Rendezvous-based Measurement,
Traffic Classification and Host Profiling

by

David J. Plonka

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2013

Date of final oral examination: August 23, 2013

The dissertation is approved by the following members of the Final Oral Committee:
Paul R. Barford, Professor, Computer Sciences
Suman Banerjee, Associate Professor, Computer Sciences
Remzi H. Arpaci-Dusseau, Professor, Computer Sciences
Benjamin R. Liblit, Associate Professor, Computer Sciences
Robert D. Nowak, Professor, Electrical and Computer Engineering

To my dear mother and my departed father for life-long support of my child-like wonder.

ACKNOWLEDGMENTS

In 1997, I moved to the beautiful city of Madison intending to continue my education in graduate school. However, I made the "mistake" of getting a job first. That was a good mistake, as I joined the Division of Information Technology (DoIT) at the University of Wisconsin-Madison and learned much from my coworkers and friends there, especially Bill "wej" Jensen, Paul Nazario, Mark Karls, Michael Hare, and Dale Carder. I thank my managers, Mike Dorl, Perry Brunelli, LaVonne Frank, Hideko Mills, and Annie Stunden who all gave me enough rope to hang myself by; luckily, I never quite managed to tie a proper knot.

After some years at DoIT, I had the benefit of meeting and collaborating with Professor Paul Barford. In 2006, he accepted me as his student and I left my full-time job to pursue a Ph.D. Paul's collegial demeanor, unwavering support, tenacious drive, and seemingly perennial optimism have buoyed my enthusiasm many times; I can scarcely imagine succeeding in this endeavor otherwise. Thank you, Paul.

I thank David Parter and Kimberly C. Claffy, both of whom played a significant part in encouraging me to follow a research and academic path.

I thank my research collaborators and coauthors: Mark Allman, Scott Campbell, Dale Carder, Archit Gupta, Jeff Kline, Sangnam Nam, Vern Paxson, Amos Ron, Andres Jaan Tack, and Vinod Yegneswaran. I also thank my friend, Mike Blodgett, who has helped me countless times in my work.

I thank my friend, Angela Thorp, who has always graciously helped me and fellow graduate students navigate the administrative intricacies of our wonderful department and university.

I am the beneficiary of the ideas and inventions of many free software authors and am thankful for their collaboration and/or friendship, especially: Nevil Brownlee (author of NeTraMet), Kenjiro Cho (author of Aguri), Mark Fullmer (author of flow-tools), Simon Leinen (author of SNMP in Perl), Tobi Oetiker (author of RRDTOOL), Paul Vixie (an author of BIND), and Duane Wessels (author of dnstop).

Lastly, I thank my good friends, Derek Inksetter, Dave Erickson, and Charles Thomas, who have each always been an inspiration to work, live, and laugh.

RENDEZVOUS-BASED MEASUREMENT,
TRAFFIC CLASSIFICATION AND HOST PROFILING

David J. Plonka

Under the supervision of Professor Paul R. Barford
At the University of Wisconsin-Madison

The Internet is arguably the largest and most complex artificial system in existence. Empirical measurement-based study is critical to examine this system's structure, behavior, and performance. These measurements inform how we understand, operate, and improve the Internet.

By design, the Internet is a packet-switched network in which conversations of many types are multiplexed. That is, the transmissions from unrelated parties share the resources of network elements whose operators may have goals that are at odds with those parties or the operators of other network elements. This sharing necessitates careful measurement that often involves Internet traffic classification, *i.e.,* observing Internet packet traffic in aggregate and then sorting packets into classes based on details. For example, a classifier might sort packets by their senders, receivers, or associated applications. However, during the Internet's continual evolution, traffic classification is confronted with numerous obstacles requiring the classification techniques to evolve as well. Classification varies with respect to the classes or categories to which traffic is assigned and with respect to the means by which that traffic, or a portion of traffic, is observed and analyzed.

In this dissertation we present a set of novel approaches to measurement based on the notion of Internet host *rendezvous*. Rendezvous is the means by which hosts initially discover each other in order to establish subsequent communication. We rely on the most pervasive rendezvous mechanism, the Domain Name System, a hitherto underutilized resource in the flexible, scalable, efficient, and accurate measurement of the Internet and its traffic. We present three case studies, each employing a different rendezvous-based technique in a different measurement problem domain. Our results demonstrate the effectiveness of this approach in situations where prior techniques are either inapplicable or unreliable.

Paul R. Barford

ABSTRACT

The Internet is arguably the largest and most complex artificial system in existence. Empirical measurement-based study is critical to examine this system's structure, behavior, and performance. These measurements inform how we understand, operate, and improve the Internet.

By design, the Internet is a packet-switched network in which conversations of many types are multiplexed. That is, the transmissions from unrelated parties share the resources of network elements whose operators may have goals that are at odds with those parties or the operators of other network elements. This sharing necessitates careful measurement that often involves Internet traffic classification, *i.e.*, observing Internet packet traffic in aggregate and then sorting packets into classes based on details. For example, a classifier might sort packets by their senders, receivers, or associated applications. However, during the Internet's continual evolution, traffic classification is confronted with numerous obstacles requiring the classification techniques to evolve as well. Classification varies with respect to the classes or categories to which traffic is assigned and with respect to the means by which that traffic, or a portion of traffic, is observed and analyzed.

In this dissertation we present a set of novel approaches to measurement based on the notion of Internet host *rendezvous*. Rendezvous is the means by which hosts initially discover each other in order to establish subsequent communication. We rely on the most pervasive rendezvous mechanism, the Domain Name System, a hitherto underutilized resource in the flexible, scalable, efficient, and accurate measurement of the Internet and its traffic. We present three case studies, each employing a different rendezvous-based technique in a different measurement problem domain. Our results demonstrate the effectiveness of this approach in situations where prior techniques are either inapplicable or unreliable.

rendezvous
Origin: Middle French, from
rendez vous **present yourselves**

Merriam-Webster Dictionary

The Internet is a massive and complex artificial system that has outgrown the bounds of its original design; it has evolved. Despite being man-made and ostensibly well-engineered, like other critical systems that evolve to accommodate expanded goals, it requires measurement and adjustment. Those who study the Internet employ empirical measurement methods in order to discern this system's structure, behavior, and performance. These measurements inform how we understand, operate, and improve the Internet. Methods for classifying and identifying key characteristics of network traffic, *i.e.,* techniques that identify *what* is being transmitted, have important implications for network management, traffic engineering and network security.

1.1 Motivation

Being a packet-switched rather than connection-oriented network, the Internet is a network in which conversations of many types are multiplexed. Furthermore, it is a network of networks such that these conversations traverse substructures that interconnect but operate independently. That is, the transmissions from unrelated parties share the resources of network elements whose operators may have goals that are at odds with those parties or the operators of other network elements. This sharing of fixed resources necessitates careful measurement involving Internet traffic classification, *i.e.,* the means by which the mix of packets in aggregate can be sorted out in detail by their senders, receivers, applications, and other properties. Traffic classification is the basis for a number of important tasks including usage trend analysis, planning of infrastructure changes, quality of service targets or guarantees, and the detection and prevention of suspicious and malicious activities. It may come as a surprise, then, that the Internet Protocol (IP) itself has no inherent basis for traffic classification. Instead, a community of researchers and operators devise classification methods for use in concert with their passive monitoring and anal-

ysis of traffic. These classification methods are not static. During the Internet's continual evolution, traffic classification is confronted with numerous obstacles requiring the classification techniques to evolve as well, both with respect to the classes and categories to which traffic is assigned and with respect to the means by which the traffic is observed and analyzed. The necessity for evolution is evidenced by the faltering performance of common early classification techniques, such as those that rely simply on TCP and UDP port numbers.

Internet traffic classification is challenging in a number of ways. The key challenge in accurately identifying different traffic types and their characteristics is that there is no inherent mechanism for this task. Instead, Internet protocols were designed with effective packet forwarding and reliable communication in mind. This has mainly been the purpose of packet headers' components, such as the IP 5-tuple: protocol number, source address, destination address, source port number, and destination port number. While there are some packet header features that have been introduced or remissioned for classification, *e.g.,* Type-of-Service (TOS) and Diff-Serv Code Point (DSCP), they are not particularly flexible and can represent only a modest number of labels, much fewer than the number of Internet applications and services that one may wish to identify or discriminate amongst in network operations. These fields are typically set by an initial traffic classifier to specify desired packet treatments during subsequent forwarding.

Also, popular and traditionally effective traffic classification techniques simply fail in some situations. While there are set of well-defined ports associated with well-known application protocols, *e.g.,* World-Wide Web / Hyper-Text Transport Protocol (WWW/HTTP), Simple Mail Transfer Protocol (SMTP), and Secure SHell (SSH), the Internet, itself, doesn't require the standard ports to be used. For instance, any application could use HTTP's TCP port number 80, and thus be misclassified as World-Wide Web traffic. Even when WWW traffic is correctly identified, the utility of the classification is dubious when it forms too large a class of traffic. For example, an administrator may wish to block traffic involving some WWW sites and expedite traffic involving others. Classifying such traffic based on port number, *e.g.,* the HTTP-Secure (HTTPS) protocol's TCP port 443, does not help meet that goal as the resulting class is too broad. Additionally, the increasingly common use of encryption makes key attributes of the packets unavailable. Indeed, with HTTPS, the Uniform Resource Locator (URL) is encrypted and unobservable to middle boxes, routers, or other network elements responsible for forwarding and performance. Today,

traffic between hosts can be easily encrypted, to provide privacy or to ensure the peer host's identity. The traffic can also be easily obfuscated to masquerade as another type of traffic. Since traffic payloads are arbitrarily arranged by user applications, they are not necessarily trustworthy as the basis for classification and their varying content presents performance challenges, both in the ability to inspect the full payload on high capacity links and in the myriad potential class identifiers that can overwhelm high performance database storage and retrieval when online operation is desired.

As Internet applications and uses evolve, we have seen a move to non-traditional patterns of communication. Traditionally, client hosts would discover the identity of a server or service, and connect to it on a well-known port. However, with the introduction of efficient and distributed peer-to-peer (P2P) file-sharing, that model is insufficient. Now, hosts often use proprietary or non-standard methods of discovering each other, and then communicate on ephemeral (non-standard) ports to exchange files. In the case where these files are not legal to exchange as in some jurisdictions, as is often the case for copyright music or video content, there is strong incentive for the participants to actively obfuscate their communications so as to avoid detection. In parallel, encryption of traffic has recently become prevalent both due to the increased awareness of the sensitive nature of the content (*e.g.,* personal financial information) and the threat of criminal compromise and due to increased host processing power that makes end-to-end encryption computationally feasible. These nascent changes in Internet traffic severely challenge the capabilities of existing traffic classification techniques.

As empirical evidence of these challenges, consider Figure 1.1 that shows IP traffic volume by application as classified using only packet header information (*i.e.,* the IP 5-tuple including port numbers) over approximately 8 years at the border of our university network. We see early on that File Transfer Protocol (FTP), HTTP, and P2P are the most prominent applications classified. In those years past, *e.g.,* circa 2000 and prior, this is indicative that port numbers could be used to classify the majority of network traffic, primarily due to the limited diversity of applications and the standard way in which client-server communications were established. However, as time passes, the most prominent class is simply HTTP and half or more of the traffic remains unknown (unclassified,

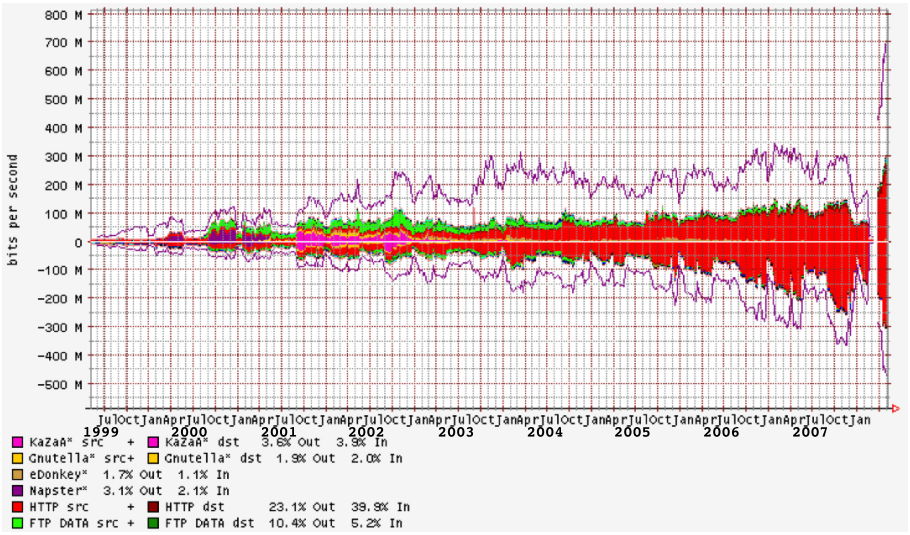


Figure 1.1: Classification of Wide-area IP traffic by volume over time, (1999–2008) at the University of Wisconsin-Madison.
Outbound traffic is shown above the horizontal axis and inbound traffic below. Traffic is classified based on protocol (TCP or UDP) and service port numbers that are well-known to be used by the given applications. The proportion successfully classified varies greatly and, unfortunately, decreases as time passes.

shown as transparent). Classification of P2P applications (by this method) decreases (*e.g.*, 2003 versus 2000). Overall, these phenomena develop: (i) WWW classification becomes increasingly dominant, (ii) the portion of unknown traffic increases, and (iii) P2P classification becoming dubious. Consequently, operators and researchers are left quite uninformed about exactly how the Internet is being used.

To address such classification short-comings in the face of changing Internet traffic, the network research and operations communities are motivated to revise their traffic classification methods and invent new techniques to remain informed on Internet use and capably provide reliable Internet service.

1.1.1 Motivation for Rendezvous-based Classification

Having outlined the general motivation for research and development of Internet traffic classification techniques, we are additionally motivated to develop a method that has the following characteristics as goals, driven by the three confounding empirical phenomena mentioned above:

- employ a classification scheme that is flexible to avoid classifying too much traffic into one common class, *e.g.*, WWW (HTTP and HTTPS), simply because it uses a popular transport protocol. To satisfy this goal, we are motivated to use an extensible, hierarchical classification scheme.
- employ a traffic-independent source of information so that it is (i) resilient to manipulation and obfuscation by the participants and (ii) performs well in situations where the traffic is encrypted or monitored by packet sampling. To satisfy this goal, we are motivate to depend on the host behavior prior to exchanging the traffic to be classified.
- be capable of reliably separating traditional client-server traffic from peer-to-peer traffic, so that traditional port-based and other classification methods can be applied only where applicable and not where they clearly yield false positives or negatives. To satisfy this goal, we are motivated to employ information about how hosts are introduced to each other, *i.e.*, the *rendezvous mechanism* that is starkly different in these situations.

1.2 Challenges in Traffic Classification

Given that traffic classification is of both a research and an operational interest, the criteria of (i) accuracy, (ii) scalability, and (iii) robustness are often those used to test and compare methodologies. [99] In light of these criteria, and to address the aforementioned problems with port-based classification, prior work has focussed on non-port based approaches to identifying network traffic including payload-based analysis, behavioral analysis, and clustering analysis. Payload-based approaches (*e.g.*, [46, 94]) 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. In addition to failing when monitoring encrypted or packet-sampled traffic, another 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 [60, 61, 97] focus on building statistical models of transport layer metrics such as connection duration and packet size to distinguish applications. These methods are limited to a set of previously observed behaviors, typically built-in to the classifier, and are challenged by

lack of information in those deployment situations where they do not have the ability to monitor absolutely all application traffic. (We discuss this in more detail in Chapter 2.) Cluster-based approaches such as [23, 49, 72] take the next logical step by using standard machine learning methods to divide traffic into groups based on similarity of transport layer characteristics. These methods are limited by their requirement to have well-labeled training sets from which to learn, thus have no automatic way to accommodate new protocols. 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, and that our approach, outlined below, can be use in concert with any of these methods or used as a leading indicator as to which of them may be appropriate.

1.3 Approach

In this dissertation, we present a novel approach to traffic classification based on the notion of Internet host *rendezvous*, *i.e.*, the method by which hosts discover each other in order to establish communication. We present three empirical measurement studies, each employing a different rendezvous-based technique in a different measurement problem domain. Our results demonstrate the effectiveness of this approach in situations where prior techniques are either inapplicable or unreliable.

In our approach, we tap a traffic-independent source of information and enable flexible organization of traffic types into arbitrary groups. Our classification methodology is based on monitoring and analysis of the traffic generated by rendezvous services that are used by Internet applications. A rendezvous service is typically operated independently of its clients and enables a client application to identify the IP addresses that are the target for communication and are known to the user by an alpha-numeric name. The canonical example of a rendezvous service is the Domain Name System (DNS), which is used by most Internet applications. The first step in our method is to build and maintain a table of active local clients by IP addresses and their respective target remote IP addresses along with the associated alpha-numeric names extracted from the locally-observed DNS traffic.

Next, we observe live traffic packet headers, *e.g.,* via packet capture, or observe a stream of flow-export records collected in the same enterprise for local and remote IP address pairs that match entries in that table populated with the preceding rendezvous information. If there is a match, the corresponding alpha-numeric (DNS) name is the basis for classification.

Both the type of rendezvous mechanism employed (*e.g.,* DNS, static, DHCP, algorithmic, etc.) and its intrinsic characteristics offer opportunities for detailed classification at a level that has not been possible with prior methods. For example, the simple fact that a given host employed the DNS to rendezvous with "www.example.com" via HTTP may allow both that client host and the exchanged traffic to be classified as WWW and prevent misclassification as P2P. And, since the DNS uses names that follow a well-known domain hierarchy, the DNS name hierarchy can serve as one way to organize and aggregate resultant traffic, but others are possible as well. For example, names can be organized into categories that are user defined, come from a standard source (*e.g.,* [7]), or are based on application type (*e.g.,* WWW, FTP, online social networks, or streaming) or by subject (*e.g.,* weather or sports). This ability to create arbitrary traffic groups may offer network operators significant flexibility in how they manage traffic going well beyond what port-based methods offer.

The key technical challenge in developing a DNS rendezvous-based classifier is that it must monitor host rendezvous traffic and, in a timely fashion, link the information gleaned with corresponding observations of the application traffic to be classified. For many an institution or enterprise, the typical scenario involves a set of client hosts that utilize a locally-designated recursive DNS service (often located in or near their Local-Area Network (LAN)) with those hosts' application traffic passing through some interesting observation point within a network element such as a high-capacity switch or border router. The observation point of the DNS rendezvous traffic need not be the same as that of the *target* traffic to be classified. With this model, we develop a software classifier that can accommodate parallel traces from multiple observation points.

Our overall development and presentation is stepwise and progressive. First, we study the DNS rendezvous traffic itself, separating the wheat from the chaff, *i.e.,* identifying and organizing, in a sort of high-performance database, the subsets of rendezvous information that can be usefully employed to perform subsequent traffic classification. This part of the approach is a key factor in performance because it enables us to populate a cache prior to classification, thus enabling operation in near real time even when monitoring traffic on

heavily-utilized, high capacity links.

Second, we directly apply the rendezvous information to the task of classifying Internet traffic in general. In this phase of study, we identify the classifiable and ostensibly unclassifiable traffic, thus determining the scope and limitations of the method. Next, we expand the method to overcome these potential limitations. Specifically, we use rendezvous information to profile hosts, so that we can meaningfully label traffic that is otherwise unclassifiable via the DNS directly, *e.g.,* that traffic arranged via a non-standard or unknown rendezvous mechanism.

1.3.1 Thesis Statement

We claim that Internet rendezvous information offers compelling advantages in Internet measurements, traffic classification, and host profiling. We further claim that traffic classifications and host profiles based on host and service rendezvous offer more fidelity and flexibility than labels from competing techniques and represent a *tacit ground truth* that differs from that used to validate prior classification and profiling techniques. Rather than comparing our methods' results to that of many prior classification techniques (because our methods yield different classes or classification labels), we demonstrate how our methods provide unique results and how they either complement prior methods or replace them in situations where those methods fail. We support these claims in three ways. First, we demonstrate the feasibility of our methods by implementing a tool called TreeTop for offline and online analyses. Next, we employ our tool in empirical studies to classify and measure traffic where classifications from prior methods results are either suspect and too coarsely-grained or where prior methods are simply not applicable, *i.e.,* to compare the performance of Internet services on Internet Protocol version 6 (IPv6) versus that on IPv4. Lastly, we develop an application-specific measurement tool called Timemail, and employ it to reverse-engineer email rendezvous information in longitudinal study, thereby exposing the structure and delay centers in the email delivery network over decades of operation.

1.4 Applications of Rendezvous-based Methods

Having developed our rendezvous-based methods, we apply them in novel ways. In one application, realizing that the only link between the same services

on IPv6 and IPv4 is the rendezvous mechanism, we present a novel way to compare availability and performance of these services during the deployment and simultaneous operation of the pervasive legacy protocol (IPv4) to the new protocol (IPv6) destined to replace it.

In our final study reported herein, we employ our rendezvous-based paradigm to study the email delivery network, past and present, by reverse-engineering rendezvous information from trace data even years after the communication occurred, thus demonstrating the unique nature and efficacy of our approach both for forward-looking online operation and offline trace and trending analyses.

1.5 Major Contributions

This thesis makes the following contributions:

- We introduce rendezvous-based traffic classification and host profiling that relies on inspection of clearly observable transport information present in packet headers in combination with selective inspection of rendezvous traffic payload information, generally utilizing the Internet's most common rendezvous service: the Domain Name System (DNS).
- Our methods represent a new privacy-respecting classification technique in that they (i) do not inspect user application traffic payloads, and (ii) consequently work in situations where those payloads are unavailable due to common monitoring limitations, *e.g.*, flow export, or where payloads are encrypted.
- We provide an approach that is uniquely capable in comparing IPv6 to IPv4 performance since the rendezvous mechanism is one of the few things common to an IP service operating on multiple IP protocol versions and is, thus, key to identifying similar application traffic over them both.
- Our method can be applied to expose structure and performance of the email delivery network based on hitherto unutilized rendezvous trace information present in email messages, enabling longitudinal study of this long-lived, pervasive form of electronic communication.
- Our technique has a number of novel applications, *e.g.*, (i) distinguishing traffic that is amenable to reliable classification by port numbers from that which is not and (ii) exposing cloud and Content Distribution Network (CDN) configurations that rely on indirection based on rendezvous. [25]

1.6 Broader Impacts

In addition to the aforementioned contributions, we foresee a number of potential broader impacts of our methods, including:

- Rendezvous-based classification methods may provide a finer-grained level of control by which an enterprise can specify packet treatments as in a firewall, middlebox, or router, *e.g.,* to prohibit unwanted traffic, such as that with unknown rendezvous mechanisms that may be malicious, or to expedite important or time-sensitive traffic.

- Since rendezvous-based classification does not inspect user application traffic payloads, it may be employed in situations where the Internet community is concerned about privacy. That is, it could be a key technique in situations where a policy, *e.g.,* between a government or Internet service provider and Internet users, specifies which portion of user traffic is private and which is available for determining packet treatment, *e.g.,* quality of service (QoS), and thus play a part in lawful intercept systems.

- Because rendezvous-based traffic analysis necessarily identifies the rendezvous mechanism employed, it could be used as an initial stage in multi-level classification, determining which subsequent classification mechanisms are appropriate. For instance, it may revitalize and bolster port-based classification schemes that have fallen out of favor because they often misclassify traffic for the peer-to-peer applications that have their own non-DNS-based rendezvous mechanisms.

- Since our classification techniques use rendezvous information that exists outside of the IP traffic itself, they provide labels for classification and measurement techniques that span IP protocol version changes, including IPv6 now, and possibly others in the future. This additional layer of indirection allows comparison and contrast of traffic in current and future networks having different address namespaces (or numberspaces) during simultaneous operation and in situations where network hosts have mixed host-attachment configurations, *e.g.,* IPv4 only, IPv6 only, or dual-stack.

- Lastly, by utilizing the rendezvous information, *i.e.,* the aspect of the Internet that revolutionary cloud-based services and CDNs exploit to match clients with content and services, our techniques accommodate

exactly those technologies that are being used to improve scalability and availability today and likely continuing in foreseeable future. This may be particularly important as cloud services and CDNs play an increasingly important role in determining Internet performance, whereas this lay with ISPs and Internet eXchanges in the past that were largely agnostic with respect to rendezvous as they rely solely on transport information to forward traffic.

1.7 Outline of this Dissertation

The remainder of this dissertation is organized as follows. We review existing classification techniques from related work and compare or contrast our work to these in Chapter 2. In Chapter 3, we provide an overview of the DNS, describe our analysis software that we employ in empirical study of DNS traffic, and present results demonstrating the feasibility of using DNS rendezvous traffic for general traffic classification. In Chapter 4, we report the results of a larger empirical study that classifies traffic and profiles hosts for a campus office and residential user population and demonstrate visualization techniques for hierarchical rendezvous-based classification. Next, in Chapter 5, we expand our analysis software framework and present a methodology that applies our rendezvous-based traffic classification technique to compare services' IPv6 performance to those services' IPv4 performance. In Chapter 6, we present a new system for studying the structure and performance of the email delivery network that involves reverse-engineering email rendezvous information from trace data stored in saved email messages, past and present. Finally, in Chapter 7, we summarize with our concluding remarks and propose future work.

> The idea of using DNS for identifying the FQDN and subsequently the associated service and resource is kind of trivial in hindsight, but not so obvious. Given all the literature in traffic classification, I have not seen anybody exploit this to build a system around it.
>
> anonymous reviewer of [25], 2012

Our work is informed by much significant prior work involving traffic classification and measurement, visualization techniques, the DNS and other host rendezvous methods, and Internet applications or protocols. In this section we provide an overview of these and highlight the prior work that introduces or elucidates each of them.

2.1 Traffic Classification

Internet traffic classification methods have been proposed and evaluated in a number of prior studies and utilized in many more. To the best of our knowledge, none prior to ours has proposed a rendezvous-based approach. However, earlier techniques relate to our work in a number of ways outlined below, and serve as a basis for comparison and evaluation of our method.

2.1.1 In-Host Monitoring

Monitoring of traffic involving Internet end-hosts can be performed from within, from without, or a combination of these. While *in-host* methods are invasive, they set the highest standard of ground truth in traffic classification. For example, Liao *et al.* [68] use context information gathered within hosts to identify the exact application associated with each network connection in a network of hosts under complete administrative control. Furthermore, Caballero *et al.* [29] employ static and dynamic program analysis to automate the process of reverse-engineering custom application protocols (*e.g.*, botnet command-and-control

traffic) thereby determining the provenance of the IP addresses, for instance, those addresses that an application uses to initiate remote communication. While of the utmost accuracy and usefulness for system administrators of individual computers [107], in-host techniques are prohibitively difficult to employ in most network environments due to factors including user privacy, in-host agent liability, and agent development or deployment costs. Thus in practice, alternative methods are typically employed that inspect only the packets that traverse external network elements and observation points.

2.1.2 In-Network Monitoring

In most traffic classification work, the data, in the form of packets, are observed in the network as they traverse the interfaces of hosts, routers, and switches using packet sniffer software, built-in network equipment features, or dedicated passive wire taps.

Fairly recent work by Kim *et al.* [63] provides a thorough overview and performance comparison of popular *in-network* traffic classification methods and implementations from the literature and from practice. We refer the reader to [63] for a survey of other prior work involving traffic classification based upon port [4, 83], payload [99], host-behavior [61], or flow-features [76]. Won *et al.* [115] proposes a hybrid classification technique involving both Deep Packet Inspection (DPI)-based signature matching and transport-layer session behaviors. Our technique also involves a similar combination, although ours employs DPI only for the rendezvous traffic, therefore both our development and computational costs are lower. Madhukar and Williamson [69] provide a sample study involving P2P traffic classification using (i) port-based, (ii) signature-based, and (iii) host transport-layer behavior approaches. These latter works highlight the trade-offs and complications in evaluating candidate classification methods.

2.1.2.1 Port-based Classification

Port-based classification techniques employ *shallow packet inspection* (SPI), *i.e.*, any traffic monitoring technique which utilizes packet information from no deeper than the packet's header such as the transport-layer information. For Internet traffic, this generally means the header fields that are involved in either forwarding and demultiplexing IP traffic. These include field values such as *IP protocol, destination port number, source port number, destination IP address, and*

source IP address, this often being referred to as the IP "5-tuple." Port-based classes are thus often named by IP services names, *e.g.,* `ftp`, `http`, `ssh`, or groups of these services, *e.g.,* "Login" combining `ssh`, `rlogin`, and `telnet`.

As early application-specific protocols were developed, they were each assigned a reserved port number; for instance, the telnet service is assigned TCP port 23. These port number assignments are maintained by the Internet Assigned Numbers Authority (IANA) and are generally referred to as "well-known" port numbers. These well-known numbers, agreed upon by consensus, are the basis for why port-based classification can be effective. While there is no guarantee that traffic on a given well-known port is actually that of the application or service to which it is assigned, for decades it was simply the case that these conventions were generally followed and, thus, port-based classification was usually sufficient well into the 1990s.

Because it utilizes the packet transport information directly, port-based classification is based in the TCP/IP protocol specification itself and the way an implementation (in an operating system) delivers traffic to applications [106]. TCP/IP implementations use this transport-layer information to route traffic and determine the host and the application on that host that should receive (or has sent) any given packet. As such, classification based on port-based SPI is inherently computationally-feasible in a monitoring system because it minimally requires only the same high-performance data structures and algorithms already used by network elements such as routers, switches, and hosts to distinguish separate traffic flows and forward them appropriately. For instance, the FlowScan [83] measurement and classification tool performs lookups using code from MRT [84], which in turn, utilizes the data structure employed in the BSD Unix TCP implementation. [101]

Common packet sniffing tools such as tcpdump [2] and wireshark [3] (formerly ethereal), expose and rely on this transport-layer information to determine how to display the payload of a packet. For instance, if a given packet is destined for TCP port 80 (the "http" service), tcpdump will attempt to decode and display the packet's payload as a Hyper-Text Transport Protocol (HTTP) World-Wide Web request or if it were sourced from TCP port 80, as an HTTP response. Likewise, a typical port-based classifier (such as CoralReef [4]) would simply classify such packets as Web traffic.

SPI is also the basis for *packet filters* [31, 33], a simple form of firewall or access control. Numerous classification tools use transport-layer information gleaned via SPI in network elements. [4, 27, 45, 83]

2.1.2.2 Signature-based Classification

Signature-based or "application signature-based" classification is typically used to identify specific applications or protocols (*e.g.,* BitTorrent) or malicious exploits (*e.g.,* root shell) by inspecting the arbitrary payload of packets or sequences of packets (*e.g.,* "P2P" combining BitTorrent, Gnutella, and others).

Signature-based classification techniques therefore employ *deep packet inspection* (DPI): a monitoring technique that examines the packet payload, *i.e.,* the full content of the packet, to detect or classify traffic of particular types. For Internet traffic, this typically involves either looking for values at specific packet locations beyond the packet header or scanning for strings at arbitrary locations within the packet.

Sen *et al.* [99] describe how to identify P2P traffic by scanning for application signatures via DPI. Intrusion Detection System (IDS) software such as Bro [81] and Snort [94] can be employed to perform DPI. For instance, Wanner [111] describes how to write Snort rules that will report hosts that appear to be using the BitTorrent P2P file sharing protocol when the string "torrent" occurs in the name of the object fetched via an HTTP GET command.

Some commercial products used in broadband ISP environments, such as those from Sandvine [10] and ipoque [8], use proprietary and hardware-enhanced techniques to classify traffic by application signatures. Such approaches often hit performance barriers on very high-capacity links and are vulnerable, *i.e., likely ineffectual*, in the face of packet payload obfuscation (*e.g.,* by introducing chaff or employing deliberate TCP segmentation). [1] Recent work to improve performance in both software and hardware-based implementations of DPI [102, 103], serve as evidence that it is an ongoing problem. Furthermore, the effectiveness of signature-based classification is somewhat limited when monitoring by packet sampling and it is rendered practically useless when packet payload is encrypted. Thus, our work, instead of relying solely on DPI, attempts to minimize reliance on it by applying DPI only to the easily (or feasibly) separable, unencrypted, low-volume rendezvous traffic.

2.1.2.3 Host Transport-Layer Behavior

Plonka [83] introduced a stateful classification method to identify elusive passive (PASV mode) ftp application traffic and Napster P2P application traffic by

[1] The file sharing community has proposed BitTorrent protocol options to obfuscate remote peer IP address information. [56]

inspecting only transport-layer information in flow-export data. Both of these application protocols (that employ unreserved or ephemeral port numbers) are classified based upon the participating host's transport-layer behavior, *i.e.*, the context determined by the host's earlier communications rather than by well-known port numbers.

The Napster P2P traffic classification in [83] is clearly an ancestor of our rendezvous-based method and is similar in spirit, if not implementation. Like rendezvous-based traffic classification, it relies on knowledge of the way in which peer hosts introduced themselves to each other to participate in a particular service; in this case, that service involved the exchange of MP3 audio files via TCP typically on unreserved port number pairs. Also, similarly, it uses this knowledge to maintain state information that is subsequently used to classify application traffic. To the best of our knowledge, this is the only classification method that could accurately identify and measure Napster traffic without application traffic payload inspection, which was generally infeasible at the time. While quite effective, classification was limited to the popular Napster service because of a requirement that the network administrator have "insider knowledge" of the IP address prefixes within which centralized Napster's servers operated and specify these in the FlowScan measurement tool's configuration. In contrast, our current general rendezvous classification method automatically discovers services' IP addresses and therefore does not require insider knowledge nor special access to the services whose traffic it classifies. Later, Karagiannis *et al.* attempt transport-based identification of traffic involving decentralized P2P services with some success. [60]

Karagiannis *et al.* [61] subsequently introduced BLINC, a more general classification method based on host-behavior. BLINC classifies traffic involving a given host by matching the host's communication behavior to application server behavior signatures (that are built in to BLINC.) The method we describe in this dissertation is similar to BLINC in that we do not rely directly on ports nor target traffic payload and is also similar in that our profile-based method employs a kind of "social" behavior of each host. (By "target traffic," we mean the traffic that one intends to classify and measure. In contrast, rendezvous traffic is typically not the target because it generally does not consume much resource; it merely supports the use of network applications that can consume significant resources.) However, our traffic classes are based on innumerable domain names rather than a small, fixed set of application groups. Also, our direct method, described in Section 4.3, neither employs heuristics (based on

previously observed behaviors) nor requires tuning. It has been found [63, 108] that BLINC's graphlet approach experiences problems when the target traffic is sampled ("1 in n" packets) or when the target traffic is not observed symmetrically at a gateway near the end-hosts. In contrast, our technique is robust at high traffic rates (where sampling is often employed) because it gleans social behavior of hosts from the complete, low-volume DNS rendezvous traffic rather than attempting to divine social behavior only from observation of the high-volume target traffic itself.

2.1.3 Hierarchical Traffic Classification

Both Cho *et al.* [34] and Estan *et al.* [50] describe and implement traffic measurement systems that use the hierarchical IP address space to profile or classify traffic in aggregates. Somewhat similarly, our method employs a hierarchy, but instead uses the hierarchical domain name space to form aggregates classes; domain names have advantages in terms of readability and persistence over IP addresses. Additionally, our classification groups hosts with similar profiles, and thus bears some similarity to aggregation in these prior works, but without our having to rely on structural cues from the hosts' IP addresses and their associated blocks assigned by the Internet Assigned Numbers Authority (IANA) and Regional Internet Registries (RIRs) or the blocks present in Internet routing tables.

2.1.4 Domain Name-based Traffic Classification

Some commercial products perform traffic classification and filtering using identifiers that often contain domain names. Products such as Websense [12] and SmartFilter [9] inspect application traffic payload for identifiers such as URLs and may, optionally, perform reverse DNS lookups. Our method differs in that it observes the content of the participating clients' DNS rendezvous traffic and thus can be effective in environments when it is infeasible to inspect the target traffic (*e.g.,* due to traffic volume, encryption, or policy). Alexa Internet [7] provides web traffic metrics labeled by domain name, such as top site lists and demographics. Their service is web-specific and observes Uniform Resource Locators (URLs), whereas our work considers all traffic and observes fully-qualified domain names (FQDNs). However, in our work we sometimes employ Alexa's categories as a convenient basis for our operator-defined host profiling.

It bears noting that "reverse DNS," *i.e.,* lookups in the `inaddr.arpa` and `ip6.arpa` domains by IP address for a hosts' domain name, are typically unreliable and computationally infeasible in online analyses. [25] While the ability of the DNS to query by IP address for a name is potentially extremely useful, this mapping is not well-maintained in today's Internet. The primary reason that this is unreliable is that there is simply no mechanism enforcing that reverse mappings match their forward mapping nor even that a reverse mapping exists. It seems likely this will remain the case, as the administration of forward and reverse mappings are even delegated to different entities: one being that which received the address allocation and the other being that which acquired a domain name. Since our method utilizes only the forward DNS lookups, it avoids the pitfalls that reverse DNS lookup presents.

2.1.5 Host profiling

While not exclusively a traffic classification technique, end-host profiling or "host profiling" has been used in prior works to classify hosts and, in turn, to *inform* traffic classification, especially when the traffic is particularly difficult to observe or classify. End-host profiling is the classification of an Internet host based on its behavior, often with the hope that it can be used to predict its future behavior. A complication of applying end-host profiling to the task of traffic classification is that it is coarse-grained (at the host, rather than session transport). A given host might exchange many types of traffic at the same time or over time; thus a given host might simultaneously fit multiple classes making the corresponding traffic classification somewhat ambiguous. (In Chapter 4, we employ only a small set of host classes, thus Venn diagrams are sufficient to present our results.)

Trestian *et al.* [108] perform traffic classification by first classifying end-hosts based on query results from a database of information that is available publicly on the web, *i.e.,* by supplying a host's IP address as a query string to Google search, with the hope that it contains correct and timely information about end-hosts of interest. The inspiration for their work is similar to ours in that in order for Internet communication to progress, an end-host must somehow discover the IP address with which to communicate. Their classification based on matching words in domain names could be applied to create aggregates for our rendezvous-based approach.

2.2 Passive DNS Monitoring

There are a number of prior works that involve the passive monitoring and measurement of DNS traffic for a variety of purposes. Wessels *et al.* [113, 114] provide a tool (*dnstop*) to measure DNS traffic by volume per client. Based on that tool, our work introduces and develops *TreeTop* [85], a tool that implements domain name-based traffic measurement in aggregate. We utilize TreeTop to track and report individual client's DNS activity and our results differ in that we apply that DNS information to label both traffic for traditional client-server applications (*e.g.*, World Wide Web and Streaming) and peer-to-peer traffic (*e.g.*, BitTorrent, Skype, and Massively Multi-player Online Gaming), the latter by integrating end-host profiling.

In [112], Weimer introduced a tool that populates a database from passive DNS traces and effectively identified abusive behavior including botnet activity. Likewise, Zdrnja *et al.* [117] record DNS trace information to a database and subsequently identify DNS anomalies including "fast flux" domains typically associated with botnet activity [43, 91]. Similarly, we passively monitor DNS query responses and store them in a sort of online database in core memory. However, rather than focussing on only anomalous or suspicious behaviors, our work differs in that we monitor all DNS query responses and/or apply it to profiling hosts and classifying those hosts' traffic.

The recent work of Bermudez *et al.*, who developed DN-Hunter [25], is the most similar to ours; this is a consequence of their work being based on ours. While their goals differ and they study Content Distribution Networks and do not assess service performance, they employ our rendezvous-based method from prior work [85, 86] to passively monitor clients' DNS traffic and annotate those hosts corresponding application traffic flows in an online fashion. Likewise, Drago *et al.* study cloud storage structure and performance by employing DN-Hunter and our DNS rendezvous method. [48, 73]

Lastly, Spring and Huth [104] consider the consequences of passive DNS monitoring on end-user privacy. Our method necessarily makes use of passive DNS monitoring, but we claim that it preserves end-user privacy more than some competing methods in that it inspects only the rendezvous traffic necessary for communications to be established. This is somewhat similar to a postal service inspecting the sender and recipient address on an envelope so that it can be delivered, but not inspecting the ostensibly private communication within. In this way, our method significantly differs from DPI-based

classification techniques that either inspect users' application traffic payloads or that fail when those payloads are obfuscated or encrypted.

2.3 IPv6 Performance Measurement

Measurement study of IPv6 has primarily focused on uptake and deployment. Indeed, there are multiple groups around the world that regularly publish reports and offer web sites that track IPv6 prevalence in the Internet (*e.g.*, see [30] for a compilation of sources) including measurements during World IPv6 Day. Similarly, Claffy identifies both available and desired sources of data that can be used to track IPv6 deployment in [36]. Several studies and reports have used both active and passive measurements to characterize the extent of IPv6 penetration at different points in time. Cho *et al.* use ping and traceroute measurements to remote hosts that have both IPv4 and IPv6 addresses as a means for identifying path problems [35]. At the highest level, their host identification method is similar to ours (*i.e.*, based on DNS names), our analysis method is more general and completely passive, including the gleaning of domain names from DNS responses. A similar study was conducted by Zhou and Mieghem in [118]. Karpilovsky *et al.* use a variety of measurements from Internet registry logs to NetFlow traces to understand IPv6 including the extent of application use [62]. Colitti *et al.* describe a methodology to measure adoption and performance of IPv6 from the perspective of a web site operator in [37]. Their work differs in that it is based on latency measurements.

Several studies have analyzed World IPv6 Day from different perspectives. Sarrar *et al.* use large NetFlow and packet traffic traces gathered from two different vantage points to characterize IPv6 volume, application mix and tunneling protocols on IPv6 day [98]. A study by Labovitz provides similar results in [66]. To the best of our knowledge, ours is the first study to passively assess services' IPv4 versus IPv6 performance.

While we augment our method with a transport and storage framework for annotating flow export records somewhat similarly to DN-Hunter [25], our goal to assess IPv6 performance by comparing it with that of IPv4 is unique. To the best of our knowledge, ours is the first study that presents a scalable and robust methodology to make such assessments for services with which clients rendezvous via the DNS.

2.4 Email Studies

In the final application of our method (presented in Chapter 6), we reverse-engineer email rendezvous information based on trace data present in stored email messages to discern the structure and performance of the email deliver network. Prior work has reported on malicious use of email (*e.g.,* spam [59, 90, 116]), on packet traffic volumes by service [71, 79], on email performance and loss by *active* measurements [19, 20], and on social relationships from email sets similar to those we use [70, 109]. Our work differs in that we study the structure and performance of the email delivery network over time by sifting and winnowing through the voluminous trace data present in stored email messages to discover the roles of hosts, *i.e.,* to profile hosts, such as Mail-eXchangers (MXers) that participated in delivery and then assess their performance in forwarding email.

2.5 Other Related Work

While we primarily focus on DNS-based rendezvous, prior work has described alternative rendezvous mechanisms. For example, Morris *et al.* propose a distributed hash table-based mechanism [77] and Walfish *et al.* propose replacing DNS with another mechanism for the World-wide Web in [110]. Baset and Schrulzrinne [24] and Rossi *et al.* [96] reverse engineer and infer Skype's application-specific rendezvous mechanism. There are other standard rendezvous protocols, *e.g.,* DHCP Options [21], DNS SRV [55], SIP [58], and P2P variant works in progress (*e.g.,* P2PSIP [6]). These protocols and others are candidates for extending our methods, especially for online rendezvous protocols where rendezvous information is easily separable (from voluminous application traffic) and where the rendezvous traffic has a standard "clear text" format.

Lastly, our work employs a number of visualization techniques to present results. Presentation of classified traffic volumes in time series are generally of the style adopted in [83]. Shneiderman [100] originated the treemap visualization that we employ to represent hierarchical data; alternatives such as directed graphs and Venn diagrams can not always be constructed to reliably present such data proportionately by volume.

> interesting idea of using DNS analysis for traffic classification
>
> anonymous reviewer of [85], 2008

In this chapter, we lay the foundation for rendezvous-based measurement, traffic classification, and host profiling by performing a detailed study of DNS queries and responses. Our goals are (i) to understand the breadth of information present within, and (ii) to select just the DNS traffic useful for classification and apply it to the measurement and analysis of general network traffic in an online fashion. [85]

3.1 Overview

The Domain Name System (DNS) is a one of the most widely used services in the Internet and it is relied upon by most Internet applications. In this chapter, we investigate the question of how DNS queries could be used to provide important and unique insights on network traffic. Our motivation for this chapter of our work is 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; 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 chapter, we refer to this type of DNS traffic as "overloaded"), which are critical for spam checking, but increasingly used for other purposes. 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 chapter, 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 will soon be made available to the community.

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. Also, 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. Our evaluation highlights both the coarse and fine level of detail that can be revealed by our method.

Ultimately we find that, having carefully separated and indexed the canonical DNS rendezvous traffic and associated host IP addresses, we can have TreeTop capture and classify *application* traffic by employing the rendezvous information gleaned earlier. 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.

3.2 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 [117], 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 chapter.

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."

To perform queries DNS client hosts typically use 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 email, 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 [106] or [74, 75] for a thorough introduction to DNS packet structure and service semantics.

[1] See Figure 3.5 for a graphical example of a portion of the DNS hierarchy.

3.3 Empirical Datasets

In this chapter, 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 [42].

For off-line analysis, we capture DNS traffic exchanged between campus client hosts and the campus' recursive anycast [18] DNS service. Our university operates a recursive name service consisting of four geographically dispersed server machines that answer queries received with one of the service's two IP addresses, which are in different campus network prefixes. As such, this recursive anycast DNS service represents 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 chapter, we consider a traffic trace from January 8, 2008 through April 21, 2008. Tables 3.1 [2] and 3.2 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 3.1 presents the traffic as a time series. Note the rich set of characteristics involving multiple dimensions in the measurement data. (The weeks of the year labeled 2, 3, and 12 are during the January inter-semester and spring recesses, thus had traffic volume due to lower active client counts.)

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 3.1 are not expected to add to 100% because any given active client can issue multiple types of queries in a measurement interval.

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 3.1: 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.

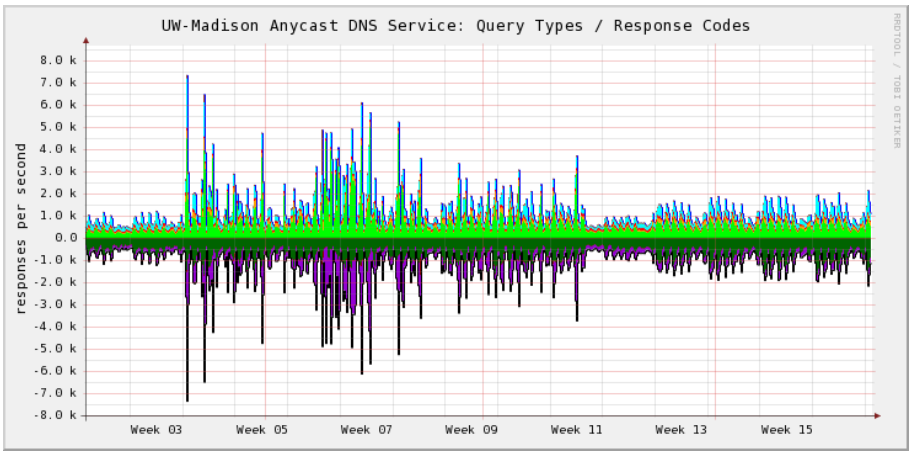


Figure 3.1: DNS query and response rates, January 8, 2008 through April 21, 2008. Query rates by type are plotted above the horizontal axis and the corresponding response rates by code are plotted below. See Tables 3.1 and 3.2 for the rate values.

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

Table 3.2: DNS response distribution: average rates by response code.

3.4 Analysis Method

Our initial observation about the measurement data, presented in time series in Figure 3.1, 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.

3.4.1 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 fields 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 [34] and [50], 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.

3.4.2 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 "blacklists." 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 "reputation" lists exist in many varieties including Real-time Blackhole Lists (RBLs), DNS Black-Lists (DNSBLs), DNS White-Lists (DNSWLs), Uniform Resource Iden-

tifier 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 3.5, 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, and 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 clus-

ters 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 [112] and [117].) The ability to store, lookup, and report IP addresses 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.

3.5 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.

3.5.1 Data structures

3.5.1.1 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 [101], but with additional features.

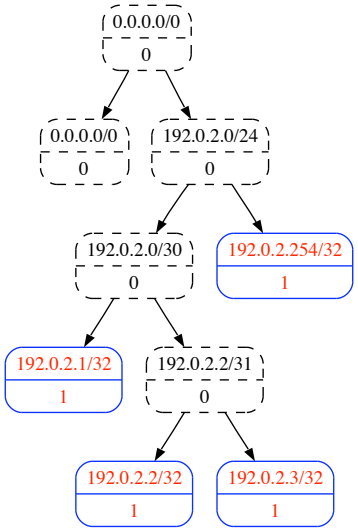


Figure 3.2: An address tree containing four IPv4 addresses, each with a count of 1. Internal nodes are shown with dashed lines and occupied nodes with solid lines.

An example address tree is shown in Figure 3.2. The address tree is based on the tree implementation in Aguri that is thoroughly described in [34], 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.

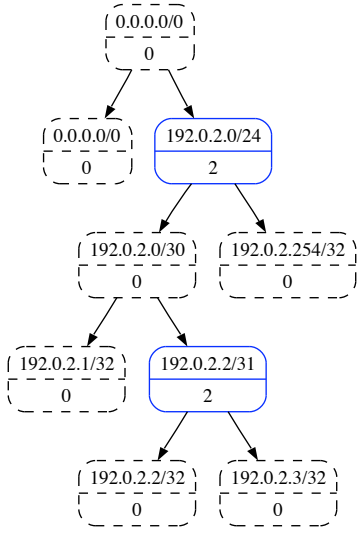


Figure 3.3: An address tree containing two IPv4 prefixes, each with "rolled-up" counts of 2. This is the result of aggregating the tree shown in Figure 3.2 with a 40% threshold.

- 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 3.3 shows a 40% threshold aggregation of the tree shown in Figure 3.2; 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.

3.5.1.2 The Domain Tree

Inspired by the effective use of prefix tries within the IP number space in both Aguri and AutoFocus [50], 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 3.4 is an example of a domain tree structure. Domain trees differ from address trees in the following ways:

- 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-nonspecific domain name labels. RFC-1035 [75] 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-

[3] See [34] 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." [57]

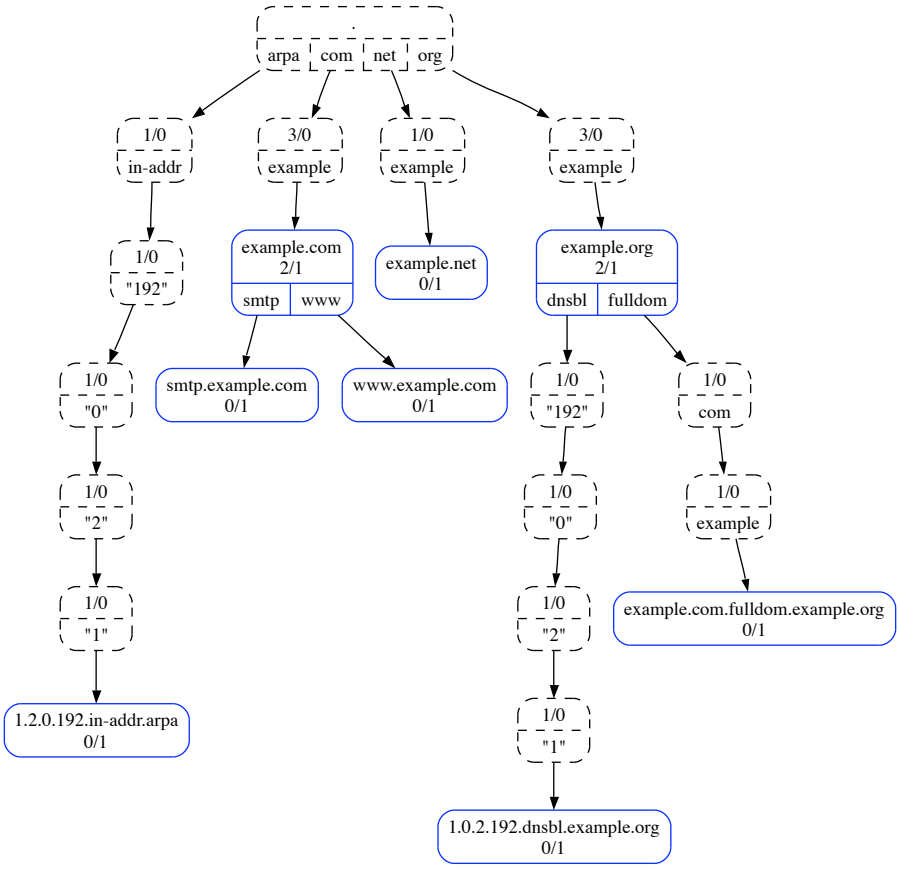


Figure 3.4: A domain tree counting references to 8 fully-qualified domain names (FQDNs). Various prefix and exact counts for those entries are shown, slash-separated, in the nodes as *prefix_count*/*exact_count*.

black tree [40], so that it is both space efficient (versus a hash) and exhibits predictable performance. [5]

- 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.

[5] Note that this portion of the domain tree does not play a prominent role in determining its functionality; sub-structures other than a red-black tree may exhibit better performance here.

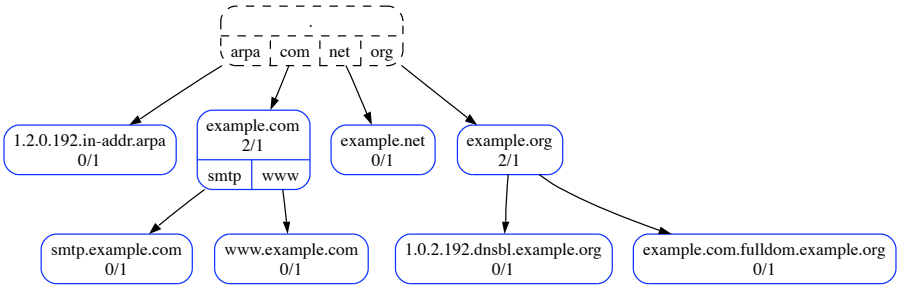


Figure 3.5: A domain tree containing 8 FQDNs; this is a level-compressed presentation of the tree shown in Figure 3.4.

- 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 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 3.5 shows a level-compressed report of the same domain tree as depicted in Figure 3.4. 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.)

3.5.2 The TreeTop Analysis Tool

We developed a DNS and general traffic analysis tool called *TreeTop*. TreeTop is implemented as a patch to dnstop [114], 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, text reports (tabular or hierarchical) 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.

3.5.2.1 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 3.4, 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 3.4 and 3.5. 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 email. 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) [44].

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 (i) an embedded IP address (as a reversed dotted-quad, *i.e.*, "1.0.2.192") or (ii) 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.

3.5.2.2 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.

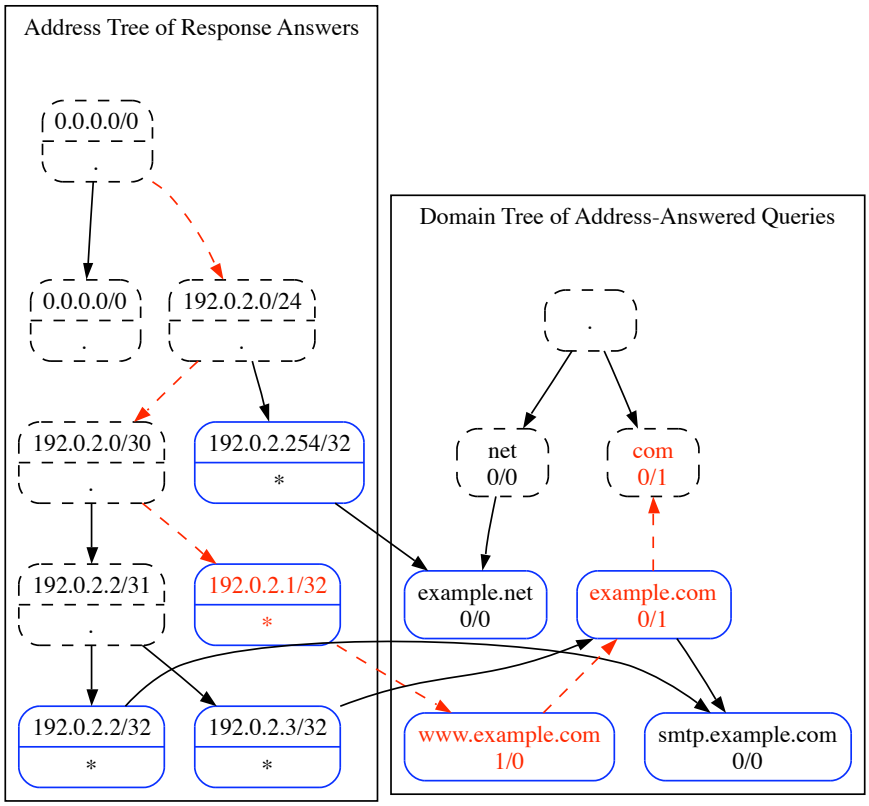


Figure 3.6: An example of TreeTop's combined use of address and domain trees to measure traffic by domain name. Here, traffic from 192.0.2.1 is observed; the dashed edges are traversed to locate the domain node counters to be incremented.

Figure 3.6 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 look-ups.

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 3.6, 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

[6] The Time-To-Live (TTL) is not shown in Figure 3.6. TreeTop could 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. Popular sites such as "blogspot.com" and "ytmnd.com" employ this technique.

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 with lower TTLs.)

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

[8] A common example of an application using a stale name to IP address translation is when it utilizes `inet_addr`, `inet_aton`, or `gethostbyname` APIs to resolve names at initialization and thus neither collects TTL information, nor resolves names and reestablishes connections when a domain name expires.

3.6 Results

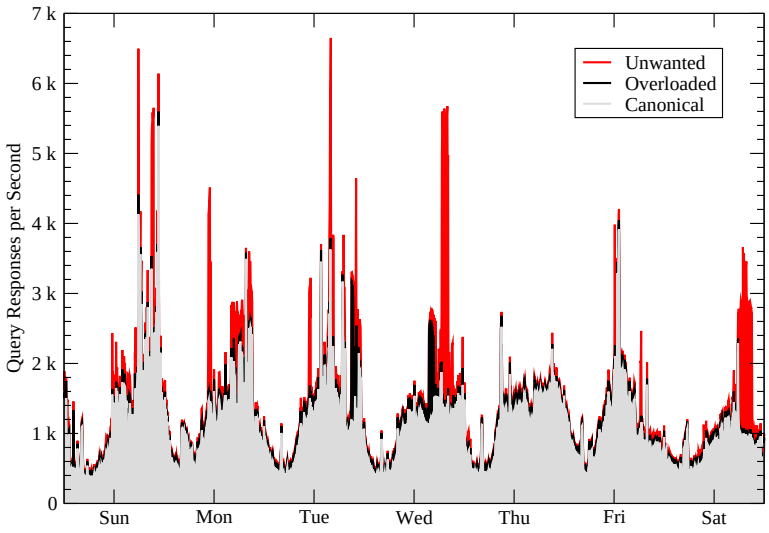


Figure 3.7: Clusters of DNS traffic during the week of March 3, 2008. Note that many spikes have been identified as unwanted and overloaded types.

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 3.7 shows an overview of all DNS traffic by type. In Figures 3.7, 3.8, and 3.9, 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.

3.6.1 Unwanted Traffic

We begin by focusing on unwanted traffic identified during the sample week. It is decomposed as four sub-clusters in Figure 3.8.

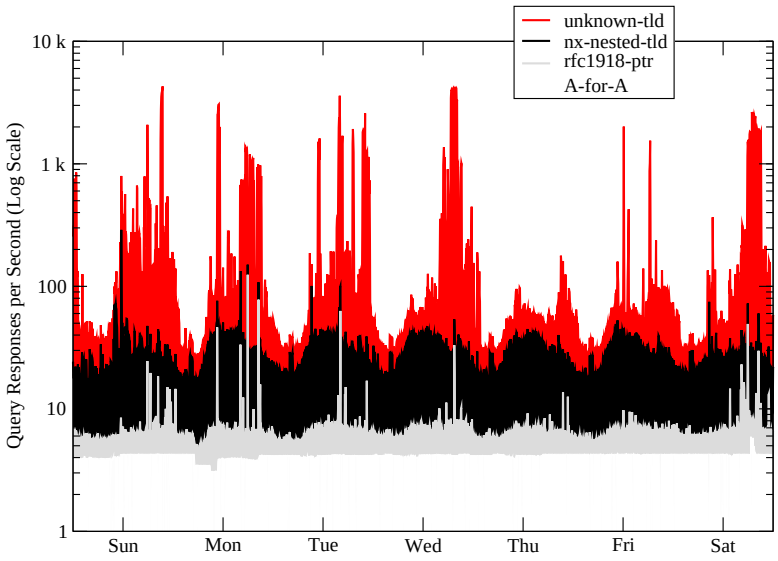


Figure 3.8: Unwanted DNS traffic during the week of March 3, 2008.

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 3.3: 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.

3.6.1.1 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 [114] as well as briefly below. We include them here since they significantly 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 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: the Microsoft Windows NT stub resolver. [114] In Figure 3.8, 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.

3.6.1.2 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 [51] 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 misconfiguration or ill-behaved resolvers rather than simply typos in FQDNs.

3.6.2 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 3.9 and Table 3.4 and described in more detail below.

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

Table 3.4: 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.

3.6.2.1 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

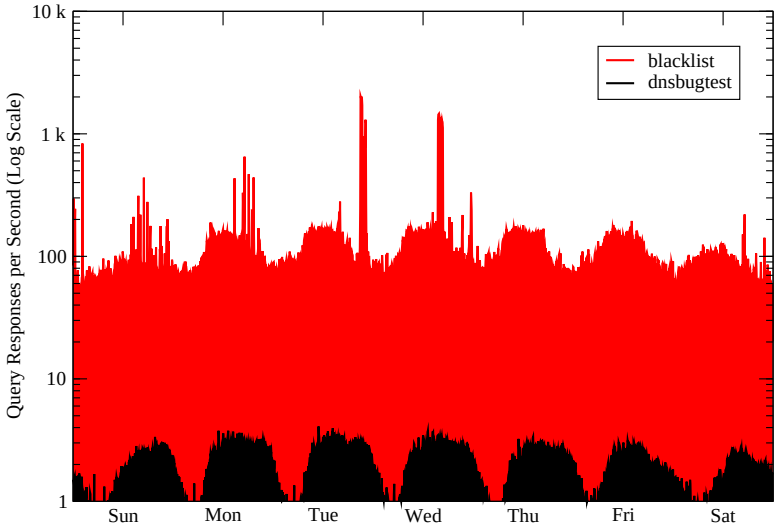


Figure 3.9: Overloaded DNS traffic during the week of March 3, 2008.

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 3.9 and Table 3.4.

3.6.2.2 dnsbugtest Traffic

A second type of overloaded DNS traffic that we identified is what we call "dnsbugtest" traffic. Described in [32], 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.

[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.

As seen in Figure 3.9, dnsbugtest traffic is strongly diurnal. This suggests that it is directly linked with user behavior. We believe the dnsbugtest technique is used by Apple's Bonjour Service, an implementation of Zero Configuration Networking (Zeroconf [105]), and is thus tied to mobile computing.

3.6.3 Canonical Traffic

One of our primary interests in the canonical DNS traffic is in the query names and the resulting A or AAAA answers. 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.

3.6.3.1 DNS Queries for Addresses

In Figure 3.10, 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 suggests 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" and "net" had sub-domains with 3% of the answered queries.

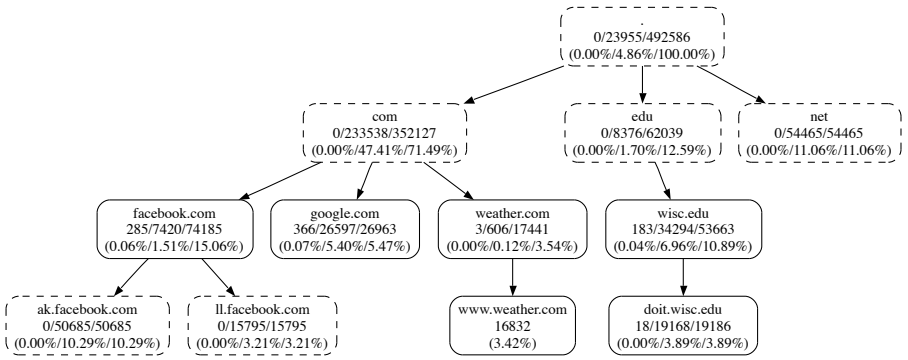


Figure 3.10: A TreeTop graph of query domains answered with Addresses during 10 minutes beginning at 1900 hours, Wednesday, March 5, 2008. The slash-separated query counts and corresponding percentages are *exact_count/roll_up_count/prefix_count*. The aggregation threshold was 3%.

3.6.3.2 IP Traffic by Domain

Lastly, we present sample general IP traffic measurements by domain, as implemented in TreeTop by the method described in Section 3.5.

In Figure 3.11, we show a TreeTop graph prepared in *real-time* by running TreeTop on a workstation to monitor that workstation's own incoming traffic. Note that this graph has the same hierarchical structure as that in Figure 3.10, but instead of counting answered queries, the graph in Figure 3.11 counts bytes received from source hosts known to the workstation by each node's given domain name.

We see, in Figure 3.11, that inbound traffic for this web browsing session was received primarily from "facebook.com," "collegehumor.com," and "youtube.com." 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 3.11, a small portion of the traffic (0.15%) was from IP sources addresses for which no domain name was known. 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

[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 determined by our processing of the corresponding DNS query responses.

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 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 IP flow data).

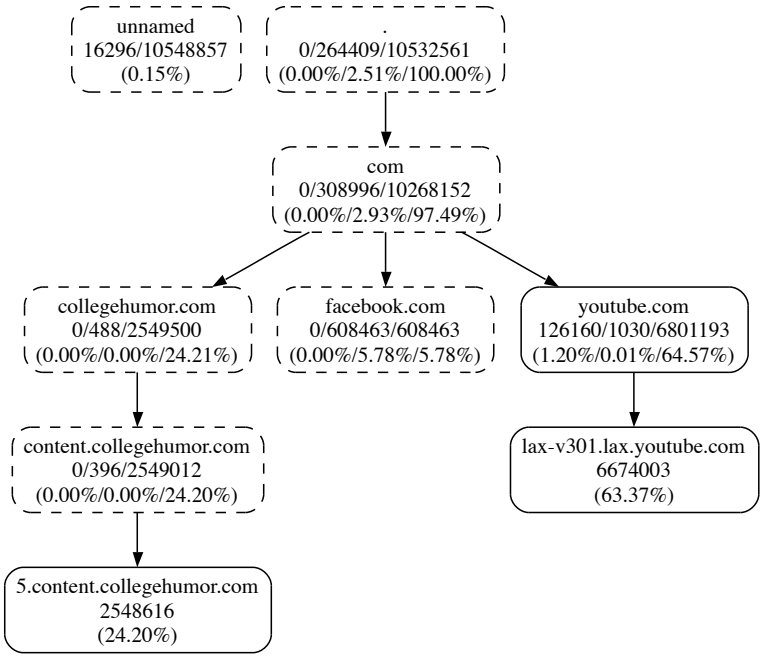


Figure 3.11: A TreeTop graph of traffic destined for a single workstation during a 5-minute web browsing session. The values shown are volume of *bytes received* from the domain specified in each node; the aggregation threshold was 5.5%.

> The approach is interesting, and for the reasons Plonka and Barford note, namely that port-based metrics no longer work. The gains they see using DNS information is quite impressive.
>
> anonymous reviewer of [86], 2011

In this chapter we fully exercise our rendezvous-based methods as implemented in TreeTop, developed in the prior chapter. We do this as a case study by analyzing the traffic involving large user-populations' client hosts. We also define our *direct* and *indirect* DNS rendezvous-based classification methods, and develop a host-profiling strategy that is particularly useful to classify traffic that is not arranged via DNS rendezvous. [86]

4.1 Overview

The past decade has seen an explosion of new network applications such as peer-to-peer (P2P) file sharing, online social networks, gaming, and VoIP. Each application requires certain network resources so that users have a satisfactory experience. The past decade has also seen many forms of malicious network use and abuse such as denial-of-service attack bot nets, phishing scams, and thefts due to compromised host or protocol insecurity. Both the ability to discriminate between application types in live traffic streams and to identify suspicious hosts is critical in order to ensure the desired level of application performance and reliability in an enterprise.

In this chapter, we describe a new method for traffic classification that taps a traffic-independent source of information and enables flexible organization of traffic types into arbitrary groups. Our classification methodology is based on monitoring and analysis of the DNS traffic that most Internet applications use for host rendezvous. Specifically, through careful tracking of client IP addresses, alpha-numeric domain names, and answer IP addresses in rendezvous traffic, we apply classification labels to end-hosts and their traffic reported by flow-export data. Additionally, we present the notion of *host profiling* as a method

for expanding traffic classification in cases where there is not a direct match between rendezvous traffic and application traffic.

To assess and evaluate the capabilities and effectiveness of our method, we collect DNS query-response traffic and flow-export records from our campus network infrastructure for over a year. Analysis of this data exposes many interesting features such as well known diurnal behavior, frequent spikes in DNS traffic, and a qualitatively different DNS behavior for subgroups within the user population in a case study we present that considers traffic for a typical day. We separate two distinct user populations: a large office/staff group and a large residential/student group. We characterize and contrast the DNS and wide-area traffic of each group showing that, while the general types are similar, the quantity of each type is dramatically different. In particular, over 90% of the office traffic is classified by domain name. Less of the residential traffic can be classified by name, ostensibly due to the use of P2P and other applications that do not rendezvous based solely on the DNS. Serendipitously, however, we find that any DNS rendezvous classification discriminates traditional client-server application from P2P application traffic.

4.2 Empirical Data sets

In this chapter we are interested in applying information gleaned from DNS queries and corresponding replies, exchanged between end-hosts and their trusted recursive name servers within an enterprise, to the task of classifying that enterprise's wide-area traffic. To this end, we monitor our campus' traffic at two observation points: (i) the campus clients' name servers, and (ii) one of the campus border routers that handles much of wide-area traffic including that for the commodity Internet. We perform full packet capture at the campus domain name servers, and collect packet-sampled flow data at the border router. [1] Thus, the payload of the DNS traffic is recorded, but the application traffic payload is not. Our interest is in the "canonical" DNS traffic, *i.e.,* the standard DNS traffic expected to precede application traffic that consists of a query by fully-qualified domain name (FQDN) and an answer containing one or more IP addresses associated with the query name.

Prior work has shown that traffic classification results can vary widely based on the trace traffic mix and observation point [63]. As such, while we monitor traffic for a single institution, we select two of its end-host/client populations that have very different characteristics, namely an *office* and a *residential* population. To expose the details, we present results for a single representative day. (Classification results from other days are consistent with the results reported here.)

Table 4.1 summarizes the characteristics of the data sets. We studied, in detail, the traffic on one typical day selected arbitrarily: April 17, 2009. Both the office and residential data sets consist of (i) all the recursive DNS traffic between end-hosts and the campus DNS service and (ii) the packet-sampled flow records collected at the campus border that represent wide-area traffic (see also Figure 4.4); only flow records involving campus hosts for which we've seen recursive DNS traffic involving the trusted campus DNS server are considered.

4.2.1 Office Traffic

The "office traffic" involves a group of staff employees on the campus. The office users are bound by the campus Appropriate Use Policy for information technology resources (that tolerates incidental personal use) and their end-hosts

[1] The flow data is based on a 1 in 1024 packet sampling rate using the "cflowd" feature on a Juniper router with 10-gigabit Ethernet interfaces; we report all our target traffic volume measurements by bits or bytes (approximated by multiplying sampled values by 1024).

Data Set	Clients	Unique FQDNs	DNS Reply Pkts	DNS Reply Volume (ave. bps)	Wide-Area Out / In Volume (ave. bps)
Office	614	19.4 K	560K	12.2K	753K / 5.66M
Residential (subpop.)	9,819 (5,583)	(143 K)	15.7M	360K	244M / 276M

Table 4.1: Characteristics of 24-hour data sets analyzed. The average wide-area traffic volume is estimated from packet-sampled flows. The parenthesized values are for a residential subpopulation that was used for the TreeTop-based results in Section 4.4. From the inbound and outbound volume values, we see that the office population primarily consumes wide-area Internet content, whereas the residential population both consumes and provides a significant amount of content.

are typically owned by the university and located in campus offices with wired Ethernet connections. During the course of the day under study, we observed 614 end-hosts with an average (over 24 hours) of 180 active hosts performing DNS queries per 5 minutes. The office wide-area traffic and DNS traffic volume and rate values are shown in Tables 4.1.

4.2.2 Residential Traffic

The "residential traffic" involves a subset of the students living in residence halls on a campus. The residential users are bound by the same Appropriate Use Policy as the office users, but their end-hosts are privately-owned and located in private residences that have wired Ethernet connections. During the course of the day under study, we observed 9,819 end-hosts with an average (over 24 hours) of 1,886 active hosts performing DNS queries per 5 minutes. The residential wide-area traffic and DNS traffic volume and rate values are shown in Tables 4.1.

4.3 Analysis Method

In this chapter we analyze and classify the DNS and wide-area (application) traffic using an improved version of the TreeTop tool [85]. Specifically, we've enhanced TreeTop to track and report the relationship between IP addresses and domain names on a *per-client* basis.

In short, TreeTop processes pcap traces of combined DNS and application traffic, requiring the payload of DNS packets but only the transport header information of other traffic to be classified. It observes all DNS replies to each client and, when there is a successful response (*i.e.,* NOERROR code) to a DNS query for an IP address (*i.e.,* type A or AAAA), TreeTop (i) stores the query name in a central *domain tree* (an n-ary prefix search tree), (ii) stores the IPv4 and/or IPv6 address answers in a client-specific *address tree* (a binary prefix search tree), and (iii) links nodes in the client's address tree to their corresponding nodes in the domain tree. Thus, these data structures store per-client *DNS rendezvous state information* as to which remote IP addresses are known by domain name. Subsequently, when TreeTop observes application traffic (*e.g.,* the wide-area traffic at a network's border router), it uses the rendezvous state information to label the client traffic as either "unnamed" or as "named," and accumulates per-client traffic counters (in bytes or packets) for those meta-categories as well as for hierarchical sub-categories by domain name.

To prepare the data sets for TreeTop, we synthesize pcap files from the flow data (with a modified flow-export utility [5, 95]) and merge them with the DNS pcap data (using mergecap) to form one coherent input data set. Note that, in general, it is sufficient for the DNS pcap records to be observed before the application traffic pcap records (from the flow data); so, for off-line studies, we can perform a single batch analysis for an entire day using TreeTop by first reading all DNS traffic data then the application traffic data. By contrast, performing an online analysis (at one observation point) obviates the need to carefully interleave the DNS and target traffic records based on their packet arrival times because the DNS responses are interspersed in the trace with the target traffic (to be measured) and would be observed before the subsequent associated target traffic.

4.3.1 Traffic Labels

4.3.1.1 Direct Classes

Direct DNS rendezvous-based traffic classification involves at least two sorts of traffic classes. The first, are the "named" and "unnamed" traffic classes, which simply indicate whether a client end-host knows the traffic's remote IP address by a domain name as the result of a canonical "forward" DNS query to translate that name to an address. The second and more challenging traffic classes are the domain names themselves. To deal with the innumerable fully-qualified-domain-names (FQDNs) that may exist in the world-wide DNS, we treat them hierarchically. For instance, traffic involving the FQDN "www.example.com" is in the "com" class, the "example.com" class, and "www.example.com" class, and thus can be presented at a number of levels of granularity. One can imagine categorizing domain names by common owner (*e.g.*, "facebook.com" and "fbcdn.net"), similar purpose (*e.g.*, weather or sports content), or even application groups such as WWW, FTP, Streaming, etc. We leave such classification by policy or operator objectives for future work by using readily available references [7].

4.3.1.2 Indirect Classes

Our indirect DNS rendezvous classification utilizes host profiles that are defined by configurable sets of domain names. We defined three such profiles for P2P clients. The P2P profiles are: "Torrent" (BitTorrent client applications, directories, and trackers), "Talk" (Skype and Google Chat applications), and "Game" (Massively Multiplayer Online Games). For instance, a client end-host that issued a DNS query trailing with "bittorrent.com" or "utorrent.com" will be profiled as a "Torrent" client because of its ostensible interest in a popular BitTorrent client application. Likewise, a client end-host that issued a DNS query trailing with "thepiratebay.org" will be profiled as a "Torrent" client because of its interest in this popular BitTorrent tracker site. These host profiles then, are used to label traffic classes. For example, the "Torrent" label would be used for traffic exchanged by a host having only the "Torrent" client profile; the "Talk+Game" label would be used for traffic involving a host having both the "Talk" and "Game" (but not "Torrent") profile. Note that we do not claim that "Torrent"-labeled traffic is necessarily BitTorrent traffic; instead, we claim that it certainly involves an end-host that matched the Torrent profile and is thus (at

least indirectly) associated with this P2P application. Each profile is defined by a set of domains that were assembled from readily available references [7, 13]. The Torrent domains (31) are popular BitTorrent Clients from Wikipedia and from Alexa's top "Torrent Directories and Tracker Domains." The Talk domains (2) are from observed behavior of the Skype application and Google Chat. The Game domains (35) consist of a well-known online game domain and Alexa's top "Massive Multiplayer Online Domains."

4.3.1.3 Port-based Classes

In addition to our rendezvous-based labels, we use traditional port-based application labels from an existing classifier [4] that has been used in prior work. [63] These are: "WWW," "P2P," "FTP," "Net. oper.," "Mail/News," "Streaming," "Encryption," "Games" (distinct from "Game" which is an indirect host profile-based label), "Chat," "Login," "Tunnels," "Other," and "UNKNOWN." While many services can be uniquely identified solely by that service's FQDN, port-based classes offer the advantage of familiarity and of distinguishing amongst multiple services that happen to be identified by a single (unspecific) FQDN.

4.3.1.4 Classification Order

In this study, we take a pragmatic approach based on flexible classification that emphasizes the complementary strengths of each method. First, we label the traffic with the direct DNS rendezvous-based classes (named and unnamed). Next, for results that involve port-based classification, we label the traffic using port-based classes, nested within "named." Finally, we label unnamed traffic with indirect labels based on profiling hosts by DNS rendezvous. This initial choice of order, we argue, is from the least to most speculative. In the general case, a complementary method, such as port-based, could be performed at any point; other orderings provide opportunities to explore how the rendezvous classes overlap with other classification schemes.

4.4 Results

In this section we report the results of a "day in the life" of an office and a residential user population, in terms of their DNS and wide-area application traffic.

4.4.1 DNS Traffic Analysis

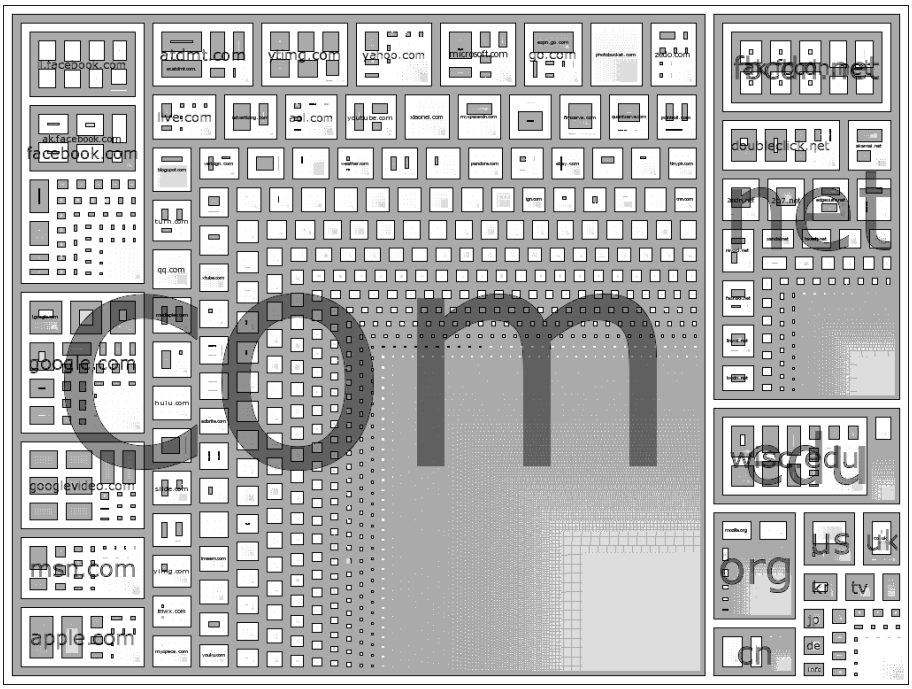


Figure 4.1: A treemap of domain popularity for all domain names queried that were answered with IP addresses during one day. This treemap for the residential population represents 142,594 unique FQDNs. The relative size of the rectangles indicate the domain names' relative popularity based on the number of IP address answers that a client knows as being associated with that name. The more clients that knew IP addresses associated with a given domain name, the more prominently it is shown.

Figures 4.1 is a treemap of domain names based on their popularity for the residential population. (The treemap for the office population was visually similar and, thus, not shown here.) For the day under study, the residential population resolved roughly 7 times more unique FQDNs than the office population (142.6K vs. 19.4K) in DNS queries from about 9 times as many client

end-hosts (5,583 residential vs. 614 office clients). From values in Table 4.1 we can see that there are roughly 32 and 26 unique FQDNs per office and residential client end-host, respectively, on this day. Many of the most popular domains are common between the office and residential populations, including "google.com", "facebook.com" and the associated Content Distribution Network (CDN) "fbcdn.net", "yahoo.com", "apple.com", "microsoft.com" or "msn.com", and the local campus' domain. The least popular domains, such as those that only a single host might know, are minuscule in the treemaps, and thus form the light gray fields in the lower right of the rectangles.

To further explore the popularity of FQDNs amongst these populations, Figure 4.2 shows the unique FQDNs known, ordered by popularity, *i.e.,* the FQDN numbered 1 on the horizontal axis is the most popular and that numbered 10 is the tenth most popular within the given user population. This figure clearly shows that most FQDNs are known by only a small percentage of the population. Specifically, only those FQDNs in the top 1000 are known by more than 5% of the hosts. The raw data from TreeTop shows that more than 68% of the FQDNs were known to only a single client end-host during this day. This underscores the need to aggregate the numerous FQDNs in some fashion, and here we do so hierarchically, beginning with TLDs, then second-level domains, and so on.

We also examined the distribution of clients based on the number of unique remote IP address answers known by domain name to the client. For these data sets, we find that 95% of the office and residential hosts learned (via DNS answer replies) of fewer than 1000 unique remote IP addresses by a domain name, and that more than 99% of all the hosts learned fewer than 2000 unique remote IP addresses by domain name throughout the entire course of this day.

In Section 4.4.2.4, we report the results of using these P2P profiles to classify traffic that doesn't directly involve the DNS for rendezvous. Appendix A lists the specific domain names associated with each of our host profiles.

4.4.2 Traffic Classification Results

Because our DNS domain name-based classification approach uses drastically different labels than prior classification work, we do not have a straightforward means of comparing performance. However, because our direct DNS rendezvous approach classifies based on domain names and IP address answers observed in each client's DNS traffic, it can be considered tacit ground truth.

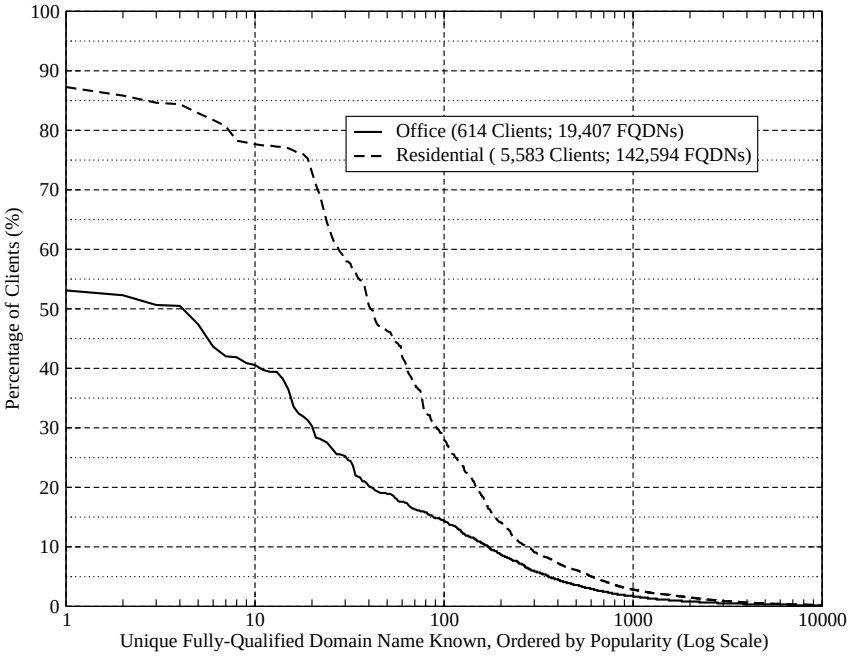


Figure 4.2: Popularity of FQDNs by client end-hosts during one day. Here we see that the most popular 10 and 100 FQDNs are known in common to more than 50% and 10% of clients, respectively. Note that the popularity ranking for office and residential populations were determined independently, thus it is unlikely that they share the same FQDN at a given rank.

That is, we are certain that the client end-host had the opportunity to know the remote IP address by that name. However, we are guided by finding 1 of Kim *et al.,* [63]:

> [The] port-based approach still accurately identifies most legacy applications [...] this suggests that ports still possess significant discriminative power in classifying certain types of traffic.

Our DNS rendezvous-based approach and a port-based approach are similar in that both of them label traffic based solely on easily-observed traffic elements, instead of labeling using heuristics and tunable thresholds.

4.4.2.1 Port-based Classification

We first classify the inbound and outbound traffic for our two populations using a port-based approach to set a baseline for comparison. Specifically, traffic identified by well-know ports is labeled either as one of 12 pre-defined application groups or UNKNOWN.

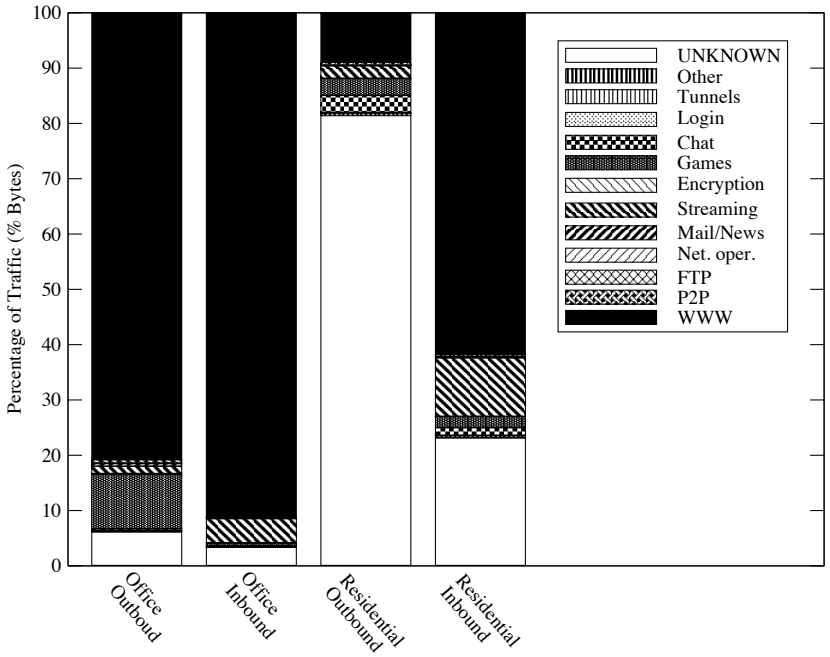


Figure 4.3: Port-based classification of traffic (bytes) for the office and residential populations during one day. While 93.9% (outbound) and 96.6% (inbound) of the office traffic is labeled (*i.e.*, not UNKNOWN), only 18.6% (outbound) and 76.9% (inbound) of the residential traffic is labeled due to the different application traffic mixes. Note the coarse labeling as only the WWW, Games, and Streaming applications represent 10% or more of the traffic by volume.

Figure 4.3 shows the classification of traffic using the simple port-based method. [2] Here we see a stark difference between the office and residential traffic; most of the office traffic is classified, but much less of the residential traffic is classified. Furthermore, the application mix differs greatly in these

[2]Our application group classes are those identified by CoralReef [4], specifically, coral-3.8.4, and thus are equivalent to those used in the work of Kim *et al.* [63].

two populations, with over 80% of the office traffic being labeled as WWW and only about 5% unknown, whereas less that 10% of the outbound residential traffic is labeled WWW, and more than 80% being left unidentified.

4.4.2.2 Direct Rendezvous-based Classification

We now classify the same office and residential traffic by our direct DNS rendezvous-based approach using labels as described in Section 4.3.1.1. We consider two broad rendezvous-based classes, "named" and "unnamed," then detailed sub-classes by domain name.

Figure 4.4 shows the time series volume for the named and unnamed portions of the office and residential traffic throughout the day under study. We see that nearly all office traffic involves DNS rendezvous and can be named. While a significant amount of inbound residential traffic can also be named, 32.1% (inbound) and 93.3% (outbound) is unnamed and, therefore, apparently does not employ the DNS for rendezvous. Also, note the correspondence between the portion of named traffic identified here by our method and that labeled by the port-based method shown in Figure 4.3; this suggests that DNS-named traffic very often uses well-known ports, *e.g.,* traditional client-server applications.

Figure 4.6 shows treemaps of the office and residential traffic. The unnamed traffic, to the left in each of the maps, is the traffic from remote peer source IP addresses that the local client end-host did not know by DNS name, *i.e.,* traffic for which the local end-host did not arrange via DNS rendezvous. The named traffic, to the right in each of the maps, is sub-classified based on the traffic volume from source IP addresses associated with each specific domain name known by destination client end-host. In Figure 4.6a note that about 90% of the office traffic (by packet volume) is named. The following domains are amongst the most prominent sources of traffic: "msn.com", "google.com" and "googlevideo.com", "facebook.com" and the associated "fbcdn.net". In Figure 4.6b, we see that only about half of the residential traffic is named. Since we are confident that the significant amount of unnamed residential traffic had no DNS preamble, thus it is likely to have been arranged by either a P2P rendezvous mechanism, an application-specific rendezvous mechanism, or an out-of-band rendezvous technique.

Figure 4.5 shows the time series residential traffic volume detailed by specific domain names, specifically the five domains involving the highest traffic volume. (This corresponds to the residential named traffic shown in Figure 4.4b.) Of

the named residential inbound traffic, *i.e.,* from source IP addresses that the clients know by domain name, the following are amongst the most significant: "facebook.com", "googlevideo.com", and "edgefcs.net". The prominence of this last domain led us to discover that the majority of traffic that is named by our method yet UNKNOWN to the port-based method is associated with the "edgefcs.net" domain; we discuss this in detail in Section 4.4.2.3.

4.4.2.3 Hybrid Classification

We demonstrate how we can improve port-based classification when it is performed in combination with our rendezvous-based method.

First, consider Figure 4.7a which shows the port-based application groups overlaid within the DNS rendezvous-based classes, named and unnamed. In the lower right, a significant portion (6.9%) "named UNKNOWN" traffic is shown. This means our DNS rendezvous method classified that traffic as being associated with remote IP addresses that the residential clients knew by one or more domain names. However, the port-based method did not identify this traffic as belonging to a known application group, hence it was classified as "UNKNOWN."

Now consider Figure 4.7b, in which we drill down on the "named UNKNOWN" traffic to examine this class by domain name. We see clearly that the majority of it is associated with the "net" TLD, and furthermore nearly all of that with the second-level domain "edgefcs.net". This domain hosts streaming content (presumably on Macromedia Flash Communication Servers, hence the name "fcs") atop the Akamai CDN. These servers deliver content by the proprietary Real Time Messaging Protocol (RTMP, port 1935) or by tunneling via HTTP (port 80) and HTTPS (port 443). Now, informed by our rendezvous-based approach, the formerly "named UNKNOWN" traffic can be labeled as Streaming, leaving only 0.2% of the total traffic as "named UNKNOWN." (Figures 4.7a and 4.7b are captioned with "Before" to indicate they were prepared before this Streaming traffic was identified by port, as show in Figure 4.7a.)

This example illustrates the power of DNS rendezvous-based classification in three ways: (i) traffic for a popular streaming protocol was shown prominently (as "edgefcs.net") even though the well-known ports database did not have an entry for this protocol, [3] (ii) the "forward" domain name by which

[3] We have provided an entry for the macromedia-fcs port to CAIDA to update the CoralReef application master.

the clients accessed this service was the only good indication of the traffic's application (since the content was hosted on a CDN, neither the service's reverse DNS names nor its IP addresses indicated the application), and (iii) the fact that this traffic could be named suggested that it is a traditional client-server application and thus should be able to be labeled by an improved port-based method.

4.4.2.4 Host Profiling and Classification Results

We now apply our indirect DNS rendezvous-based approach, using labels as described in Section 4.3.1.2.

As shown in Figure 4.4b, our direct DNS rendezvous-based classification method determined that only 6.7% of the outbound residential traffic was named, and, in Figure 4.3, we see the majority of this traffic is UNKNOWN by port. We expect this unnamed traffic might be dominated by P2P file transfer (*e.g.,* BitTorrent), game, and/or talk (*e.g.,* VoIP) traffic, *i.e.,* those groups of applications that do not typically use the DNS for rendezvous and also often use unreserved (not well-known) port numbers.

To classify this traffic, we employ the DNS rendezvous information *indirectly* by labeling local hosts according to P2P client *profiles* based on their DNS rendezvous activity. The resulting assignments are shown in Figure 4.8.

Then, in Figure 4.9, we correspondingly label portions of the unnamed residential outbound traffic (93.3% unnamed, as seen in Figure 4.4b). That is, when traffic is classified as "unnamed," we determine if that traffic involved one of the 1,252 P2P profiled residential hosts, and if so, we label that portion of the traffic by the given host's P2P profile name: "Torrent," "Talk," and/or "Game." For instance, clients running the Skype application are known to resolve "ui.skype.com", thus this is one of the domain names that causes it to fit the "Talk" P2P profile. While somewhat speculative, DNS rendezvous profiles are flexible and configurable; we find that our initial effort attributes 82.3% of the otherwise unlabeled traffic to the 22.4% of the hosts that fit a P2P profile, indicating the traffic was sourced from hosts that had resolved popular Torrent, Talk, or Game-related DNS domain names.

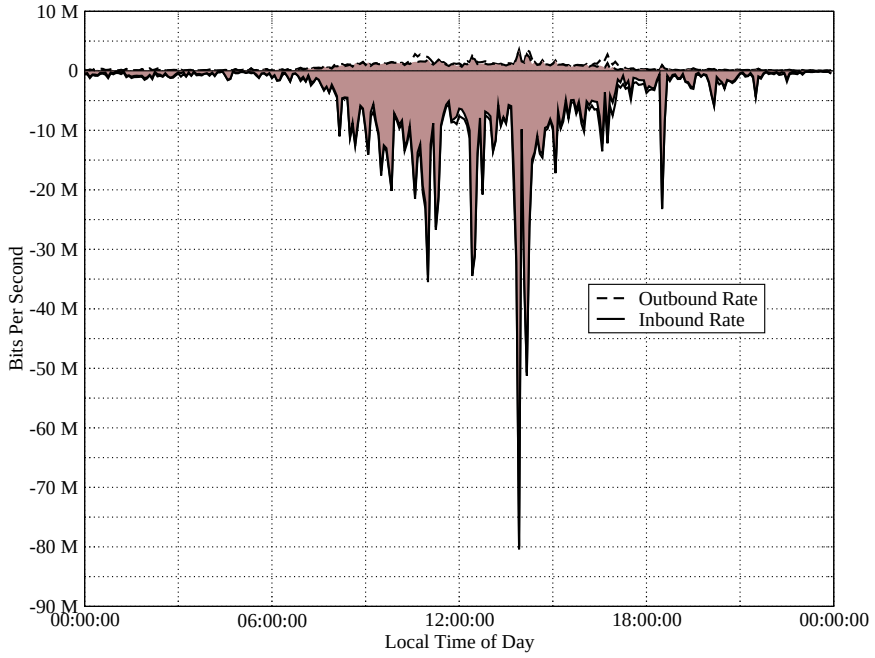
4.4.2.5 Results Summary

Table 4.2 summarizes the overall classification performance of the port-based method and ours.

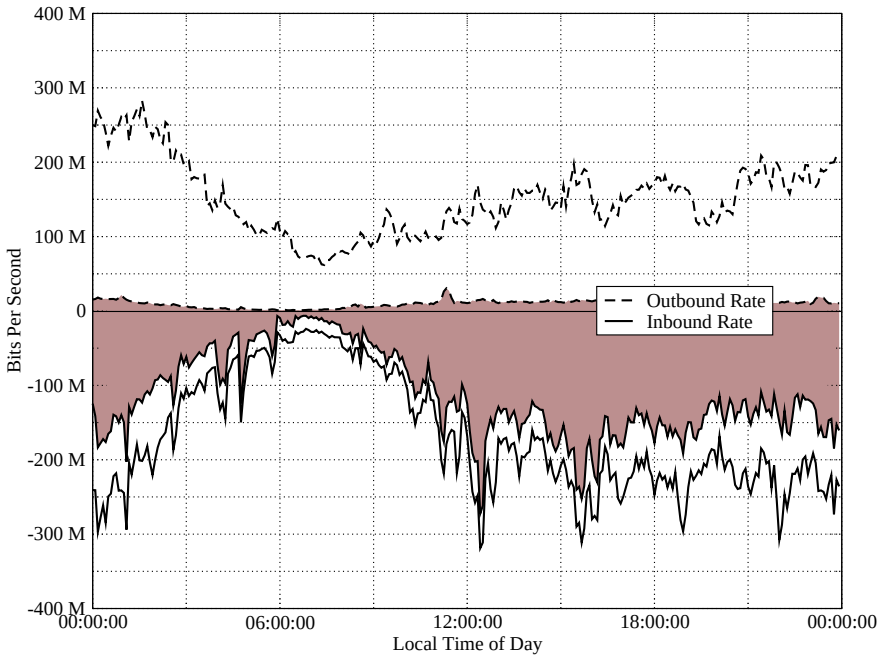
Data Set	Port-known	DNS-named and Port-known	DNS-named	DNS-named and DNS-Profiled
Office Out	93.9%	80.5%	81.8%	91.9%
Office In	96.6%	91.8%	93.2%	95.4%
Residential Out	18.6%	6.2%	6.7%	83.5%
Residential In	76.9%	58.3%	67.9%	88.2%

Table 4.2: Traffic classified (bytes) by each method: Port-known (by the port-based method), DNS-named (DNS rendezvous named), DNS-named and DNS-Profiled (DNS rendezvous named plus unnamed matching a P2P host profile).

The significant proportion of "DNS-named" traffic that also has "Port-known" for the office traffic (98%) suggests that one can be somewhat confident in the port-based method there. The lesser proportion for the residential traffic (86% outbound, 93% inbound) suggests that port-based result is suspect given that traffic mix. Lastly, for residential outbound traffic, we realize a 64.9% increase in volume classified by our DNS rendezvous method over the port-based method.



(a) Office



(b) Residential

Figure 4.4: Wide-area traffic rate, as observed at campus border during one day. Outbound rate (from campus) is plotted above the horizontal axis and the corresponding inbound rate (to campus) is plotted below. Clearly the office population is primarily a consumer of wide-area Internet content, whereas the residential population is both a significant consumer and provider of content. The portion of "named" traffic (*i.e.,* by DNS rendezvous) is shaded; while 81.1% (outbound) and 93.2% (inbound) of the office traffic is named, only 6.7% (outbound) and 67.9% (inbound) of the residential traffic is named.

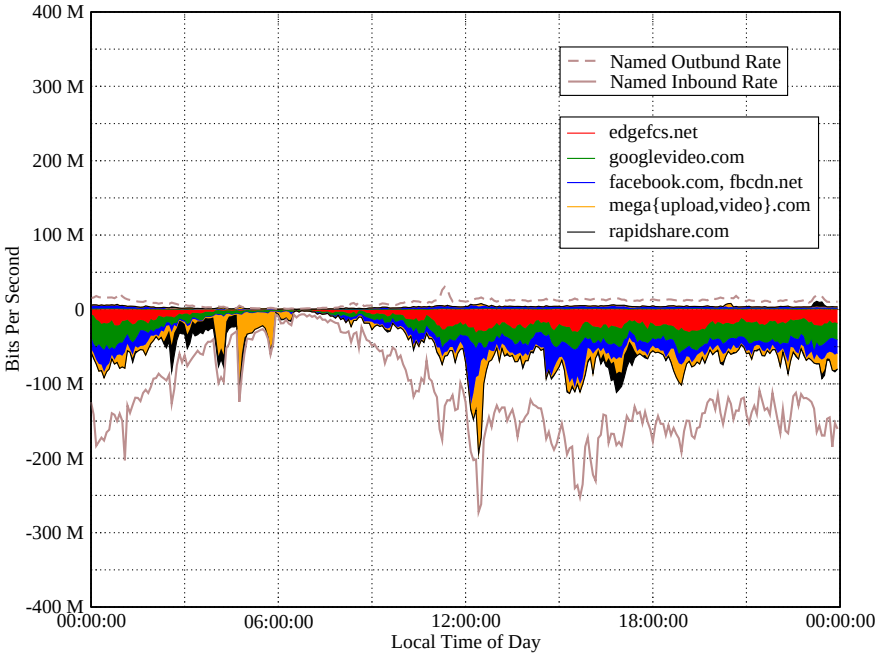
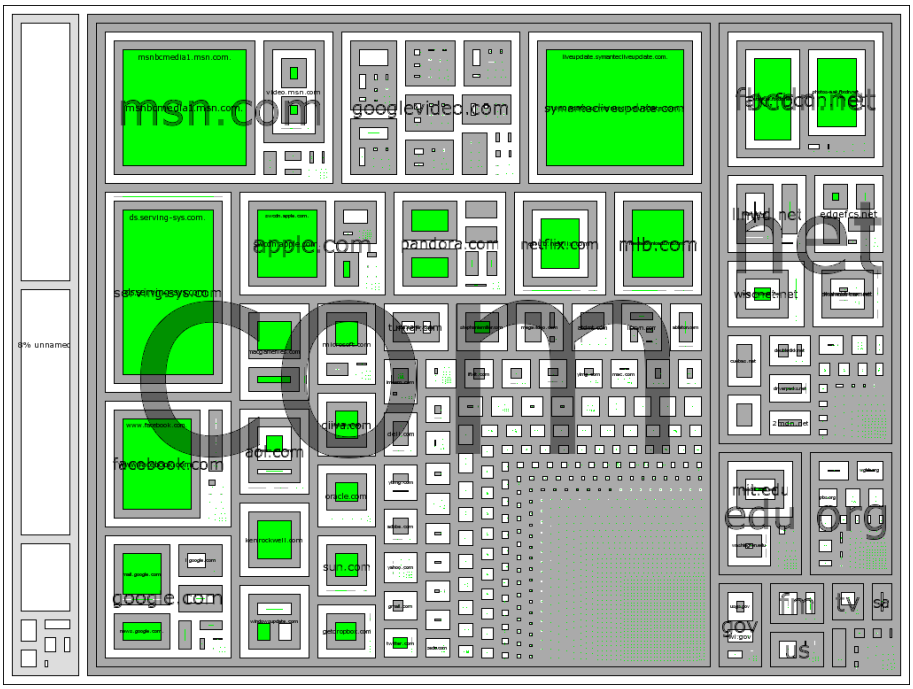
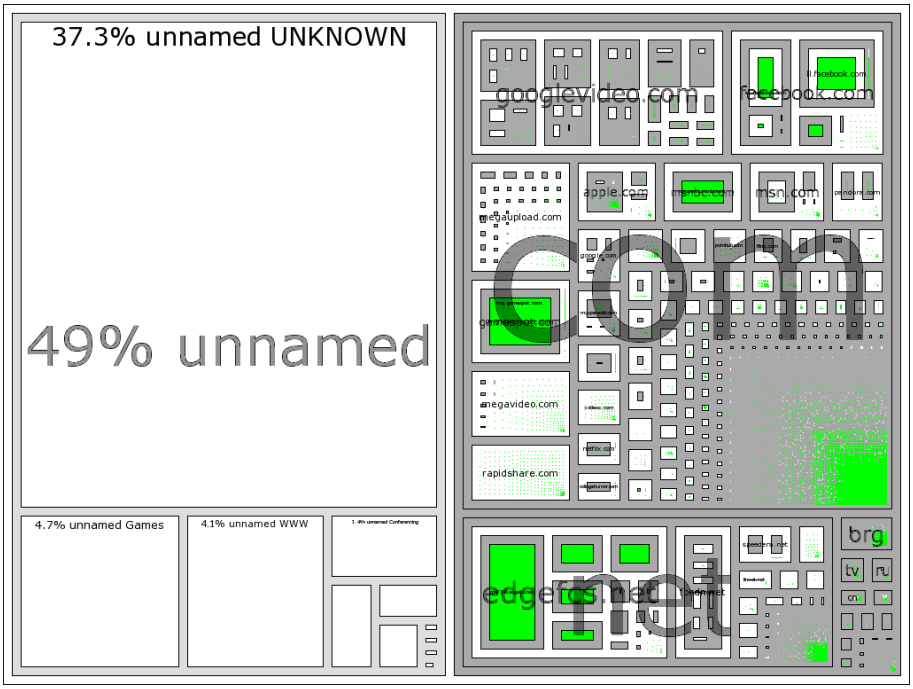


Figure 4.5: Wide-area traffic rate for the residential population, as observed at campus border during one day, labeled by domain. Outbound rate (from campus) is plotted above the horizontal axis and the corresponding inbound rate (to campus) is plotted below. The portion of "named" traffic (*i.e.,* by DNS rendezvous) is labeled by top domains, showing that merely five domains identify approximately half of the inbound residential traffic.

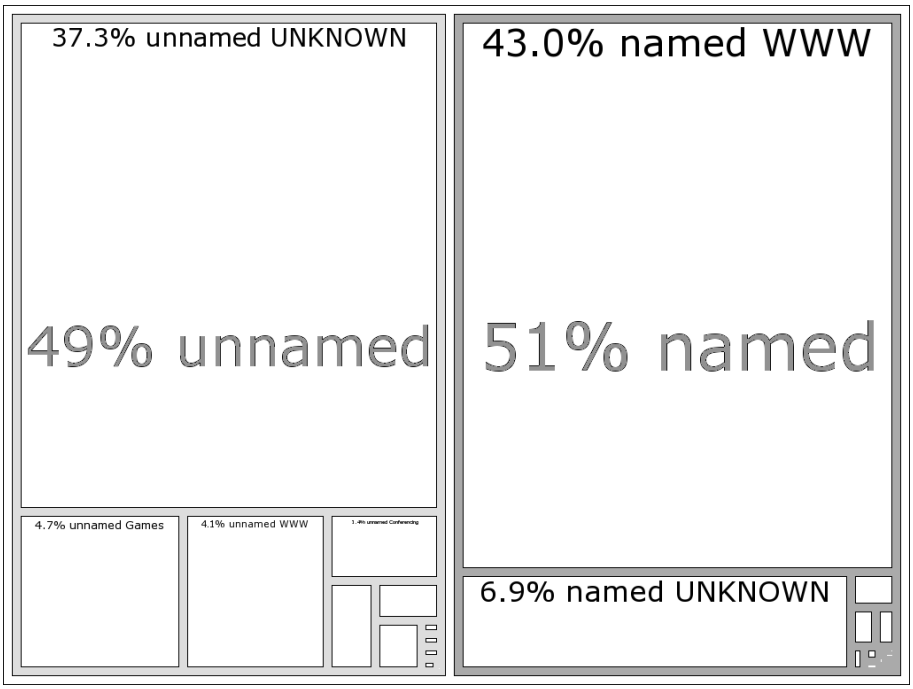


(a) Office

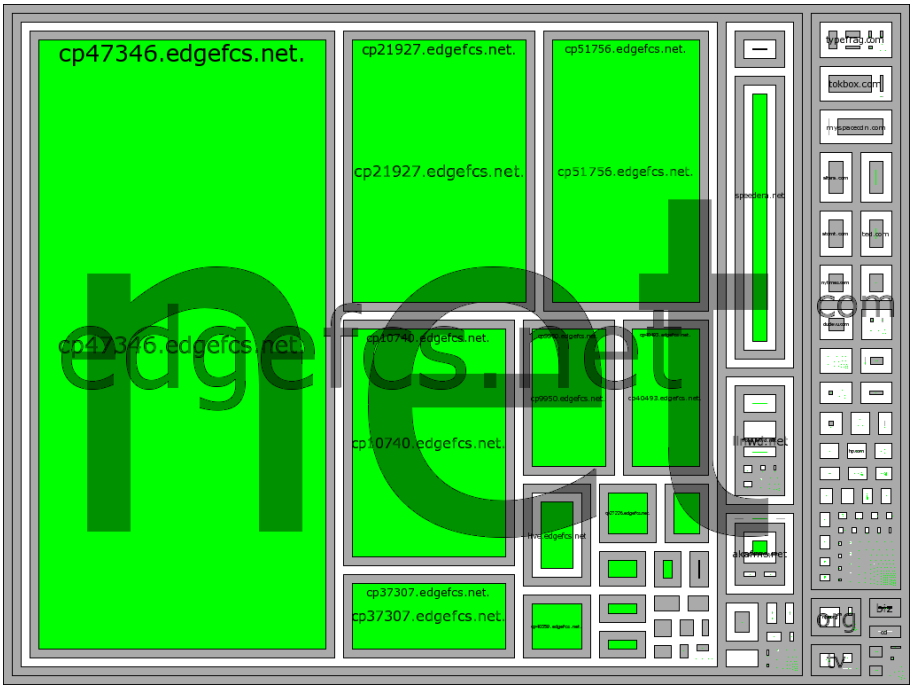


(b) Residential

Figure 4.6: Treemaps of inbound traffic (packets) from domain names. The relative sizes of the rectangles indicate the proportion of traffic (packets) from the given domain name. That is, the more traffic from that domain, inbound to the clients, the larger its rectangle. Here we see clearly that much more of the office traffic is arranged by DNS rendezvous than the residential traffic.



(a) Residential Inbound Before



(b) Residential "named UNKNOWN" Before

Figure 4.7: Treemaps of residential inbound traffic (packets) and residential "named UNKNOWN" subset of that traffic (bottom). Note there is a significant amount of "named UNKNOWN" traffic in Figure 4.7a that we show detailed by domain names in Figure 4.7b; the majority of this traffic was associated with the domain "edgefcs.net" - a domain that hosts streaming content atop a content distribution network. Thus, we can now label that content more appropriately as Streaming traffic.

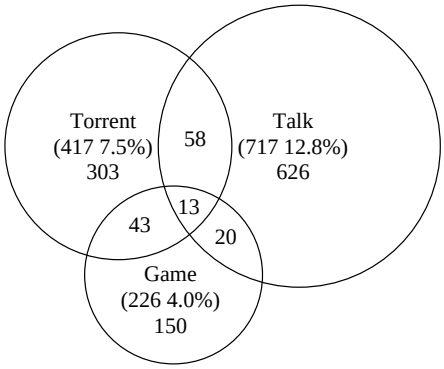


Figure 4.8: Residential subpopulation host counts by P2P application type based on their DNS queries during one day. Here we see that 1,252 hosts (22.4% of 5,583 total) appear to run one or more P2P applications. (Parenthesized values are totals for that subpopulation's circle.)

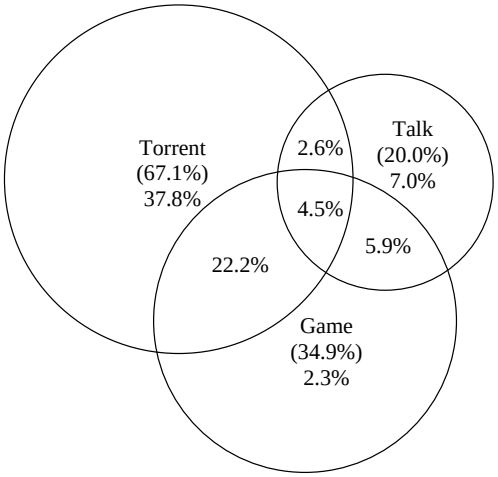


Figure 4.9: Residential unnamed outbound traffic volume (bytes) by P2P client profile. Here we see that 67.1% of this unnamed outbound traffic involved local hosts that were profiled as BitTorrent clients based on their DNS rendezvous activity. (Parenthesized values are totals for that subpopulation's circle.)

> The fresh angle of looking at the performance when using IPv6 vs. IPv4 is a strength, and the measurement methodology seems to be a partially useful contribution.
>
> anonymous reviewer, 2013

In this chapter, we report on a novel application of rendezvous-based traffic classification. Having realized that the only link between a given service operating on both IPv6 and IPv4 is the rendezvous mechanism, we present a passive measurement technique to compare performance of these services on both protocol versions during the IPv6 deployment and simultaneous operation with IPv4. [87]

5.1 Overview

With the exhaustion of the IPv4 address space, the deployment and operation of Internet Protocol version 6 (IPv6) in production networks is upon us. Indeed, years have passed since World IPv6 Day, June 8, 2011, when numerous Internet service providers demonstrated their readiness by providing their services over IPv6, and many continue to do so. There are myriad reports of significant amounts of IPv6 traffic being transited [30], suggesting that more and more users and services are utilizing IPv6.

However, achieving good end-to-end performance over IPv6 is challenging for a number of reasons. First, IPv6 must be deployed and operated in parallel with the existing IPv4 infrastructure. Second, new network configuration tasks must be introduced and performed. Also, network monitoring and performance assessment is more complicated with IPv6 because there are now populations of users with differing environments due to multiple IP versions; many users have IPv4, some might have only IPv6, and an increasing number have both, *i.e.,* "dual-stack" hosts. Such challenges motivate the need for general methods to assess IPv6 performance during deployment and during the long-lived simultaneous operation of both protocol versions.

In this chapter we present a new method for assessing the performance of Internet services over IPv6. Our objective is to (i) accurately assess performance based on passive measurements, (ii) provide the capability to compare and contrast IPv6 performance with that of IPv4, and (iii) provide an assessment that is both independent of the end-hosts and the Internet services they access. We perform this assessment based on passive measurements gathered at two observation points: one at or near the clients' recursive Domain Name System (DNS) resolver and the other at any point along the end-to-end path. Our approach does not need privileged knowledge of the Internet services nor special access to the end-hosts involved in the exchange of traffic. We then develop a framework and tools to detect and inspect performance differences between IPv6 and IPv4 for Internet services.

Our performance assessment method is predicated on the fact that client hosts, on IPv4 and IPv6, necessarily employ some *rendezvous* mechanism to discover the IP address(es) of an Internet service before interacting with it. For many types of port-based services (*e.g.*, HTTP, HTTPS, SMTP, and IMAP) and for most of those with early IPv6 support, the DNS is that rendezvous mechanism. For instance, users' client hosts rendezvous with Facebook, an Internet service, by resolving the domain name "www.facebook.com." When a service supports IPv4 and IPv6 simultaneously, clients use a common rendezvous mechanism for both Internet protocol versions, such as the DNS.

Our framework determines service performance in three steps: (i) **measurement** in two forms: *(a)* full capture of low-volume DNS query/response packets and *(b)* collection of 5-tuple IP flow export records with duration and byte count of high-volume application traffic; (ii) **classification** of flows source and/or destination IP addresses by matching them to their corresponding domain names (when possible) based on query names and the resulting IP addresses in DNS responses; (iii) **performance inference** by constructing a distribution of flow bit rates and applying statistical techniques.

The common DNS rendezvous mechanism is an aid in the transition to IPv6 because it keeps the user from having to make the difficult decision about whether to use IPv4 or IPv6. Modern dual-stack Internet hosts issue DNS queries to request IPv6 and/or an IPv4 address(es) for the domain name of the desired service. When a given service supports both IPv4 and IPv6, the client host, having retrieved one type of address or both, makes a decision as to which peer address to use; this decision is typically performed in the IP implementation, resolver, or application. For example, a dual-stack client wishing to access

a World-Wide Web (WWW) service named "www.example.com" might issue a AAAA (or "quad-A") query for that service's IPv6 address: 2001:0db8::2:1. It might also, simultaneously or subsequently, issue an A address query for that service's IPv4 address: 192.0.2.1. One or the other of these peer addresses will be selected, possibly influencing the resulting performance and user experience.

We demonstrate the capabilities of our method by collecting traffic trace data for a campus population having dual-stack client hosts. This trace data, collected on World IPv6 Day, consists of 24 hours of: (i) all recursive DNS queries issued by those client hosts and (ii) complete flow export records (from NetFlow configured without sampling, *i.e.,* non-packet-sampled) for those client hosts' traffic as it traverses a campus core router.

During the course of this work we encountered a number of challenges. We desire a common stream and storage format that can encapsulate rendezvous information (*i.e.,* DNS queries and responses), packet capture, and flow export information, for online and offline processing. Furthermore, we require high-performance C data structures and other APIs available in a scripting language for rapid prototyping and ad hoc analysis and reporting. Another challenge arose from the complication inherent in dual-stack hosts: namely, they have multiple IP address "identities" and it is non-trivial to determine by passive observation whether or not a given IPv4 and IPv6 address are actually bound to the same host. This is pertinent because hosts often use one IP address for their DNS queries and responses, and the other to interact with other remote hosts; by observation, it can seem as if the host has a covert channel by which it gleans rendezvous information. We addressed this by developing a consensus-based strategy to infer IPv6 rendezvous information. Other challenges arose from the way in which Internet operators are deploying the IPv6 portion of their services. For instance, we found that many domain names were multiplexed to one IPv6 address, sometimes for seemingly unrelated services. This practice, sometimes called "virtual hosting," was popularized in the IPv4 Internet, in part, because of address scarcity. This suggests IPv6 service architects and operators mimic techniques used with IPv4 that may not be necessary nor be ideal with IPv6.

Our results expose various performance characteristics of Internet services that support IPv6: (i) Robust measures of services' flow bit rate distributions vary significantly by time of day, numbers of active local clients, and by IP protocol version (6 or 4). (ii) These rate characteristics differ amongst services. (iii) There are regimes of time in which IPv6 flow bit rates exceed those of IPv4 and others where the IPv4 rates exceed those of IPv6.

5.2 Method and Implementation

Our goal is to develop a performance assessment framework with the following characteristics or features:

- a method employing rendezvous-based traffic classification and robust statistics to determine and expose IPv6 and IPv4 performance phenomena for Internet services.
- an extensible mechanism for encapsulating the requisite rendezvous and traffic trace data prior to classification and for annotating IP traffic trace data afterward.
- a mechanism for transmitting streams of the encapsulated data to distributed framework components for online analysis in near real time.
- a serialized data file format for storing the encapsulated and annotated data for offline analysis, as in this study.
- a scripting language interface for the high-performance C code data structures and APIs that can be used to build distributed framework components and to conveniently run ad hoc analyses and reports.

We utilize this framework to assess the performance of traffic exchanged between hundreds of IPv6-capable campus hosts and Internet services with which these hosts rendezvous via the DNS. To this end, we've reimplemented TreeTop (previously a standalone tool) by incorporating the features above as the *TreeTop Framework*. The framework is shown in Appendix B. TreeTop now processes "nmsg" streams rather than pcap input. The nmsg format [15] is an extensible encapsulation scheme, based on Protocol Buffers [53], that provides both transmission (for online analysis) and serialization to a data file (for storage and offline analysis.) Here, we first give a brief overview and highlight our TreeTop framework's features.

TreeTop processes incoming streams of DNS and flow export data for application traffic. The streams are network messages (nmsg) of two types: (i) DNS query and response (dnsqr) and (ii) NetFlow data (nfdump). The dnsqr messages contain all the interesting parts of a DNS query and response plus the time observed and delay between the corresponding query and response. The nfdump message contains flow data with typical transport header information and other attributes of unidirectional 5-tuple IPv4 and IPv6 flows. We

introduced the nfdump message type to ISC's nmsg framework and based it upon the records from the nfdump tool [14]. A sample nfdump message is shown in Figure 5.1.

In an input nmsg stream, TreeTop observes the dnsqr messages that contain DNS reply information to each client; when there is a successful response to a DNS query for an IP address, TreeTop (i) stores the query name in a central *domain tree* (an n-ary prefix search tree), (ii) stores the IPv4 and/or IPv6 address answers in a client-specific *address tree* (a binary prefix search tree), and *(c)* links nodes in the client's address tree to their corresponding nodes in the domain tree. Thus, these data structures store per-client *DNS rendezvous state information* as to which remote IP addresses are known by domain name(s). Subsequently, when TreeTop observes application traffic flows (in nfdump messages in the input stream), it uses the prior rendezvous state information to annotate the nfdump message with source or destination domain names (corresponding to the source and destination addresses), and accumulates per-client traffic counters (in bytes or packets) for those meta-categories as well as for hierarchical sub-categories by domain name.

For the data sets in chapter, we collect a pcap trace of DNS traffic on the recursive name servers used by the campus population described in Section 5.3. We also collect NetFlow version 9 flow export data from a campus core router that forwards traffic between the population and the Internet beyond the campus border. To prepare the data sets for TreeTop, we convert the pcap traces of DNS using nmsgtool [15] and convert nfdump using a tool of our own (the nfdump2nmsg script). Subsequently, these sets of messages are merged by timestamp so that the DNS and flow information are properly interleaved to form one coherent input stream in which DNS query responses will be observed prior to their associated application traffic flow data.

To annotate flows with domain names (or domain suffixes) as classification labels, we utilize two rendezvous-based labeling methods: *direct* and *consensus*.

5.2.1 Direct Labeling

Direct DNS rendezvous-based labeling is performed when TreeTop discovers that a given client end-host knows a peer remote IP address by a domain name as the result of a canonical "forward" DNS query to translate that name to an address. In this case, an nfdump record can be annotated because the client involved has used the DNS to resolve the name of its peer; we call this

```
[2011-06-08 21:52:26.000000000] [7:1 WISC nfdump]
sa: 203.0.113.71
da: 192.0.2.32
sp: 80
dp: 55983
pr: 6
ibyt: 396630
td: 0.064000
snamed: CLIENT_DNS_NAMED
sn: static.ak.facebook.com
ip_version: IPV4
```

Figure 5.1: An nfdump message in presentation form, annotated with source domain name (`sn`).
The source name is known because the client (with the destination address in `da`) performed an A query that resulted in this peer source address (`sa`) as the answer.

"CLIENT_DNS_NAMED". For instance, the sample nfdump record in Figure 5.1 has been annotated with "static.ak.facebook.com", as shown in the `sn` (source name) field.

This direct labeling is the most reliable, but it requires the client host to use the same IP address as both the source of its DNS queries and as its local address when exchanging related application traffic. If that is not the case or if TreeTop does not observe a given client's DNS query response that contained a given peer IP address in an answer, we resort to "consensus" labeling.

5.2.2 Consensus Labeling

In our observations, the dual-stack hosts use just one of their IP addresses as the source of their DNS queries: a host's IPv4 address. Thus, for IPv6 flows in this chapter's work, we often can't perform direct labeling since the client host's IPv6 address is not usually the client address in the corresponding dnsqr messages. To label these flows' sources or destinations, we, instead, use a *consensus-based* approach based on the domain names resolved by other DNS clients in the population studied. If another host or hosts resolved a name to the peer address in question, it is generally agreed, by rough consensus of the population, that this host could also have used the DNS and named the peer (source or destination) similarly; we call this "INFERRED_DNS_NAMED". As can be seen in Figure 5.2, the `snamed` annotation has been set to `INFERRED_DNS_NAMED` meaning that we have determined the source name by consensus to be the value shown in the `sn` source name annotation, *i.e.*, "*.facebook.com". Note that this is not a fully-qualified domain name (FQDN), but rather a domain

suffix; this is because the consensus of the population was that multiple names resolve to the given source address. To improve the classification, it is useful to *sample* and present those names.

```
[2011-06-08 20:14:11.000000000] [7:1 WISC nfdump]
sa: 2001:0db8::face:b00c:0:3
da: 2001:0db8::2:1
sp: 443
dp: 53646
pr: 6
ibyt: 34297
td: 0.064000
snamed: INFERRED_DNS_NAMED
sn: *.facebook.com.
sn_sample: de-de.facebook.com.
sn_sample: check6.facebook.com.
sn_sample: ar-ar.facebook.com.
ip_version: IPV6
```

Figure 5.2: An IPv6 nfdump message annotated with a partially ambiguous source name (`sn`) determined by consensus and samples (`sn_sample`) of the resolved FQDNs that matched.
The source name is inferred because this client (with the destination address in `da`) was not observed to have resolved a name to this source address (`sa`), but other clients in the locally monitored population resolved at least three names to this source address.

5.2.3 Name Sampling

Whether direct or consensus labeling was performed, it is certainly possible that a given flow's source or destination may be known by more than one domain name. For instance, a service with the name "www.example.com" might also be known as "login.example.com". In such a case, we would like to annotate an nfdump record with a single name for the peer, such as "*.example.com", but also with what caused the ambiguous label. To expose this information, we perform "name sampling."

In Figures 5.2 and 5.3, note that the source name `sn` and destination name `dn`, respectively, do not contain an FQDN. Instead they contain ambiguous domain suffixes: "*.facebook.com" and "*." (the DNS root), respectively. This is somewhat unsatisfactory for traffic classification, especially when we only have the DNS root or merely a Top-Level Domain (TLD).

To deal with this situation, we sample and report a number (*e.g.*, 3) of the FQDNs that led to the ambiguity. Specifically, we search the resolved domain names to report a diverse set of FQDNs that differ in the DNS label where the ambiguity occurs, *i.e.*, where the "*" is in the aforementioned examples. Once these samples are gathered (and added as annotations to the nfdump

records) we can either (i) compose class labels from multiple domain names or (ii) consider whether or not the names are likely associated with one common service. In the former situation, for example, we might re-label an ambiguous "*.com" to "Gmail" if the samples were "mail.google.com" and "gmail.com"; in fact, this is exactly what we do for Internet services that have conflicting labels or TLDs, such as Gmail, in the results in the Section 5.4. In the latter case, for example in Figure 5.3, "rss.slashdot.org" and "www.beantownbloggery.com" seem to be unrelated; this confounding ambiguity results from (i) the hosting of unrelated services on the same IPv6 address and by (ii) using dual-stack hosts during the transition to IPv6.

```
[2011-06-08 00:11:10.000000000] [7:1 WISC nfdump]
sa: 2001:0db8::2:1
da: 2001:0db8:fff4::79
sp: 56451
dp: 80
pr: 6
ibyt: 849
td: 0.128000
dnamed: INFERRED_DNS_NAMED
dn: *.
dn_sample: rss.slashdot.org.
dn_sample: www.beantownbloggery.com.
ip_version: IPV6
```

Figure 5.3: An IPv6 nfdump message annotated with an ambiguous destination name (`dn`) determined by consensus and samples (`dn_sample`) of the resolved FQDNs that resolved to the same IP address.

5.2.4 Port-based Classification

To complement the aforementioned DNS rendezvous-based classifications, we employ traditional port-based application labels from an existing classifier [4] that has been used in prior work. [63] These are: "WWW," "P2P," "FTP," "Streaming," etc., and allow one to distinguish amongst multiple service types that happen to be identified by a single domain name or prefix, such as distinguishing IMAP from HTTPS traffic for Gmail.

The last part of our methodology to assess performance of IPv6 (and IPv4) flows is to calculate the bit rates based on fields already present in the nfdump records: `ibyt` (input bytes) and `td` (time, duration), which we do for all flows having a non-zero duration. We rely on the flow export implementation (in the commercial router) in that we assume sufficient granularity, range, and accuracy of these values for the distributions of rate values used in our performance analyses and results.

5.3 Empirical Data set

Since we are interested in assessing the performance of Internet services over IPv6 (as compared with IPv4), we select a campus population whose network and client hosts are IPv6-capable. On our campus, there are thousands of dual-stack hosts that reside within 22 IPv4 subnets and one IPv6 subnet and are mixed-use in campus offices and labs.

To gather the traffic traces and input data for this chapter's work, we monitor campus traffic at two observation points: (i) the campus clients' recursive name servers, and (ii) a campus core router that forwards traffic between the client hosts and the commodity Internet. We perform full packet capture at the campus domain name servers, and collect non-packet-sampled NetFlow version 9 data at a campus core router. Thus, the payload of the DNS traffic is recorded, but the application traffic payload is neither needed nor recorded. Such monitoring of DNS traffic between the client end-hosts and their recursive DNS service and router-based flow export is feasible within the typical networks of large institutions, enterprises, or Internet service providers. Our interest is in the "canonical" DNS traffic, *i.e.,* the standard DNS traffic expected to precede application traffic that consists of a query by FQDN and an answer containing one or more IP addresses associated with the query name. Because we assess performance using flow bit rates, we use non-packet-sampled flow export data that has complete byte and packet counts as well as start time and flow duration.

Both the DNS and flow export data were collected for the 24 hours of World IPv6 Day. We collected ~14.2M DNS query responses for 2028 total IPv4 and 23 IPv6 client addresses; of these, ~114,300 AAAA queries resulted in ~6,200 NOERROR responses. The client hosts' total traffic was represented as ~58.8 million IPv4 flows and ~2.4 million IPv6 flows. The number of active IPv6 and IPv4 client hosts numbered in the the hundreds and is shown in Figures 5.4a and 5.4b (Section 5.4).

5.4 Results

In this section we provide a sample assessment of the IPv6 and IPv4 performance for the World-Wide Web traffic (HTTP and HTTPS) involving two popular services, Facebook and Google Mail (Gmail), as observed during the 24 hours of World IPv6 Day (June 8, 2011). We selected these services due to the high number of active local client hosts that utilized them, thus providing a larger sample of hosts and their respective flows for each hour of the day. First we consider how the traffic was classified as being associated with each service and the differences by IP protocol version, then the active clients, and finally, the flow bit rate as distributions in time series with hourly bins.

5.4.1 Service Domain Names

As discussed in Section 5.2, we perform our analyses with scripts that process an nmsg stream of nfdump messages annotated with the domain names that client hosts resolved to the source and/or destination addresses of each flow. The traffic is labeled by domain name (FQDN or domain suffix) and that label is the basis for classification, *i.e.,* Facebook or Gmail.

The IPv4 Facebook traffic is that labeled with one of 950 FQDNs that have the suffix "facebook.com" (and were resolved by this population), such as "www.facebook.com", "developers.facebook.com", "ssl.facebook.com", "login.facebook.com", "upload.facebook.com", etc., including 867 different FQDNs matching "*.channel.facebook.com". The IPv6 Facebook traffic is that labeled with domains including: "www.facebook.com", "developers.facebook.com", "check6.facebook.com", and various others matching "*.facebook.com". This Facebook classification yielded ~618K IPv4 flows and ~128K IPv6 flows.

The IPv4 Gmail traffic is that labeled with the following domains: "gmail.com", "mail.google.com", and "www.gmail.com". The IPv6 Gmail traffic is that labeled with the following domains: "gmail.com", "mail.google.com", "www.gmail.google.com". This Gmail classification yielded ~785K IPv4 flows and ~463K IPv6 flows.

5.4.2 IPv4 and IPv6 Service Asymmetries

We observe that these services exhibit some asymmetry with respect to the specific DNS names resolved to access them over IPv4 versus IPv6. This is apparently due to differences in implementation of the IPv4 and IPv6 por-

tions of the service. For Facebook, we see that the FQDNs matching "*.channel.facebook.com" were not resolved by AAAA queries (but were resolved by A queries for IPv4 addresses), thus it's probable that Facebook Chat was not yet supported via IPv6 and may fall-back to IPv4 on a dual-stack host. Alternatively, it's possible that the Chat service via IPv6 was overloaded on another FQDN or that it used a non-DNS rendezvous mechanism, and thus may be structured differently (with respect to DNS names).

For Gmail, similarly, names such as "imap.gmail.com" and "smtp.gmail.com" were not resolved by AAAA queries; thus, we believe that these Google Mail features (IMAP and SMTP access) were not available via IPv6 at the time. To accommodate this in these results, and guided by finding 1 of Kim *et al.* [63] and our prior work, [86] we select only WWW traffic (by selecting flows with the port numbers for HTTP and HTTPS) so that IPv4 Gmail traffic involving IMAP and SMTP would not be mixed into the performance results for comparison (below).

Such asymmetries or differences in service implementation between IPv4 and IPv6 are a challenge to attempts to directly compare service performance between IPv4 and IPv6. The initial performance analysis presented here assumes that the IPv4 and IPv6 traffic classifications are equivalent for these two services, ignoring the Facebook Chat complication noted above.

5.4.3 Active Hosts

In Figures 5.4a and 5.4b we plot the total number of active local host IP addresses, IPv4 (solid line) and IPv6 (dashed line) for Facebook and Gmail, respectively. The horizontal axis above the plot in Figure 5.4a is labeled with the hour of day in local time, five hours west of UTC; the lowest level of activity is at about 0600 and the highest (for these services) during the noon hour, with activity decreasing toward the end of the work day (after 1700 hours). Also, note that there are two regimes: roughly the first 12 hours of World IPv6 Day have low activity and thus fewer flows for which we examine their bit rates; the latter 12 hours have high activity with many more hosts and flows being used to calculate bit rate distributions.

5.4.4 Flow Rates

Bit rate distributions for unidirectional flows were calculated for all non-zero duration flows, simply by dividing the number of bits by the flow duration (in

seconds). Bit rate is labeled on the vertical axis in Figures 5.5 and 5.6, and the horizontal axis is the hour of the day.

Figures 5.5a, 5.5b, 5.6a, 5.6b are box plots presenting the hourly outbound and inbound flow bit rate distributions for each service. The outbound rates (from local clients) are plotted above the horizontal access, and the inbound rates (to local clients) are plotted below as negative values. The boxes plot the 25th percentile, median, and 75th percentile, and the error bars plot the 1st percentile and the 99th percentile. For each hour bin, the IPv4 flow rates are darker with a solid line and the IPv6 rates are lighter with a dashed line.

Figure 5.5a shows the Facebook flow rates on a coarse scale, to highlight the error bars extending to the 1st and 99th percentile. Note that the 99th percentile flow in hour 21 (UTC) had a rate of nearly 50 megabits per second; this is the flow shown in Figure 5.1 (Section 5.2), where we see that it is very short-lived: ~0.064 seconds. In this plot we also see that the 99th percentile inbound flows from Facebook via IPv4 greatly exceed the rates via IPv6. Furthermore, inbound flow rates from Facebook generally peak higher than those outbound to Facebook; this might be expected for this service when most of its bulk data transfer is content pushed to the client host's web browser application.

Figure 5.5b shows the Gmail flow rates on a coarse scale. Here we see that the 75th to 99th percentile flow rates outbound and inbound are roughly equal, as are the 99th percentile flow rates for both IPv4 and IPv6; note, though, that there are hours of the day, *e.g.,* hours 1, 3, 5, 7, 9 UTC, when the 99th percentile IPv6 rate exceeded the IPv4 rate, suggesting that the IPv6 service, at least occasionally, provided similar throughput. We also see that there are similar 75th to 99th percentile flow rates outbound to Gmail and inbound from Gmail, suggesting that this bidirectional symmetry is characteristic of a WWW email service; both sending and receiving messages result in similar workload in HTTPS flows.

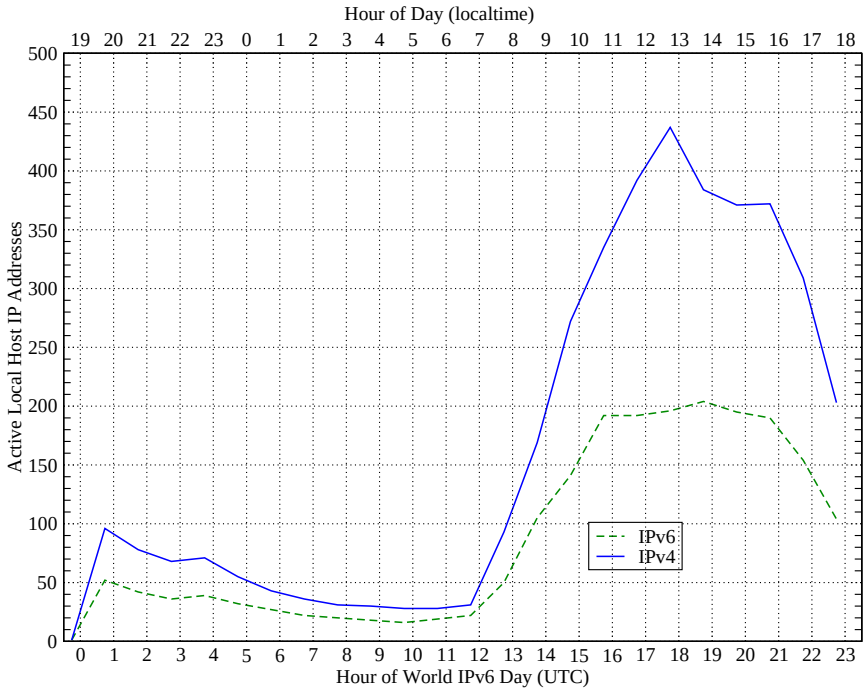
In Figures 5.6a and 5.6b we show a finer-detail representation, where the vertical axis has been clipped to highlight the interquartile range and median in the box plots of the flow rate distributions; these correspond to the distributions plotted in Figures 5.5a and Figure 5.5b, respectively. Figure 5.6a plots the Facebook flow rates lower than 200K bits per second. Here we see that the flow rate distributions vary with activity level and/or the number of flows. Such measurements, whether due to load or sample size, suggest a direction for subsequent forensic investigation.

Lastly, by examining Figure 5.6b, we see differing performance between

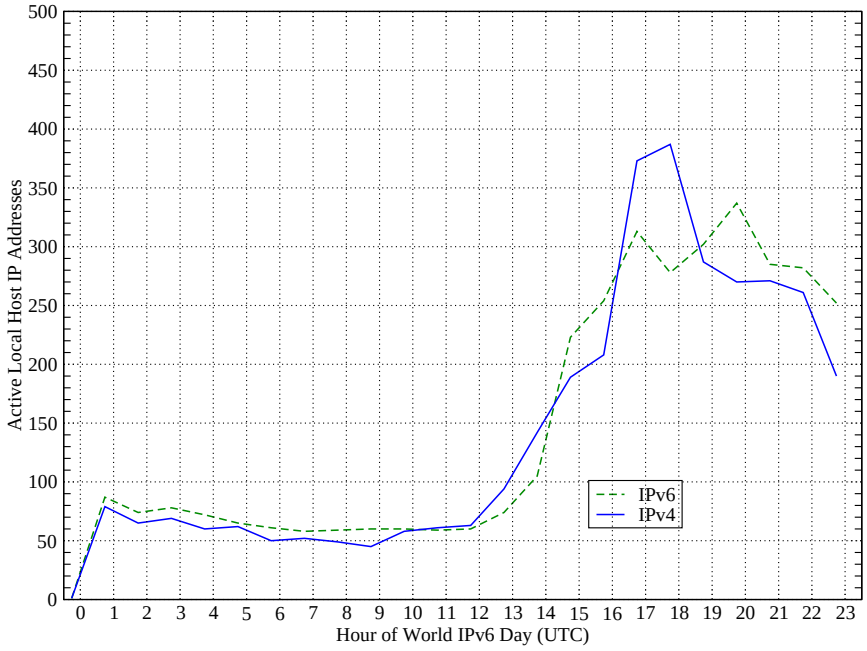
local nighttime and daytime. First, the interquartile range of IPv4 flow rate distributions exhibit higher bit rates than the corresponding interquartile range for IPv6 flows; note the number of active IPv4 and IPv6 hosts are nearly identical (Figure 5.4b). Second, in the high-activity local daytime, the IPv4 and IPv6 performance seem roughly comparable, until after 1700 hours UTC (noon local time), when the interquartile range for IPv6 flows contains consistently higher values than those for IPv4 flows. This change in IPv6 performance is not correlated merely with a change in the number of active hosts observed.

In these results we employed robust statistics to broadly compare IPv4 and IPv6 performance for two popular services. We find that the number of active hosts (observed via their flows) greatly influences the bit rate distributions. Second, we see evidence of wildly varying near peak (99th percentile) rates in flow export data for a given Internet service. Third, we see that there are regimes in which IPv6 rates are higher and others in which IPv4 rates are higher.

These results show that ostensibly the same services over IPv4 and IPv6 exhibit different performance as measured by the clients' sessions' flow bit rate distributions, meeting our objective to develop an analysis method and presentation by which one could expose performance phenomena and assess IPv6 performance. These observations motivate and guide future work including other visualizations and forensic tasks to determine the root causes of performance anomalies for services on both IPv4 and IPv6.

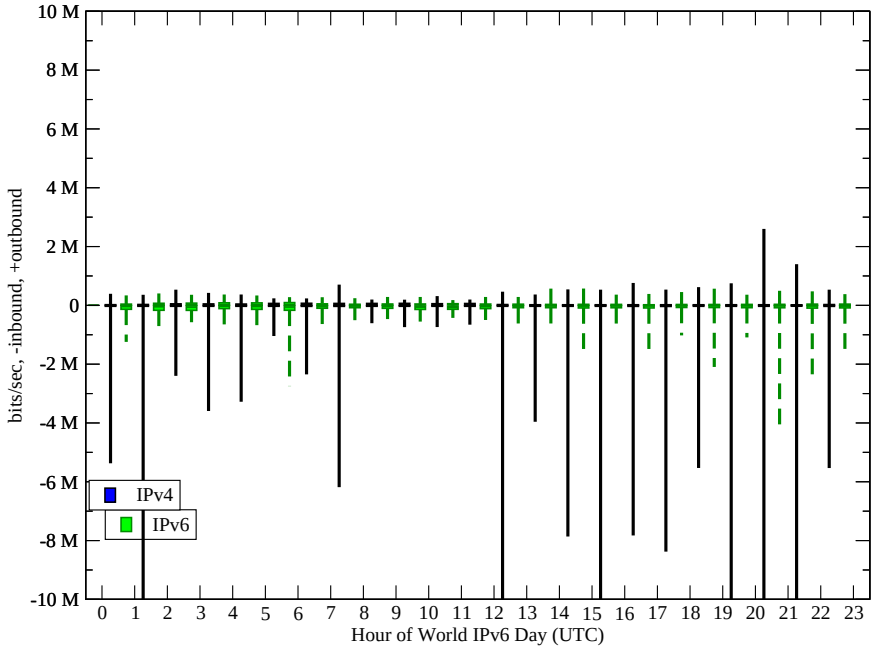


(a) Facebook Active Client IP Addresses

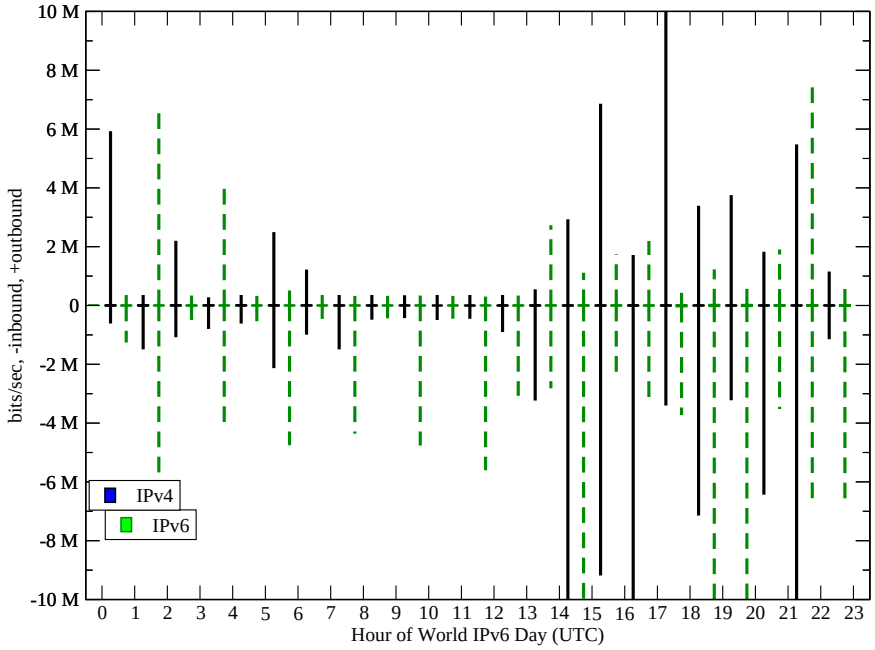


(b) Gmail Active Client IP Addresses

Figure 5.4: Active clients for Facebook and Gmail on World IPv6 Day.

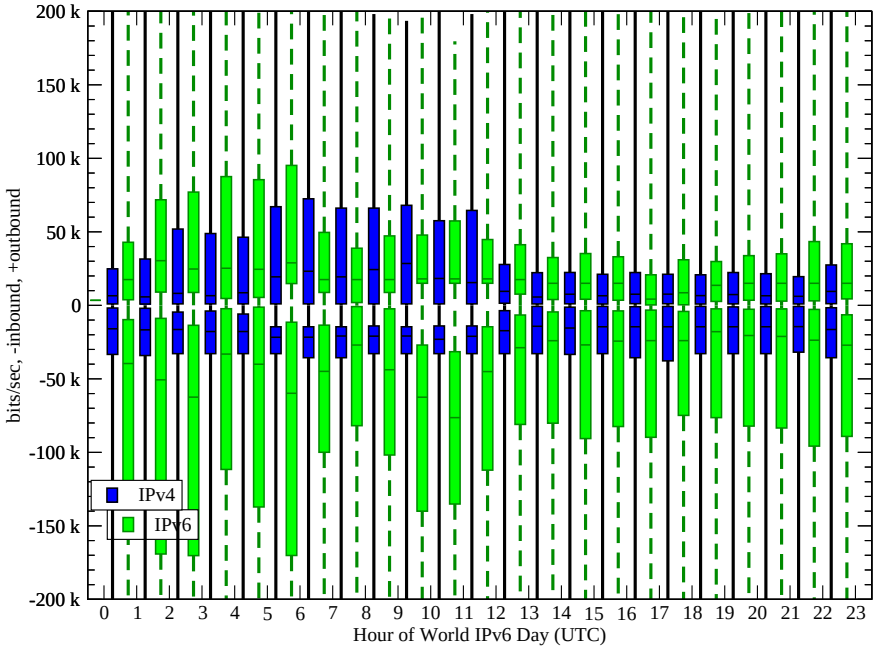


(a) Facebook WWW Flow Bit Rates

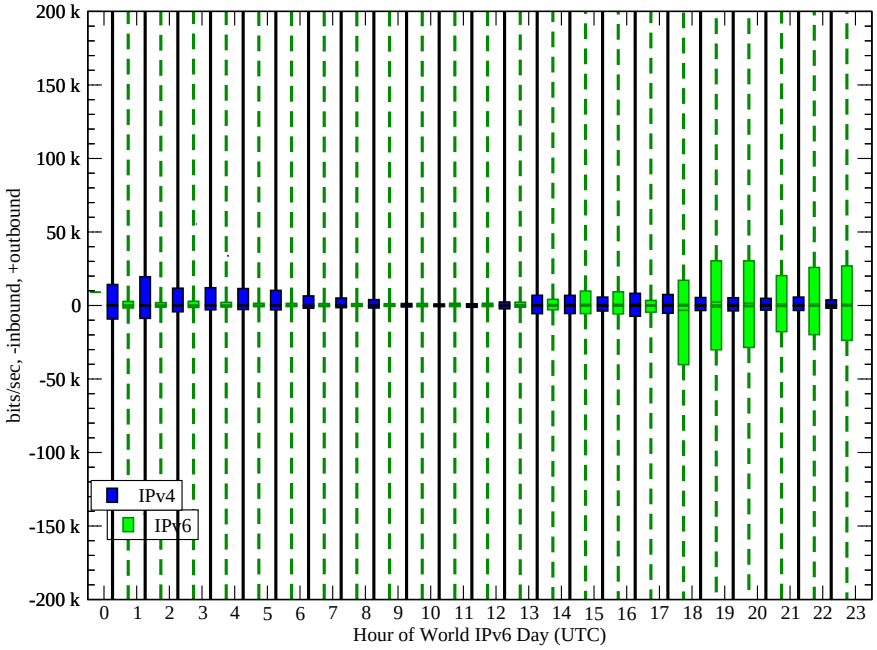


(b) Gmail WWW Flow Bit Rates

Figure 5.5: Flow peak performance for Facebook and Gmail on World IPv6 Day.



(a) Facebook WWW Flow Bit Rates (detail)



(b) Gmail WWW Flow Bit Rates (detail)

Figure 5.6: Flow peak performance detail for Facebook and Gmail on World IPv6 Day.

> The system introduced to analysis email headers is interesting and worthwhile. Clearly there is useful things to study in this space, and getting a large volume of donated user email data is interesting.
>
> anonymous reviewer, 2013

In this chapter, we take a new direction in exploring the utility of the notion of rendezvous in host profiling. Here, rather than observing DNS rendezvous traffic and associating it with application traffic, we hypothesize that we can reverse-engineer the rendezvous information from voluminous logs of email communication and knowledge of email routing, past and present. We develop a new analysis system called "Timemail," purpose-built to perform longitudinal study of email communication. What follows is a report on research amongst a team of collaborators consisting of Mark Allman, Paul Barford, Scott Cambpell, Vern Paxson, and David Plonka, whose personal email archive data sets are referred to by their respective names.

6.1 Overview

Given email's pervasive use over multiple decades, it is striking how the behavior and evolution of its transmission process has seen little study, especially in the face of seemingly ever-increasing complexity and challenges from nascent alternatives, *e.g.,* Facebook messaging. Certainly one impediment to study has been the lack of access to logs of email activity from a large set of sites and users and over extensive periods of time. The goal of our work is to develop privacy-respecting methods to broadly examine basic questions such as: What interconnection structure do nodes that participate in email communication exhibit? What is the nature of variability in end-to-end email delivery performance? How well has email functioned over time, what trends does its performance exhibit, and how might we improve its performance?

For decades email has been delivered on the Internet primarily via SMTP conversations using a common message format [41]. However, during this time

email delivery has evolved to include (i) additional protocols (*e.g.*, POP, IMAP), (ii) alternative routing methods (*e.g.*, MX routing), and (iii) additional message content processing (*e.g.*, virus and spam filtering). Each of these evolutionary steps has implications on the structure of the email system and ultimately its performance.

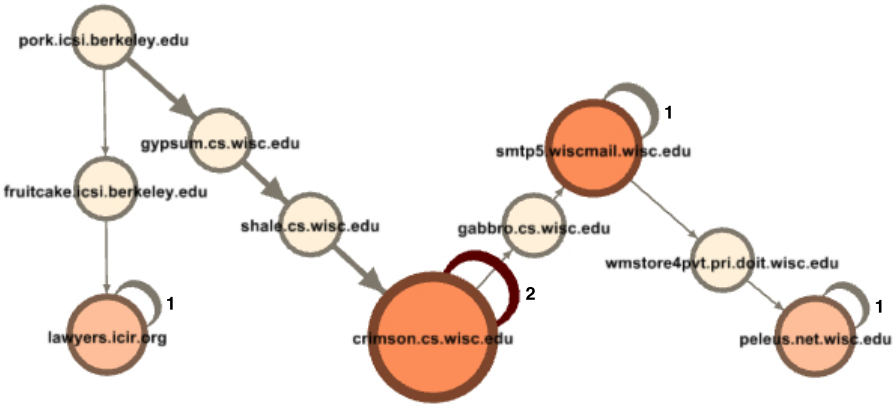


Figure 6.1: A sample of the delivery network as discovered by examining the Received headers of a message sent by one collaborator as delivered to three recipient collaborators.

We motivate our work with an example of an email sent in January 2010 from one of our collaborators (Paxson) to three others (Allman, Barford and Plonka), as illustrated in Figure 6.1. When we examine the received messages we find three qualitatively different delivery paths. All paths begin with Paxson's SMTP forwarder at ICSI: pork.icsi.berkeley.edu. Allman's path includes two distinct hosts, terminating at lawyers.icir.org, which is visited twice. (Intra-host loops are labeled here with the number of additional hops they represent.) The message to Barford visits three distinct hosts at the University of Wisconsin–Madison (UW–Madison) after the initial visit to the ICSI SMTP server, terminating at crimson.cs.wisc.edu. Plonka's path starts with the same SMTP hops as Barford's, but his path continues for a total of 11 hops, with visits to 7 distinct hosts within UW–Madison, terminating at peleus.net.wisc.edu. In fact, Plonka's email visits crimson.cs.wisc.edu three times while Barford's email visits it only once. We also note that delivery times vary from 5 sec (Barford) to 39 sec (Allman) to 91 sec (Plonka). This small example highlights a variety of email delivery paths and performances and raises numerous questions about the characteristics of email traffic, both in terms of modern traffic and how our current systems compare

to past incarnations.

To study the email delivery process we develop a methodology for large-scale, longitudinal analysis, and apply it to expose some of email's structural and performance characteristics. The genesis for our work is the observation that the header information included in the common message format [41] used to transfer and *store* email holds a tantalizing opportunity for large-scale longitudinal analysis. To leverage this potential, we create an infrastructure for gathering and extracting header information—mostly "Received" lines—from stored messages and develop methods for normalizing header information and for identifying the key components of email paths. We apply our methods to a set of personal email, mailing list, and spam archives to assess the efficacy of our approach. Our analysis reveals the diverse characteristics of email header information, structure of the email delivery network, facets of performance variability, and characteristics of the key structures in end-to-end delivery paths.

6.2 Data & Method

Our methodology is based on the collection meta-information about message delivery from archived email. The most crucial artifact we gather from the messages is the path a message took, as encoded in "Received" headers. These headers were made standard circa 1982 and are defined in a series of RFCs [41, 64, 65, 89, 92, 93]. Received headers differ from most email headers in that they are added to a message along its delivery path (*i.e.,* SMTP hops, IMAP hops, etc.) by mail transfer agent (MTA) software rather than by the sender's mail user agent (MUA) software [80]. Each Received header typically contains several key pieces of information, including: (i) a name for the remote host transmitting the message, (ii) a name for the receiving host, (iii) the protocol used for delivery and (iv) a timestamp. With the variety of message corpora used in this study we clearly observe the limitations of this data. For instance, we see timestamps from unsynchronized clocks, incorrect timezone offsets and names, servers that report no hostname or an ambiguous hostname, and forged or otherwise bogus Received lines and timestamps. Our analysis deals with all of these issues and either discards or repairs message information that is obviously bogus in some way. We generally make two key assumptions about the data we analyze:

Order: We assume the order of the Received lines has not been perturbed during delivery and therefore reflects the proper sequence of hops (modulo bogus insertion or deletion). This assumption allows us to, for instance, better detect clocks that are not working properly, *e.g.,* a clock that is way ahead of both its neighbors.

Legitimacy: We assume that the vast majority of the mail transfer agents that insert headers are acting legitimately and not intentionally using incorrect timestamps, deleting Received headers, adding extra Received headers or otherwise trying to obfuscate the delivery path. We understand that this does occur to be evasive (*e.g.,* spammers) or to keep internal server identities private. Our assumption is that these are outliers and therefore that statistical techniques can be used to treat them as such.

To analyze email meta-information we use a three step process: (i) collect data from donors and available archives, (ii) parse, store and clean the data, (iii) structure, query and graph the data.

We collect message data from donors using a script we call "`timemail`" that processes locally stored mail folders or an IMAP-accessible store of messages.

The script scans each message's headers and constructs a one-line summary of the delivery that includes name, protocol and timestamp for each hop in the path. The message size and "Message-ID" header, if present, are also recorded. We do not collect information from headers such as "To", "Cc", or "From" as these might reveal information about users who have privacy concerns. With an eye towards large-scale data collection, `timemail` is a self-contained, portable script that a data donor runs personally on their own computer and donates just its output that is uploaded to a central repository and imported into a database. Thus we neither store, nor even have access to, a donor's private email messages.

In practice, `timemail` has grown quite complex as we have had to deal with a wide variety of timestamp formatting, names and optional information in the message headers—which are often designed for human inspection as opposed to programmatic analysis. However, we find that we can determine an unambiguous timestamp to second granularity for 98.6% of the 42.3M Received headers represented in our data sets; the remainder is due to missing a timestamp or parsing failure (*e.g.*, missing seconds field, ambiguous hour of the day, foreign languages, etc.). To diagnose such issues, we preserve the timestamp text just as it appeared in the Received header; this has the added benefit of retaining locale-specific timezone offsets and names that offer hints for coarse geolocation.

Donor / Data set	Dates	Total Msgs.	Culled-out Msgs.	Net Msgs.
Plonka	1997–2013	169K	57K (33%)	113K
Campbell	1997–2009	39K	1K (3%)	38K
Allman	1994–2010	792K	303K (38%)	490K
Paxson	1988–2010	269K	50K (19%)	218K
Barford	2001–2010	27K	3K (11%)	24K
FreeBSD	1995–2010	220K	39K (18%)	182K
Linux	2000–2008	882K	669K (76%)	213K
IETF	1992–2013	1.9M	852K (45%)	1.0M
The Mail Archive	2013	1.8M	205K (11%)	1.6M
Guenter spam	1998–2007	948K	307K (32%)	641K
Dornbos spam	2002–2003	3.5K	719 (20%)	2.8K
TREC '07 ham	2007	23K	5K (21%)	18K
TREC '07 spam	2007	50K	19K (37%)	31K
CEAS '08 ham	2008	28K	24K (87%)	4K
CEAS '08 spam	2008	113K	47K (41%)	66K
Totals	1988–2013	7.2M	2.6M (36%)	4.7M

Table 6.1: Characteristics of data sets analyzed.

Our data sets include personal, mailing list, and spam-related email archives summarized in Table 6.1. We use a modest donor-base of personal email, *i.e.*, our own and those of our collaborators, as a means to work out the significant issues dealing with the data, as well as to begin getting a handle on whether there are useful and interesting insights to be gained. These personal archives vary in length and size and are biased by each donor's archiving habits and generosity; *i.e.*, a donation may be an unknown sampling of messages delivered.

Secondly, we use a set of mailing list archives: the FreeBSD list "freebsd-current" [11], the Linux Kernel Mailing List a.k.a. "lkml" [78], the Internet Engineering Task Force (IETF) mailing lists archive [17], and The Mail Archive [26]. The latter two are aggregate archives of hundreds and thousands of mailing lists, respectively. There are two sorts of mailing list archive corpora: (i) those collected at or near the list server itself, and (ii) those collected from the vantage point of a list subscriber; messages in the latter typically, but not always, record a series of two complete deliveries in their Received headers: a delivery to the list itself and a second delivery to the list subscriber.

Last, we include a set of spam archives. These include the Guenter and Dornbos archives [47, 54], and the public spam with accompanying "ham" corpora from spam-related conferences [38, 39]. We report only culling and classification statistics for these archives due to questions regarding the legitimacy of initial Received headers from spammers and regarding how the spam training and testing corpora were assembled.

Figure 6.2 offers a glimpse of the richness and irregularity of the collected data; it shows the message delivery time for 152K messages in Plonka's data set. Note clusters with delays of minutes to hours, typically due to multiple deliveries through mailing lists. The plateaus at 5 minutes are due to POP and IMAP polling to pull mail, and are separated by a period of direct delivery (c. 2002–2006). Plots of the other data sets (now shown) show similar variability, but with very different episodes, clusters, or plateaus in delivery times.

As shown in Table 6.1, 36% of the message records across our 15 data sets are culled-out before the analysis described in subsequent sections. Once imported into the database, messages are examined and culled by our `cullmail` tool. The most common reason to exclude an email message from an analysis (82.4% of the removals) is that the Received headers in the message show at least one instance where the timestamp for hop N is less than the timestamp for hop $N - 1$. This is indicative of either unsynchronized clocks and/or incorrect timezone information, either of which would clearly skew some of our

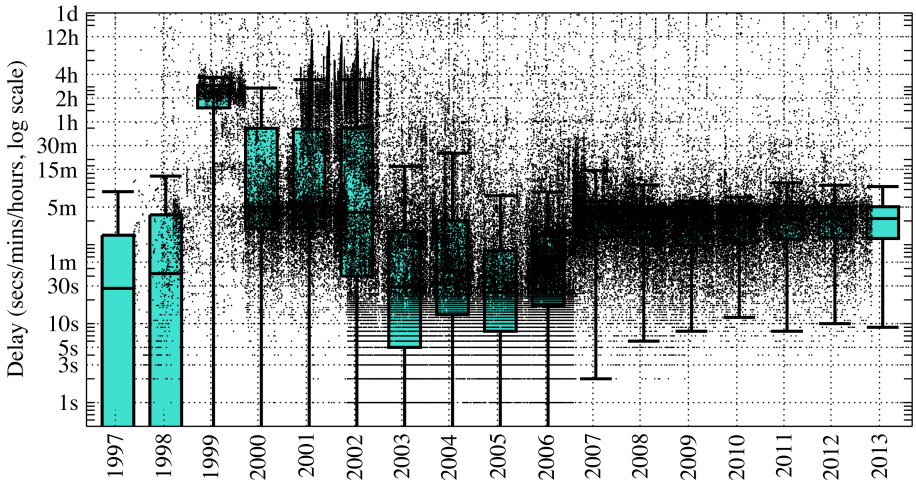


Figure 6.2: Scatter and box plots of delivery times for 152K of Plonka's messages, prior to culling. The boxes represent the middle fifty percent about the median, and the whiskers extend to the 9th and 91st percentiles.

subsequent analyses (for which we do not assume tightly synchronized clocks, but removing cases with obvious issues is worthwhile). Messages were also removed from further consideration when timestamps from the distant past or from the future (after `timemail` processed the messages) were encountered, as these point to obvious clock issues.

While having removed many erroneous timestamps by the method outlined above, we are quite certain that some remain, *e.g.*, when the first or last hop timestamps err toward the past or future, respectively. Without a ground truth reference clock, our results rely on our assumption of legitimacy. We also retain the details of culled-out messages in our database because they are useful for a subset of analyses; for instance, the delivery path can expose network structure even when the accompanying timestamps are suspect.

6.3 Structure

Aiming to understand performance characteristics in topological context and to identify sources of delay, we determine the overall structure of the email delivery network at points in time. This allows us to correlate delivery latencies with specific network elements and their role in the network. For instance, a given MTA host in a message delivery path might be an email submission server, a mailing list processor, a full Mail-eXchanger (that accepts MX routed email for a domain via SMTP), a spam or anti-virus scanner, or a personal workstation.

We organize the data by introducing a schema that specifies a structure and a type system for its elements. This structure involves three types of elements: *messages*—the delivered email objects, *nodes*—the agents that handle and transfer email messages, and *edges*—the connections that are traversed by email messages during transmission. The type system classifies the nodes (based on their name as reported in a Received header) and, in turn, classifies edges based on type, name, and protocol of their end-point nodes.

From this network structure, we define a relational database having rows that store instances of message deliveries, node visits, and edge traversals. The message element is used to (i) store a message's summary information for each instance of delivery and to (ii) group sequences of nodes and edges in the delivery path. The Timemail database schema is shown in Appendix C. The node and edge elements are described below.

6.3.1 Nodes

Nodes represent visits to the MTA that wrote a Received header, typically either a server transferring the message or software processing the message within a server. Servers that write Received headers can use arbitrary names to identify themselves; that is, the names are not necessarily globally unique Fully-Qualified-Domain-Names (FQDN) within in the Domain Name System. While most reported names *appear* to be FQDNs, *e.g.,* mail.example.net, it is quite common to see ambiguities such as bare hostnames, *e.g.,* mailhost, localhost and variants, *e.g.,* localhost.localnet, bare domain names, *e.g.,* example.com, invalid Top-Level-Domains (TLDs), private IP addresses (RFC1918), or bogon IP addresses that are not globally routed. This necessitates the disambiguation of the node names so that we avoid mistakenly assuming these nodes have very

high degree of connectivity and grossly misrepresenting the overall structure of the email delivery network. To that end, our tool called `typemail` uses rules and regular expressions to assign types to the nodes, as shown in Table 6.2.

Node Type	Count (%)
Valid-TLD (V)	34,338,330 (81%)
Invalid-TLD (I)	4,181,576 (10%)
Localhost (L)	2,346,671 (5%)
Private-IP-Address (P)	1,322,433 (3%)
Valid-IP-Address (A)	242,223 (< 1%)
Bogon-IP-Address (B)	2805 (< 1%)
Total (Received headers)	42,313,038

Table 6.2: Node types and their respective number of occurrences in our data sets. 81% of Received headers identify a host with a valid Top-Level-Domain.

6.3.2 Edges

The nodes are connected by edges representing the transition from visiting one node to the next as an email message traverses the delivery network. Introducing this network element is motivated by the fact that a Received header reports a single timestamp, rather than one for arrival and one for departure. Thus, it is difficult to determine whether delay is due to the origin node, the propagation between nodes, or the destination node. Given that we ultimately wish to attribute observed delays to subnetwork and specific server bottlenecks, our type system assigns types to edges as well, as shown in Table 6.3, indicating their position or role in the network's structure. These annotations are performed by the `typemail` and `rolemail` tools, the latter assigning the role of "P" to edges that involved Pull-based protocols (*e.g.*, POP, IMAP) so that we can separate these user-induced delays from those of push-based (*e.g.*, SMTP), store-and-forward email delivery.

Note that, in this current type system, there are two situations in which an edge representing a Pull-based delivery will be assigned a role of "P" but does not involve a user-induced polling delay. The first situation, likely quite rare, is when a message happens to become available on the server (*i.e.*, just following its arrival) immediately before a client polls for (new) messages. The second situation is specific to the IMAP protocol; although IMAP originally always required a client to poll the server for changes to a mailbox, an IDLE command was introduced that allows a client to tell a server that it is able

to accept real-time updates, such as an EXISTS notification that new email message has arrived. [67] If the client reacts to this notification by immediately performing a FETCH of that email message, then there is no arbitrary delay due to polling. Explicitly handling this latter situation is left for future work; one candidate method to address this would be to introduce another role, such as "p", to indicate that an IMAP FETCH appears to have been prompted by a real-time notification. The appropriate assignment of the "P" role (polled Pull) versus "p" (prompted pull) for an edge involving IMAP could, perhaps, be achieved by a heuristic that considers whether the distribution of delivery times observed across the given edge matches known distributions for typical polling behavior or not.

When determining types for edges, we also fix ambiguous origin and destination names such as localhost. For instance, it is common to see a transfer from mail.example.com to localhost, followed by localhost to mail.example.com. In such instances, we rewrite localhost to mail.example.com and set the type of the two edges to "H" (intra-Host). Such heuristics help determine the domain in which elements reside.

Edge Type	Count (%)
Cross-Domain (X)	11,904,180 (34%)
Intra-Host (H)	11,746,685 (33%)
Intra-Domain (D)	10,571,255 (30%)
Unknown (U)	869,575 ($<$ 3%)
Total	35,091,695

Table 6.3: Edge types and their respective number of occurrences. Over 97% in our imported data sets were identified as one of the three types.

6.3.3 Graphs

Having defined nodes and edges, we naturally wish to produce a graph for visualization and analysis of the email delivery network. These are generated by our `graphmail` tool that selects messages based on donor, time frame, domains or other path characteristics, and weights edges by message volume, delay, measures of delay dispersion, or type. We find two styles of graph useful: overlay graphs and flow graphs.

Overlay graphs are directed graphs of a set of observed paths in which all nodes of a given name are overlaid on each other, such that a node's name is

unique in the graph. Figure 6.9 is an example of an overlay graph. While this graph is concise, it is imprecise since it may contain cycles and therefore paths that were never observed in Received headers and may be impossible due to MX or other routing policies. However, we use the overlay graph to calculate a node's aggregate in-degree; we hypothesize that SMTP Mail-eXchangers can be identified by their high in-degree, given sufficient sampling of delivery paths, and have confirmed this for the collaborators' local domains, but general validation is future work.

Flow graphs are directed graphs that more precisely represent a set of observed delivery paths in which nodes of a given name are not necessarily unique; nodes represent visits to a given MTA at a specific *position* in the delivery path and the number of hops on paths through this graph matches the number of Received headers in the messages having that delivery path. Flow graphs are actually trees that are rooted at the terminal nodes and having branches containing each and every observed delivery path. A hybrid of overlay and flow graphs overlays only adjacent nodes of the same name. For an even more concise graph, nodes can be labeled by MTA domain instead of full MTA name. Samples of such graphs are shown in Figures 6.3 and 6.4. In these figures, peripheral nodes with high in-degree are those representing domains hosting mailing lists that are ultimately archived within ietf.org or mail-archive.com, respectively.

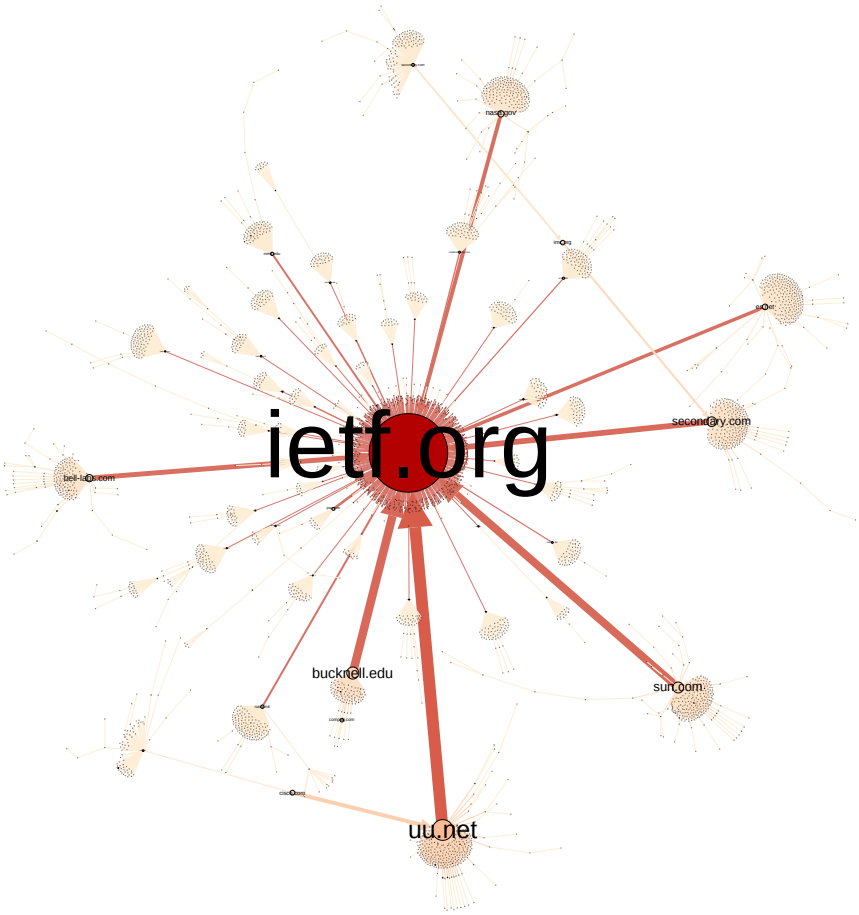


Figure 6.3: An MTA domain-level flow graph for message deliveries in the year 2000 in the IETF data set. Edges are weighted by message volume and nodes are sized and shaded by weighted degree.

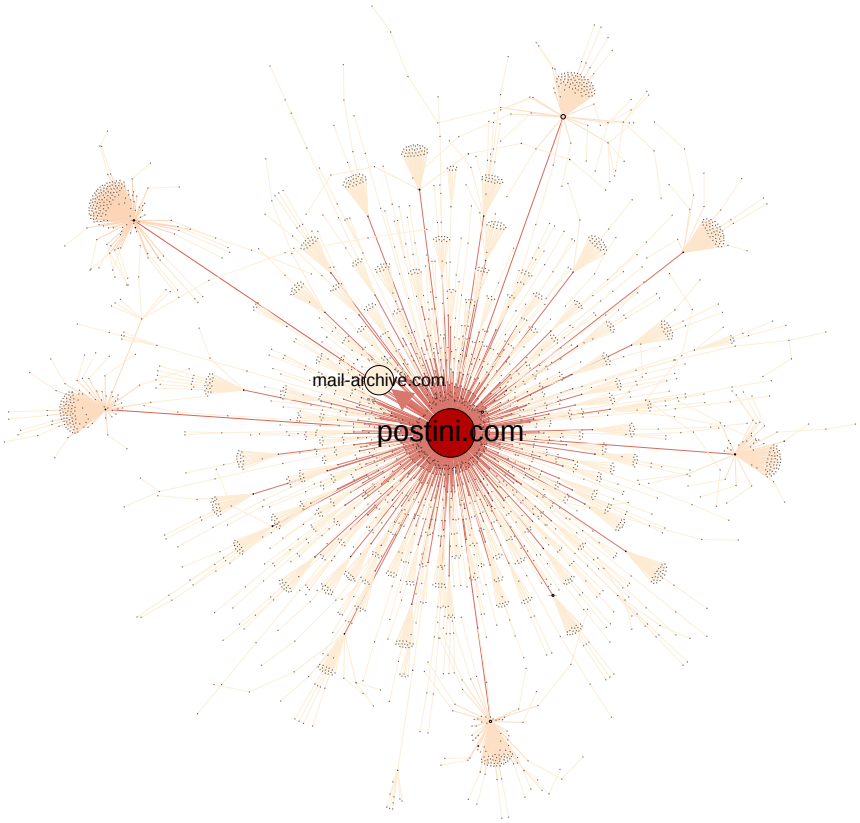


Figure 6.4: An MTA domain-level flow graph for message deliveries on April 1, 2013 in The Mail Archive data set. Edges are weighted by message volume and nodes are sized and shaded by weighted degree.

6.4 Results

The email data we've collected and organized offer myriad possibilities for data mining. Initially, we validate our findings by leveraging our familiarity with the collaborators' personal archives and transparent organizations hosting publicly available mailing lists.

6.4.1 End to End Delay

To investigate end to end performance, we first identify each data set's most common terminal domains, *i.e.*, domains in which the MTAs performing the final delivery reside. Then, we choose a set of the most common initial domains, *i.e.*, the domains of initial MTAs that have forwarded at least 500 messages in the data set, and plot the distribution of delivery times of all messages, pairwise between those initial and terminal domains. Figure 6.5 is one example. Here we see a wide range of delivery times and delay dispersion for the messages forwarded from 35 domains in the Allman data set, with his local domains being two of the three with least delay.

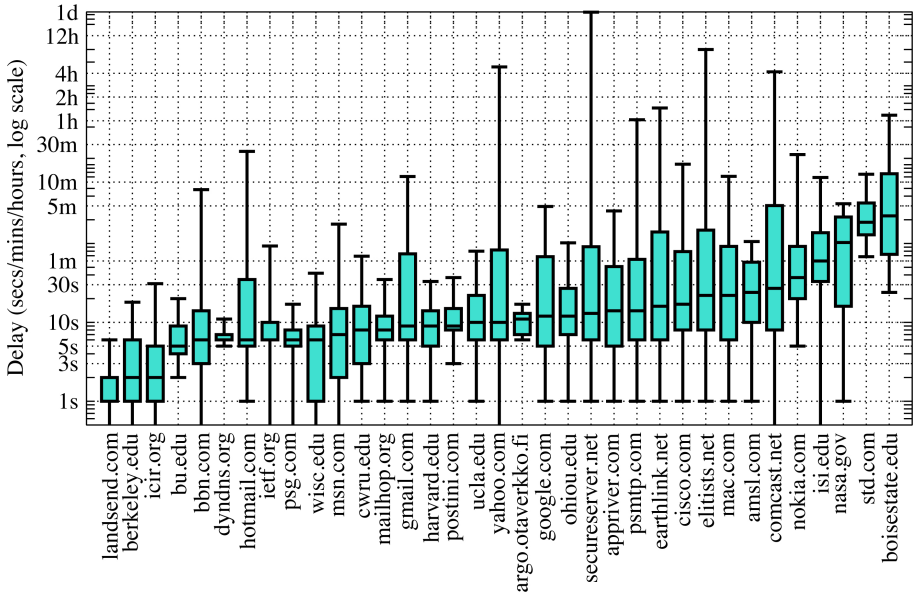


Figure 6.5: Delivery times for 292K of Allman's messages from the most common initial domains in delivery paths to his terminal domains, 2003–2010, sorted by median delay. The boxes represent the middle fifty percent about the median, and the whiskers extend to the 9th and 91st percentiles.

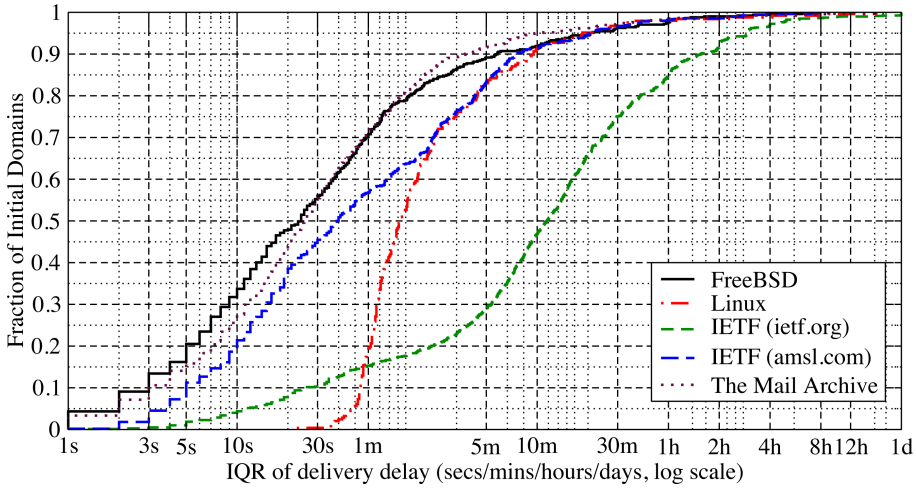


Figure 6.6: CDFs by mailing list data set for the IQRs of delivery times of each set's messages from significant initial domains: 256M messages from 2,310 domains.

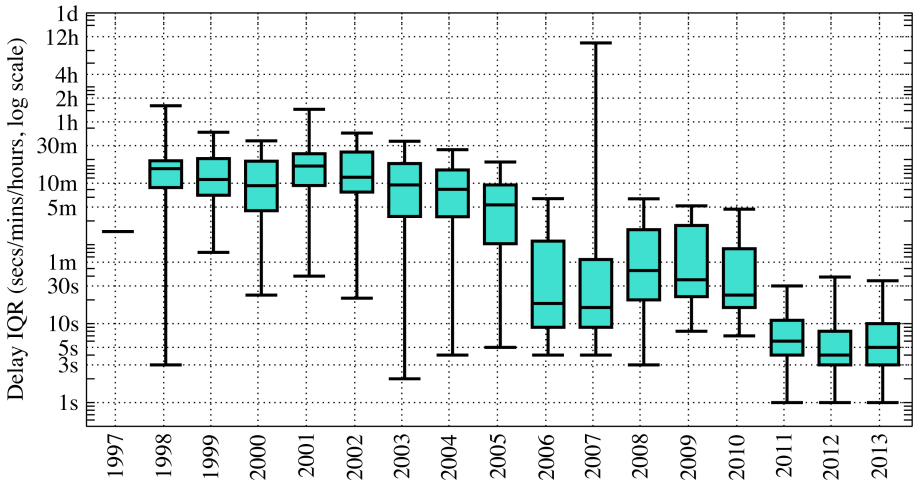


Figure 6.7: Interquartile ranges of delivery times for 433K messages from 798 initial domains in the IETF data set since 1997.

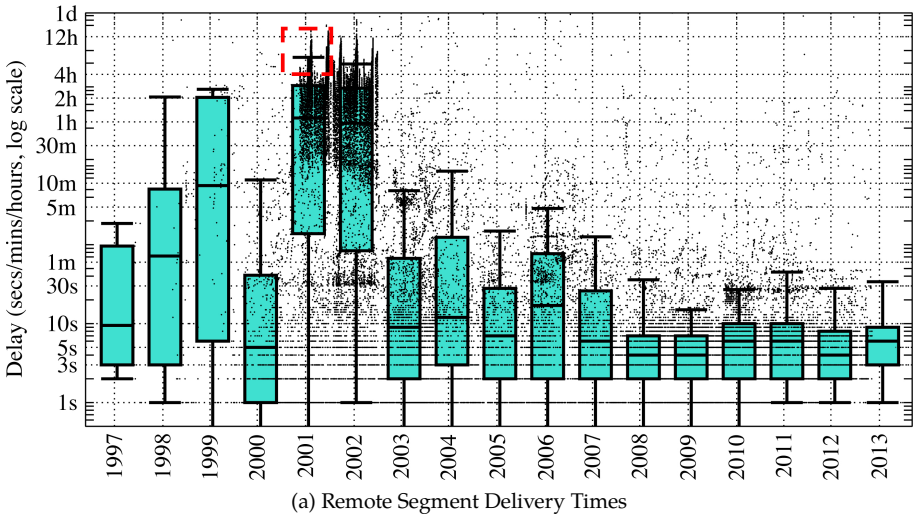
Figure 6.6 is a plot of the cumulative distribution function (CDF) for each of the mailing list archives delay dispersion as measured by interquartile range (IQR), for initial domains that have forwarded at least 50 messages in the data set. (Our intent is to select only those domains with a sizeable number of sample deliveries to calculate delivery delay dispersion.) We've split the IETF data set in

two, when the archives' terminal domain changed from ietf.org to amsl.com with the move to a new managed services provider on February 1, 2008 [82]. There is significantly less delay dispersion afterward for "IETF (amsl.com)." To examine this delay dispersion over time, we first calculate the IQR of the delay for each of those significant initial domains by year, then produce box plots from those domains' delay IQR values, as shown in Figure 6.7. Here we see a trend toward lower delay dispersion over the past decade for the IETF mailing lists archive. The significant changes from 2005 to 2006 and 2007 to 2008 are validated by their correspondence to (i) when the IETF acquired data management services under a 2-year contract at about the beginning of 2006 [52], and (ii) when the IETF switched to a new service provider on February 1, 2008 [82] that continues to operate the IETF mailing lists infrastructure today.

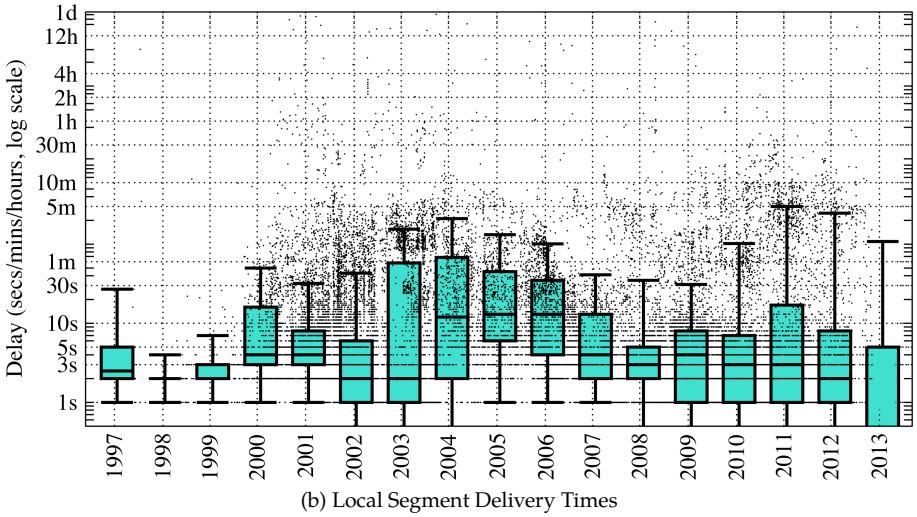
6.4.2 Delay Centers

One goal of our work is to identify the elements of the email delivery network most responsible for delay and delay dispersion and how these change in time. To identify delay centers, we first divide the delivery path into remote and local segments. We do this by considering each message's path individually and finding its last cross-domain edge (type "X") that has a destination MTA within the data sets' terminal domains. We typically find that MTA to be a Mail-eXchanger for a terminal domain. The path up to and including that edge is the remote segment and the rest of the path is the local segment. Figures 6.8a and 6.8b are sample plots of the delays observed while traversing those segments of the messages' delivery paths. Here we see that the remote and local segments of the delivery path exhibit very different performance.

In Figure 6.8a, we see some very long delivery times in 2001 and 2002. To identify the source of those delays, we use `graphmail` to select messages in 2001 with remote delay between 4 and 15 hours, *i.e.*, those in the dashed-line quadrant, and graph the delivery network. The result, in Figure 6.9, prominently shows that nearly all the delay (93%, in fact) was incurred on an intra-host edge to/from loki.ietf.org, a server used to send messages to subscribers of the "IETF-Announce" and "Internet-Drafts" mailing lists. This graph also shows Plonka's complete delivery terminus infrastructure at the time, including redundant anti-virus scanners {av1,av2}.doit.wisc.edu, other email servers {mail2,mail3,mail4}.doit.wisc.edu, and terminal host mil.doit.wisc.edu. The other high delay-inducing MTA discovered, roam.psg.com, is verified to be a laptop



(a) Remote Segment Delivery Times



(b) Local Segment Delivery Times

Figure 6.8: Delivery delay for the remote segments and local segments of delivery paths for 34K of Plonka's messages ostensibly having a single delivery in their path, *i.e.*, just one type "X" edge. The boxes represent the middle fifty percent about the median, and the whiskers extend to the 9th and 91st percentiles.

host that was sometimes disconnected from the Internet, thus its outbound email sometimes experienced these long delays. [28]

Our results thus far have shown both challenges and opportunities in studying the trace data. While trends have generally not emerged and each data

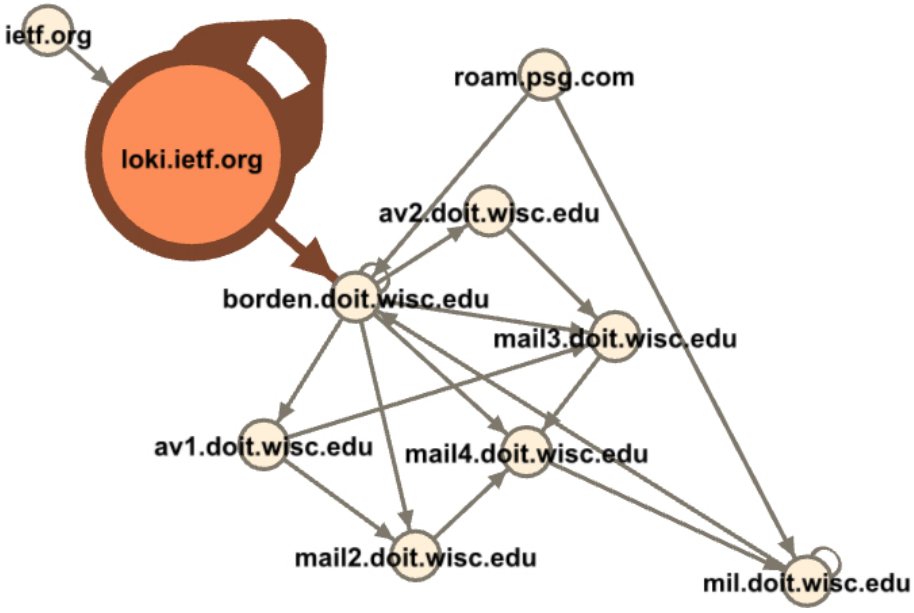


Figure 6.9: A graph of the delivery network discovered from 873 messages having delays shown in the dashed-line quadrant in Figure 6.8a. Edges are weighted by delay and nodes sized by weighted out-degree.

set has curiosities, *e.g.,* due to capricious email archiving behaviors or idiosyncrasies in terminal domains, we've isolated performance phenomena and employed our tools to identify their source in structural context.

7 SUMMARY, CONCLUSIONS, AND FUTURE WORK

> The idea of mapping IP addresses with domain names is cool.
>
> anonymous reviewer of [25], 2012

7.1 Summary

In this dissertation we advocate novel analysis techniques involving Internet rendezvous, *i.e.,* the method by which Internet hosts are introduced prior to establishing communication, to perform traffic measurements, traffic classification, and host profiling. We develop these techniques in four parts and employ them in four empirical studies that demonstrate their performance. In this chapter, we summarize that work as outlined in prior chapters and close by enumerating directions for future work.

7.1.1 Context-aware Clustering of DNS Query Traffic

In Chapter 3 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 portion of our work are canonical queries consistent with 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 in the sense that 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 details of characteristics of unwanted traffic. We believe that these tests highlight the novelty and utility of our methods.

Ultimately, this study of DNS traffic lays the foundation for our rendezvous-based traffic classification and host profiling by identifying the subset of DNS traffic that is useful in classification, *i.e.,* that which is not just a curiosity of this popular service, the Domain Name System.

7.1.2 Flexible Traffic and Host Profiling via DNS Rendezvous

Based on the tool and foundation laid in Chapter 3, we present a novel traffic classification method based on DNS rendezvous, *i.e.,* the domain names by which end-hosts present and discover IP addresses, in Chapter 4. Our rendezvous-based approach combines some of the best characteristics of prior methods: (i) port numbers are not implicitly trusted, (ii) deep packet inspection of the target traffic is not required, and (iii) packet sampling of the target traffic is not an obstacle. The goal of our work is to add flexibility in classification with high accuracy of classification in live operational deployments. This approach gleans information from the most common rendezvous method, the DNS, which is widely used and offers flexible options to both profile hosts and classify their traffic.

We demonstrate the feasibility and utility of rendezvous-based classification by implementing and extending our method in the TreeTop tool and applying it to DNS traces and flow-export data gathered from a campus network, focusing on two starkly different user groups' traffic for a typical day. We show that a large proportion of the traffic from the office group is arranged via the DNS, enabling it to be directly classified by our method. In the residential group, where a significant amount of traffic is not is preceded by DNS queries, we implement two alternatives: (i) we apply the port-based method selectively, to just the named traffic, to minimize that method's false reports, and (ii) we infer labels for unnamed traffic by profiling the end-hosts involved, based on their DNS activity. These initial results demonstrate how a traffic classifier can make effective use of a hitherto untapped, independent source of information, *i.e.,* the Domain Name System.

Finally, we demonstrate that rendezvous-based techniques can work in concert with prior techniques to bolster overall classification performance.

7.1.3 Assessing Performance of Internet Services on IPv6

In Chapter 5 we identify a new passive measurement application for traffic classification via rendezvous that demonstrates a unique capability enabled by using a source of information from outside the target traffic itself. We present a method to examine the performance of Internet services on IPv6 and IPv4, with which clients rendezvous via the DNS. Our approach is a new application of TreeTop's traffic classification technique that doesn't perform active measurements, doesn't need "insider" knowledge about those services IP addresses, and doesn't require inspection of application traffic payloads that may be encrypted, obscured, or otherwise unavailable. Instead, it relies on low-volume DNS query/response traffic and easily-obtained application transport information from packet headers.

We demonstrate the feasibility of the approach by implementing our method in the TreeTop Framework and a set of assessment tools. We demonstrate its utility by analyzing DNS traces and flow export data gathered from a campus network with an advanced deployment of IPv6 via dual-stack hosts, focusing on their traffic on the World IPv6 Day. A large proportion of traffic involving services running IPv6 is arranged via the DNS, allowing the associated service (*e.g.,* Facebook or Gmail) to be directly identified. While we find that dual-stack IP implementations complicate measurement, our method is able to infer service identities by a consensus of hosts in the monitored population.

These sample results demonstrate how a rendezvous-based technique can be effective in assessing the IPv6 performance of services during their deployment and side-by-side operation over IPv4. We apply robust statistics to observed flow bit rate distributions for Facebook and Gmail traffic and present them as graphs allowing visual detection of performance differences and anomalies. Such capability can inform developers, operators, and users with respect to selection of which Internet Protocol version is likely to yield better performance, making it more likely that Internet services' transition to IPv6 will meet or exceed expectations.

7.1.4 Toward Longitudinal Study of Email Communication

Lastly, motivated by anecdotal experiences of highly variable email delivery performance and the opportunity presented by ubiquitous Received headers present in everyone's email message archives, we take a new direction in rendezvous-based analysis that involves host profiling and rendezvous information for

the email application. Rather than directly observing transmitted rendezvous information as in our other studies, here we hypothesize that we can reverse-engineer some rendezvous configurations based on Received headers that are artifacts evidencing prior rendezvous.

In Chapter 6, we describe a method for studying the Internet email communication system, past and present. With data donors' privacy in mind, we've developed and exercised the Timemail System [88], applying it to both personal and publicly available email message archives. A sample of early results expose interesting aspects of performance and structure and suggest richer results at large scale. Our current goal is, thus, to collect and analyze a broader range of data to better understand this critical system's behavior and to inform efforts in its ongoing design and operation.

7.2 Conclusions

In conclusion, the stepwise work outlined in this dissertation has culminated in the development and operation of new Internet traffic measurement techniques and of two tools, TreeTop and Timemail, that we have applied in our research to yield numerous novel results. Furthermore, our rendezvous-based method has been made available to industry for commercial use [22] in network operations and has been independently reimplemented yielding novel results in others' research studies reported in the literature. [25, 48] Thus, we assert that we have validated our thesis: our rendezvous-based methods are feasible and useful in measurement, classification, and profiling.

7.3 Future Work

The focus of our work in this dissertation is to improve the state of traffic classification for Internet research and operations. While we believe that the empirical studies described herein evidence our methods' effectiveness, there are a number of promising future directions suggested by both the rendezvous classification and email delivery studies, including the following:

1. While we develop and demonstrate rendezvous-based traffic classification based on the DNS, there are many other rendezvous mechanisms that are used; tapping those other rendezvous services, *e.g.,* P2P and SIP, would improve overall classification. It is especially tempting to pursue those where the rendezvous is easily separable from the application traffic when passively observed in the network. Also, where the rendezvous information is not easily discerned, *e.g.,* when it passes over encrypted channels, temporal analysis involving prior communications could help to determine the possible channels by which a host may have learned another host's IP address. This may be similar to work that investigates host compromises and command and control channels since they often involve serial communications wherein one stream is "dependent" on another (via a prior compromise or via tunneling).

2. The reliance on an application-independent source of information, such as the DNS, to perform traffic classification raises the question: in what situation should or shouldn't one trust passively observed rendezvous information? That is, do we know that the DNS queries and responses that we observe are legitimate and, thus, trustworthy for use in classification? While we trust the recursive DNS service infrastructure on our campus in these studies, our work could be extended to validate DNS responses based on Secured DNS (DNSSEC). DNSSEC information can be verified even when observed by an element that neither generated the query nor response. Otherwise, this concern of legitimacy may be able to be addressed similarly to that in work that determine IP address or domain *reputation*; *i.e.,* what is the past reputation of a given DNS server or those servers that are authoritative for a domain? Also, it would be useful to determine whether a given domain allows dynamic update, *e.g.,* dyndns.org, since the DNS rendezvous information is no longer independent in such a case.

3. While we've introduce a notion of host profiling based on a host's peer-to-peer-related DNS queries related to file sharing, gaming, and chat/messaging services, there are many other possibilities. Rather than a network administrator having to develop these profiles in advance, future work could investigate automatically generating dynamic rendezvous-based host profiles based by observation of a population of hosts. Subsequently, other hosts could be profiled by similarity or difference to those dynamically created profiles. It is interesting, for instance, to consider where and when one might trust one host's rendezvous behavior to define a profile/class for others.

4. While we've successfully applied the DNS query responses to the classification of associated traffic, it remains the case that this isn't an intended purpose of the DNS. Thus, it's worthy to consider how the DNS and other rendezvous services and protocols could be improved to explicitly account for traffic classification. For instance, the DNS (Service Location) SRV [55] query approaches something like this by including both the name and desired service or port number in the query. By augmenting rendezvous queries with clear information about a client's intended use of the host identity that it receives, a rendezvous service could play an increasingly valuable role in traffic classification and determination of packet treatment. (Conversely, if rendezvous information is encrypted, it may severely limit traffic classification, possibly resulting in too much traffic being forwarded on only a "best effort" basis.)

5. In studying IPv6 versus IPv4 performance, we provide a rendezvous-based measurement method to expose performance, however we do not determine the root cause of differences, when they exist. Future work exploring other visualizations and forensic tasks are necessary to determine these root causes. These causes may relate to the protocol, clients, servers, or other elements in the end-to-end path.

6. In our studying structure and performance of the email delivery network and its processes, we note that the performance information is gathered from messages stored in personal archives that are likely an "unknown sample" of all messages that were delivered. The sample is likely biased based on the interests and behaviors of capricious users. It would, of course then, be useful to determine how representative a given sample

of email messages is, with respect to the delivery performance recorded in those messages' Received headers. To this end, we propose studies that gather data more broadly by operating of filters (such as `timemail`) on every one of a recipient's messages, rather than just those that they choose to save in their personal archives.

7. In Chapter 6, we examine the structure of the email delivery network at arbitrary points in time past or present. We hypothesize that an MTA node's in-degree and corresponding node and edge characteristics can be used to reverse-engineer whether or not that MTA was, or is, an SMTP Mail-eXchanger. This hypothesis has yet to be fully tested, but could be done on a large scale by utilizing existing passive DNS monitors and databases, such as DNSDB [16], in future work.

The following are the domain names and suffixes that define the P2P profiles used in the work presented in Chapter 4. The Torrent entries are domains of popular BitTorrent Clients from Wikipedia [13] and from Alexa's top "Torrent Directories and Tracker Domains." [7] The Talk entries are based on observed behavior of the Skype application and a well-known domain. The Game entries are a well-known online game domain and Alexa's top "Massive Multiplayer Online Domains." [7]

- Torrent:
  bittorrent.com
  utorrent.com
  bitcomet.com
  turbobt.com
  pingpong-abc.sourceforge.net
  bittornado.com
  bitlord.com
  azureus.sourceforge.net
  azureusplatform.com
  vuze.com
  limewire.com
  acquisitionx.com
  aria2.sourceforge.net
  ktorrent.org
  transmissionbt.com
  thepiratebay.org
  isohunt.com
  mininova.org
  demonoid.com
  hongfire.com
  torrentportal.com
  bwtorrents.com
  torrentbox.com
  torrentresource.com
  legaltorrents.com
  mybittorrent.com

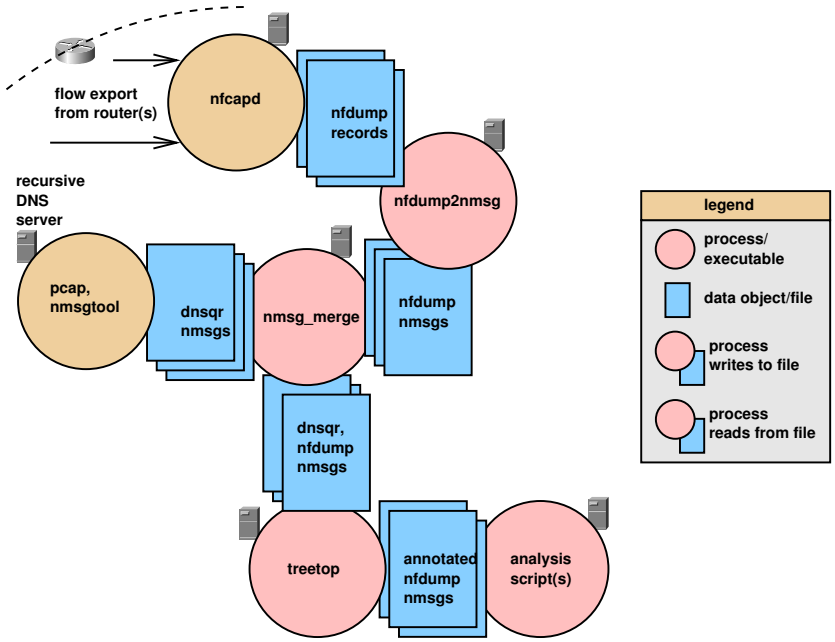
boxtorrents.com
tuxdistro.com
thebeehive.info
bittorrenttopsites.com
torrenttyphoon.com

- Talk:
  ui.skype.com
  talk.google.com

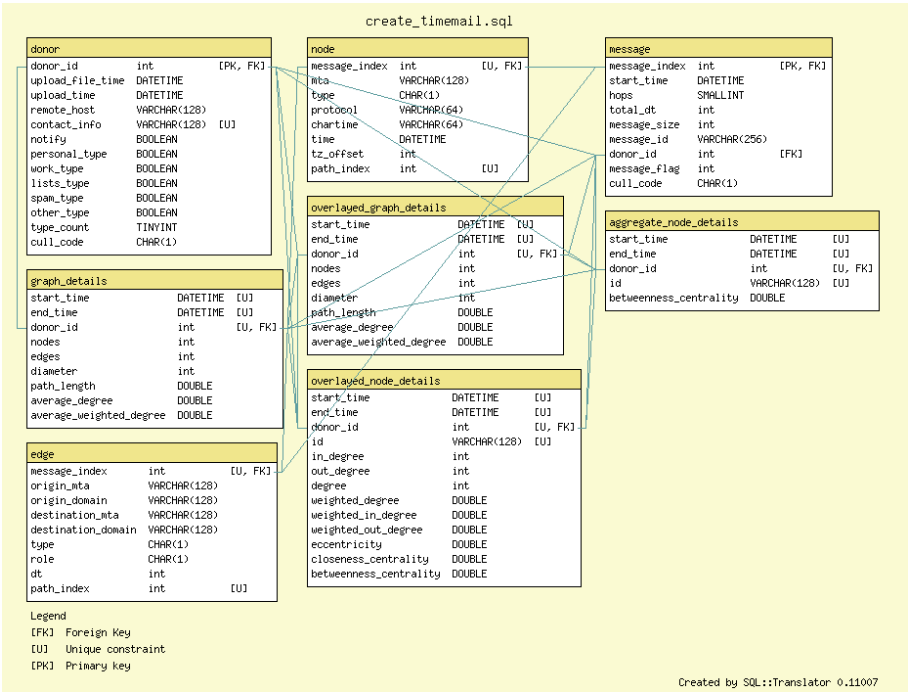
- Game:
  battle.net
  worldofwar.net
  wowhead.com
  runescape.com
  worldofwarcraft.com
  wowwiki.com
  mmo-champion.com
  curse.com
  thottbot.com
  dofus.com
  mmosite.com
  tibia.com
  mmorpg.com
  startrekonline.com
  guildwars.com
  allakhazam.com
  onrpg.com
  elitistjerks.com
  arenajunkies.com
  wowinterface.com
  wow.allakhazam.com
  warcraftmovies.com
  zybez.net
  tankspot.com
  ddo.com
  wowprogress.com
  runehq.com

maxdps.com
runescape.salmoneus.net
lotro.com
play.toontown.com
lineage2.com
guildomatic.com
mabinogiworld.com
ragnarokonline.com

The following figure shows the TreeTop Framework for rendezvous-annotated flow export as discussed in Chapter 5.



The following figure shows the relational database schema for the Timemail system discussed in Chapter 6.



DISCARD THIS PAGE

> Besides, the writing and the plots for this paper is not polished.
>
> anonymous reviewer, 2013

This dissertation was prepared using LaTeX and the Wisconsin dissertation template by William C. Benton. Figures were prepared using RRDTOOL, Grace, Graphviz, xfig, Gephi, and SQL::Translator.

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

[2] tcpdump. http://www.tcpdump.org.

[3] wireshark. http://www.wireshark.org.

[4] 2008. CoralReef. http://www.caida.org/tools/measurement/coralreef/.

[5] 2009. flow-tools: Tool set for working with NetFlow data. http://code.google.com/p/flow-tools/.

[6] 2009. Peer-to-Peer Session Initiation Protocol (p2psip). http://www.ietf.org/dyn/wg/charter/p2psip-charter.html.

[7] 2010. Alexa Internet, Inc. http://www.alexa.com/topsites.

[8] 2010. ipoque GmbH. http://ipoque.org.

[9] 2010. McAfee SmartFilter. http://www.mcafee.com/us/enterprise/products/email_and_web_security/web/smartfilter.html.

[10] 2010. Sandvine Incorporated. http://www.sandvine.com/.

[11] 2010. The freebsd-current Mailing List Archive. ftp://ftp.freebsd.org/pub/FreeBSD/doc/mailing-lists/archive/.

[12] 2010. Websense, Inc. http://www.websense.com.

[13] 2010. Wikipedia: Comparison of BitTorrent Clients. http://en.wikipedia.org/wiki/Comparison_of_BitTorrent_clients.

[14] 2012. NFDUMP. http://nfdump.sourceforge.net/.

[15] 2012. nmsg: network message library. http://rsfcode.isc.org/git/nmsg/.

[16] 2013. Internet Systems Consortium DNS Database (DNSDB). https://dnsdb.isc.org/.

[17] 2013. The IETF Mailing Lists Archive. http://www.ietf.org/mail-archive/text/.

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

[19] Afergan, M., and R. Beverly. 2005. The State of the Email Address. *ACM SIGCOMM Computer Communication Review* 35(1):36.

[20] Agarwal, S., V.N. Padmanabhan, and D.A. Joseph. 2007. Addressing Email Loss with SureMail: Measurement, Design, and Evaluation. In *Proceedings of the 2007 Usenix Annual Technical Conference (USENIX'07)*. Santa Clara, CA.

[21] Alexander, S., and R. Droms. 1997. DHCP Options and BOOTP Vendor Extensions. IETF RFC 2132.

[22] Barford, P., and D. Plonka. 2011. Apparatus and Method for Classifying Network Packet Data. United States Patent 7907543.

[23] Barros, M. Taynnan, R. Gomes, M. S. Alencar, and A. Costa. 2013. IP Traffic Classifiers Applied to DiffServ Networks. In *Proceedings of the Eighteenth IEEE Symposium on Computers and Communications (ISCC 2013)*. Split, Croatia.

[24] Baset, S.A., and H. Schulzrinne. 2006. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *Proceedings of IEEE INFOCOM '06*. Barcelona, Spain.

[25] Bermudez, I., M. Mellia, M. Munafo, R. Keralapura, and A. Nucci. 2012. DNS to the Rescue: Discerning Content and Services in a Tangled Web. In *Proceedings of ACM Internet Measurement Conference (IMC '12)*. Boston, MA.

[26] Breidenbach, J. 2013. The Mail Archive. `http://mail-archive.com`.

[27] Brownlee, N. 1997. Traffic Flow Measurement: Experiences with NeTraMet. IETF RFC 2123.

[28] Bush, R. 2013. Personal correspondence.

[29] Caballero, J., P. Poosankam, C. Kreibich, and D. Song. 2009. Dispatcher: Enabling Active Botnet Infiltration Using Automatic Protocol Reverse-Engineering. In *Proceedings of the 16th ACM conference on Computer and Communications Security*. Chicago, IL.

[30] Centre, RIPE Network Coordination. 2012. IPv6 Measurement - A Compilation. https://labs.ripe.net/Members/mirjam/content-ipv6-measurement-compilation.

[31] Chapman, D.B. 1992. Network (In)Security Through IP Packet Filtering.

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

[33] Cheswick, B., and S. Bellovin. Firewalls and Internet Security: Repelling the Wily Hacker. 1994.

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

[35] Cho, K., M. Luckie, and B. Huffaker. 2004. Identifying IPv6 Network Problems in the Dual-stack World. In *Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality*. New York, NY.

[36] Claffy, K. 2011. Tracking IPv6 Evolution: Data We Have and Data We Need. *ACM SIGCOMM Computer Communications Review* 41(3).

[37] Colitti, L., S. Gunderson, E. Kline, and T. Fefice. 2010. Evaluating IPv6 Adoption in the Internet. In *Proceedings of the Passive and Active Measurement Conference*. Zurich, Switzerland.

[38] Cormack, G. et al. 2007. The Sixteenth Text REtrieval Conference (TREC 2007) Public Corpus. http://plg.uwaterloo.ca/~gvcormac/treccorpus07/.

[39] Cormack, G. et al. 2008. The Fifth Conference on Email and Anti-Spam (CEAS 2008) Challenge Lab Evaluation Corpus. http://plg.uwaterloo.ca/~gvcormac/ceascorpus/.

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

[41] Crocker, D. 1982. Standard for the Format of ARPA Internet Text Messages. IETF RFC 822.

[42] Dagon, D., N. Provos, C. Lee, and W. Lee. 2008. 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.

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

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

[45] Deri, L., R. Carbone, and S. Suin. 2001. Monitoring Networks Using Ntop. In *2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings*, 199–212.

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

[47] Dornbos, J. 2013. The Dornbos Spam Archive. `http://www.dornbos.com/spam01.shtml`.

[48] Drago, I., M. Mellia, M. Munafo, A. Sperotto, R. Sadre, and A. Pras. 2012. Inside Dropbox: Understanding Personal Cloud Storage Services. In *Proceedings of ACM Internet Measurement Conference (IMC '12)*. Boston, MA.

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

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

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

[52] Godwin, P. 2005. New agreement marks major milestone in IETF Administrative Restructuring. *IETF Journal* 1(2):1.

[53] Google. 2012. Protocol Buffers. `https://developers.google.com/protocol-buffers/`.

[54] Guenter, B. 2013. The Guenter Spam Archive. `http://untroubled.org/spam`.

[55] Gulbrandsen, A., P. Vixie, and L. Esibov. 2000. A DNS RR for Specifying the Location of Services (DNS SRV). IETF RFC 2782.

[56] Harrison, D., A. Ciana, A. Norberg, and G. Hazel. Tracker Peer Obfuscation. `http://www.bittorrent.org/beps/bep_0008.html`.

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

[58] J. Rosenberg, G. Camarillo A. Johnston J. Peterson R. Sparks M. Handley E. Schooler, H. Schulzrinne. 2002. SIP: Session Initiation Protocol. IETF RFC 3261.

[59] Jung, J., and E. Sit. 2004. An Empirical Study of Spam Traffic and the Use of DNS Black Lists. In *Proceedings of ACM Internet Measurement Conference*. Taormina, Italy.

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

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

[62] Karpilovsky, E., A. Gerber, D. Pei, J. Rexford, and A. Shaikh. 2009. Quantifying the Extent of IPv6 Deployment. In *Proceedings of the Passive and Active Measurement Conference*. Seoul, South Korea.

[63] Kim, H., kc claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. 2008. Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices. In *Proceedings of the 4th ACM International Conference on emerging Networking EXperiments and Technologies (ACM CoNEXT 2008)*. Madrid, Spain.

[64] Klensin, J. 2008. Simple Mail Transfer Protocol. IETF RFC 5321.

[65] Klensin, J. et al. 2001. Simple Mail Transfer Protocol. IETF RFC 2821.

[66] Labovitz, C. 2012. Six Months, Six Providers and IPv6. `http://ddos.arbornetworks.com/2011/04/six-months-six-providers-and-ipv6/`.

[67] Leiba, B. 1997. IMAP4 IDLE command. IETF RFC 2177.

[68] Liao, Q., A. Blaich, A. Striegel, and D. Thain. 2008. ENAVis: Enterprise Network Activities Visualization. In *Proceedings of the 22nd Large Installation System Administration Conference (LISA '08)*. San Diego, CA.

[69] Madhukar, A., and C. Williamson. 2006. A Longitudinal Study of P2P Traffic Classification. In *14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006*, 179–188.

[70] McCallum, C., X. Wang, and A. Corrada-Emmanuel. 2007. Topic and Role Discovery in Social Networks with Experiments on Enron and Academic Email. *Journal of Artificial Intelligence Research* 30:249–272.

[71] McCreary, S., and K. Claffy. 2000. Trends in Wide Area IP Traffic Patterns - A View From Ames Internet Exchange. CAIDA Technical Report.

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

[73] Mellia, M. 2012. Personal correspondence.

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

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

[76] Moore, A.W., and D. Zuev. 2005. Internet Traffic Classification Using Bayesian Analysis Techniques. *ACM SIGMETRICS Performance Evaluation Review* 33(1):50–60.

[77] Morris, R., D. Karger, F. Kaashoek, and H. Balakrishnan. 2001. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM 2001*. San Diego, CA.

[78] Morton, A. 2010. The Linux Kernel Mailing List Archive. `http://userweb.kernel.org/~akpm/lkml-mbox-archives/`.

[79] Pang, R., M. Allman, M. Bennett, J. Lee, and B. Tierney. 2005. A First Look at Modern Enterprise Traffic. In *Proceedings of ACM Internet Measurement Conference*. Berkeley, CA.

[80] Partridge, C. 2008. The Technical Development of Internet Email. *IEEE Annals of the History of Computing* 30(2):3–29.

[81] Paxson, V. 1999. Bro: A System for Detecting Network Intruders in Real-time. *Computer Networks* 31(23-24):2435–2463.

[82] Pelletier, R. 2008. Secretariat Transitions 1 Feb 2008. `http://www.ietf.org/mail-archive/web/ietf/current/msg50098.html`.

[83] Plonka, D. 2000. FlowScan: A Network Traffic Flow Reporting and Visualization Tool. In *Proceedings of the USENIX Fourteenth System Administration Conference (LISA XIV)*. New Orleans, LA.

[84] Plonka, D. 2000. Net::Patricia. `http://search.cpan.org/search?query=Net::Patricia`.

[85] Plonka, D., and P. Barford. 2008. Context-aware Clustering of DNS Query Traffic. In *Proceedings of the ACM SIGCOMM / USENIX Eighth Internet Measurement Conference (IMC 2008)*. Vouliagmeni, Greece.

[86] Plonka, D., and P. Barford. 2011. Flexible Traffic and Host Profiling via DNS Rendezvous. In *Proceedings of the Securing and Trusting Internet Names Workshop (SATIN 2011)*. Teddington, UK.

[87] Plonka, D., and P. Barford. 2013. Assessing Performance of Internet Services on IPv6. In *Proceedings of the Eighteenth IEEE Symposium on Computers and Communications (ISCC 2013)*. Split, Croatia.

[88] Plonka, D., and S. Campbell. 2013. The Timemail System. `http://timemail.wail.wisc.edu`.

[89] Postel, J. 1982. Simple Mail Transfer Protocol. IETF RFC 821.

[90] Ramachandran, A., and N. Feamster. 2006. Understanding the Network-Level Behavior of Spammers. In *Proceedings of ACM SIGCOMM '06*. Pisa, Italy.

[91] Ramachandran, A., N. Feamster, and D. Dagon. 2006. 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.

[92] Resnick, P. et al. 2001. Internet Message Format. IETF RFC 2822.

[93] Resnick, P. et al. 2008. Internet Message Format. IETF RFC 5322.

[94] Roesch, M. 1999. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the USENIX Thirteenth System Administration Conference (LISA XIIV)*. Seattle, WA.

[95] Romig, S., and M. Fullmer. 2000. The OSU Flow-tools Package and Cisco NetFlow Logs. In *Proceedings of the USENIX fourteenth system administration conference LISA XIV*. New Orleans, LA.

[96] Rossi, Dario, Marco Mellia, and Michela Meo. 2009. Understanding Skype Signaling. *Computer Networks: The International Journal of Computer and Telecommunications Networking* 53(2):130–140.

[97] Roughan, M., S. Sen, O. Spatscheck, and N. Duffield. 2004. 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.

[98] Sarrar, N., G.Maier, B. Ager, R. Sommer, and S. Uhlig. 2012. Investigating IPv6 Traffic - What Happened on the World IPv6 Day? In *Proceedings of the Passive and Active Measurement Conference*. Vienna, Austria.

[99] Sen, S., O. Spatscheck, and D. Wang. 2004. Accurate, Scalable In-network Identification of P2P Traffic Using Application Signatures. In *Proceedings of the 13th International Conference on World Wide Web*, 512–521. ACM New York, NY, USA.

[100] Shneiderman, B. 1992. Tree Visualization with Tree-Maps: 2-d Space-Filling Approach. *ACM Transactions on Graphics* 11(1):92–99.

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

[102] Smith, R., C. Estan, S. Jha, and S. Kong. 2008. Deflating the Big Bang: Fast and Scalable Deep Packet Inspection with Extended Finite Automata. *ACM SIGCOMM Computer Communication Review* 38(4):207–218.

[103] Smith, R. D. 2009. Toward Robust Network Payload Inspection. Ph.D. thesis, University of Wisconsin-Madison.

[104] Spring, J., and C. Huth. 2012. The Impact of Passive DNS Collection on End-user Privacy. In *Proceedings of the Securing and Trusting Internet Names Workshop (SATIN 2012)*. Teddington, UK.

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

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

[107] Theocharides, A., D. Antoniades, M. Polychronakis, E. Athanasopoulos, and E.P. Markatos. 2008. Topnet: A Network-aware top(1). In *Proceedings of the 22nd Large Installation System Administration Conference (LISA '08)*. San Diego, CA.

[108] Trestian, I., S. Ranjan, A. Kuzmanovi, and A. Nucci. 2008. Unconstrained Endpoint Profiling (Googling the Internet). In *Proceedings of ACM SIGCOMM 2008*. Seattle, WA.

[109] Tuulos, V. et al. 2005. Multi-Faceted Information Retrieval System for Large Scale Email Archives. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*. Salvador, Brazil.

[110] Walfish, M., H. Balakrishnan, and S. Shenker. 2004. Untangling the Web from DNS. In *Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*. San Francisco, CA.

[111] Wanner, R. Detecting BitTorrent Using Snort. `http://www.sans.edu/resources/student_presentations/`.

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

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

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

[115] Won, Y.J., B.C. Park, H.T. Ju, M.S. Kim, and J.W. Hong. 2006. A Hybrid Approach for Accurate Application Traffic Identification. *IEEE/IFIP E2EMON, Vancouver* 1–8.

[116] Xie, Y., F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. 2008. Spamming Botnets: Signatures and Characteristics. In *Proceedings of ACM SIGCOMM '08*. Seattle, WA.

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

[118] Zhou, X., and P.V. Mieghem. 2005. Evaluating IPv6 Adoption in the Internet. In *Proceedings of the Passive and Active Measurement Conference*. Boston, MA.