, S. Savage, and G. Varghese. Automatically Inferring Patterns of Resource Consumption in Network Traffic. In *Proceedings of ACM SIGCOMM '03*, Karlsruhe, Germany, August 2003.

[15] C. Estan and G. Varghese. New Directions in Traffic Measurement and Accounting. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, August 2002.

[16] E. Gavron. A Security Problem and Proposed Correction With Widely Deployed DNS Software. IETF RFC 1535, October 1993.

[17] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. IETF RFC 4291, February 2006.

[18] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS Performance and the Effectiveness of Caching. *IEEE/ACM Transactions on Networking*, 10(5), October 2001.

[19] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy. Transport Layer Identification of P2P Traffic. In *Proceedings of ACM Internet Measurement Conference (IMC '04)*, Taormina, Italy, October 2004.

[20] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classication in the Dark. In *Proceedings of ACM SIGCOMM '05*, Philadelphia, PA, August 2005.

[21] R. Liston, S. Srinivasan, and E. Zegura. Diversity in DNS Performance Measures. In *Proceedings of ACM Internet Measurement Workshop*, Marseille, France, October 2002.

[22] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow Clustering Using Machine Learning Techniques. In *Proceedings of the Passive and Active Measurement Conference (PAM '04)*, Antibes Juan-les-Pins, France, April 2004.

[23] P. Mockapetris. Domain Names - Concepts and Facilities. IETF RFC 1034, November 1987.

[24] P. Mockapetris. Domain Names - Implementation and Specification. IETF RFC 1035, November 1987.

[25] D. Plonka. The TreeTop analysis tool. `http://net.doit.wisc.edu/~plonka/treetop/`, 2008.

[26] A. Ramachandran, N. Feamster, and D. Dagon. Revealing Botnet Membership Using DNSBL Counter-Intelligence. In *Proceedings of The USENIX Workshop on Steps to Reducing Unwanted Traffic in the Internet (SRUTI '06)*, San Jose, CA, July 2006.

[27] P. Ren, J. Kristoff, and B. Gooch. Visualizing DNS Traffic. In *Proceedings of the 3rd International Workshop on Visualization for Computer Security (VizSEC '06)*, Alexandria, VA, November 2006.

[28] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of USENIX LISA*, Seattle, WA, November 1999.

[29] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classication. In *Proceedings of ACM Internet Measurement Conference (IMC '04)*, Taormina, Italy, October 2004.

[30] K. Sklower. A Tree-Based Packet Routing Table for Berkeley Unix. In *USENIX Winter Conference '91*, Dallas, TX, January 1991.

[31] D. Steinberg and S. Cheshire. Zero Configuration Networking: The Definitive Guide. 2005.

[32] W. R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols.* Addison-Wesley, 1994.

[33] F. Weimer. Passive DNS Replication. In *Proceedings of FIRST Conference on Computer Security Incident Handling*, Singapore, July 2005.

[34] D. Wessels. dnstop. `http://dns.measurement-factory.com/tools/dnstop/`, 2002.

[35] D. Wessels. Is Your Caching Resolver Polluting the Internet? *Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality*, August 2004.

[36] D. Whyte, E. Kranakis, and P. Van Oorschot. DNS-Based Detection of Scanning Worms in an Enterprise Network. In *Proceedings of Network and Distributed System Security Symposium (NDSS'05)*, San Diego, CA, February 2005.

[37] V. Yegneswaran, J. Giffin, P. Barford, and S. Jha. An Architecture for Generating Symantic-Aware Signatures. In *Proceedings of USENIX Security Symposium*, Baltimore, MD, August 2005.

[38] B. Zdrnja, N. Brownlee, and D. Wessels. Passive Monitoring of DNS Anomalies. In *Proceedings of International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment*, Lucerne, Switzerland, July 2007.

[39] J. Zhang, J. Rexford, and J. Feigenbaum. Learning-Based Anomaly Detection in BGP Updates. In *Proceedings of ACM SIGCOMM Workshop on Mining Network Data (MineNet '05)*, Philadelphia, PA, August 2005.