Technische Universität Berlin
Fakultät für Elektrotechnik und Informatik

Residential Broadband Internet Traffic: Characterization and Security Analysis

vorgelegt von:
Gregor M. Maier (Dipl.-Inf. Univ.)

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung der akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Volker Markl, TU Berlin
Berichter:	Prof. Anja Feldmann, Ph. D., TU Berlin
Berichter:	Prof. Vern Paxson, Ph. D., University of California, Berkeley
Berichter:	Prof. Dr. habil. Odej Kao, TU Berlin

Tag der wissenschaftlichen Aussprache: 13. Juli 2010

Berlin 2010

D 83

Ich versichere an Eides statt, dass ich diese Dissertation selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Datum Gregor M. Maier

Abstract

Residential broadband Internet connectivity is a mature and popular service in many countries. Indeed, according to the Organization for Economic Co-operation and Development (OECD), there are more than 260 million broadband customers world-wide. Understanding the nature of residential traffic characteristics is imperative for network operators to design and develop future network configurations and architectures. However, the growing world-wide user population and the introduction of new services and applications continuously changes the way users use the Internet. Furthermore, users' demands and expectations change as well. Therefore, traffic and security characteristics of residential networks have to be evaluated regularly. Yet, only few studies have examined the characteristics and security aspects of residential traffic, thus its makeup, dynamics, evolution, and variations remain underexamined.

We, in this thesis, undertake such a study. We describe observations from more than 20,000 residential DSL customers in an urban area. To ensure privacy and confidentiality, all data is immediately anonymized. Our contribution is the characterization of several different aspects of residential broadband traffic: We characterize DSL sessions, prevalence and use of network address translation (NAT), and network usage in terms of application layer protocols. Furthermore, we investigate possible performance limitations and new devices that users employ to connect to the Internet. Finally, we analyze network security, security-awareness, and risky behavior in residential networks.

DSL session characteristics, such as bandwidth utilization and online times, and NAT usage, e. g., the number of hosts connected per DSL lines, have implications for accurately provisioning access networks. Optimal access network architectures can increase customer satisfactions while decreasing complexity and cost. Likewise, the makeup of traffic, such as the application protocol mix, greatly influences the decisions of network operators and content providers on where to place popular servers and what kind of network connectivity and quality-of-service is required.

Understanding performance limitations is another critical aspect of network studies. To optimize performance and quality-of-experience, current limitations need to be known and characterized so that one can develop new protocols or tune the default settings of current protocols to achieve better performance.

In addition, the ever increasing minituarization has given rise to new classes of devices that users utilize to connect to the Internet. Mobile hand-held devices (MHDs, e. g., iPhones or BlackBerrys) are ubiquitous today. However, little is

known about how they are used—especially at home. Understanding characteristics of such novel device traffic can help network operators to anticipate future networking demands.

Furthermore, while conventional wisdom holds that residential users are responsible for much of today's Internet insecurity, few systematic studies have examined whether such views in fact reflect reality. To tackle security problems, one needs to understand the prevalence of such problems and the factors that influence security problems and malicious activity.

We, in this thesis, answer such questions. We introduce a tool that enables efficient retrospective traffic analysis. We also characterize DSL sessions and NAT usage. Surprisingly, we find that DSL-session run quite short in duration, with a median duration of only 20–30 minutes. Furthermore, we show that NAT gateways are deployed on 90 % of DSL lines and that more than 10 % of DSL lines connect multiple, concurrently active, hosts.

When we investigate application protocols, we find that HTTP dominates the application mix by volume, accounting for more than 57 % of bytes, while peer-to-peer (P2P) only contributes 14–25 %. Around the turn of the century HTTP was the dominating protocol by byte volume. The advent of P2P networks changed that and P2P dominated the protocol mix. Our study indicates that today HTTP is again on the rise, while P2P is on the decline.

To assess malicious activity, we develop a set of metrics and analyze the relationship between problems flagged by these metrics and security awareness (e. g., using anti-virus software). Furthermore, we compare our results with a rural community network in India. To our surprise, we find that both environments have similar levels of problematic behavior, in both cases indicating only a small fraction of malicious hosts. However, we also find that risky behavior is quite widespread and that security awareness steps, such as using anti-virus updates, do not correlate with a lower degree of malicious activity.

Zusammenfassung

In vielen Ländern sind Breitbandinternetanschlüsse für Privathaushalte ein populärer Dienst. Gemäß OECD gibt es weltweit mehr als 260 Millionen Breitbandkunden. Einerseits ist es für Netzbetreiber essentiell, diese Netze zu verstehen, um zukünftige Netzarchitekturen entwickeln und planen zu können. Andererseits ändert die wachsende Zahl an Internetnutzern sowie die Einführung von neuen Diensten und Anwendungen ständig die Art und Weise, in der das Internet genutzt wird. Auch die Anforderungen und Erwartungen von Nutzern ändern sich laufend. Bisher haben sich jedoch nur wenige Studien mit den Charakteristiken und Sicherheitsaspekten von Breitbandanschlüssen beschäftigt.

Das Thema dieser Dissertation ist eine derartige Studie. Wir beschreiben Messungen des Netzverkehrs von mehr als 20.000 privaten DSL-Kunden in einer Stadt. Der Datenschutz wird zu jeder Zeit sichergestellt, indem alle Daten unverzüglich anonymisiert werden. Der wissenschaftliche Beitrag dieser Dissertation ist die Charakterisierung von Breitbandinternetverkehr: Wir charakterisieren DSL-Verbindungen, die Verwendung von Network Address Translation (NAT), sowie verwendete Anwedungsprotokolle. Außerdem untersuchen wir mögliche Performanzprobleme und analysieren den Verkehr neuartiger Endgeräte (z.B. Smartphones), welche von Benutzern verwendet werden, um sich mit dem Internet zu verbinden. Die Untersuchung von Sicherheitsaspekten in unseren Netzumgebungen bildet den Abschluss dieser Arbeit.

Die Eigenschaften von DSL-Verbindungen, wie Bandbreitenausnutzung und Onlinezeiten, sowie Charakteristiken von NAT, wie die Anzahl an Computern pro DSL-Anschluss, haben Auswirkungen auf die Dimensionierung von Zugangsnetzen. Optimal dimensionierte Zugangsnetze können die Nutzerzufriedenheit steigern und gleichzeitig Komplexität und Kosten senken. Auch die Zusammensetzung des Verkehrs beeinflusst Entscheidungen von Netzbetreibern und Dienstanbietern, wie die Platzierung von populären Servern, die Art der Netzanbindung und die nötigen Quality-of-Service-Garantien.

Ein weiterer wichtiger Aspekt ist die Performanz. Nur wenn aktuelle Performanzprobleme bekannt und charakterisiert sind, können neue, bessere Protokolle entworfen werden oder Standardeinstellungen von aktuellen Protokollen so angepasst werden, dass sie optimale Performanz und Kundenzufriedenheit liefern.

Des Weiteren hat die zunehmende Miniaturisierung eine neue Art von Geräten entstehen lassen, mit denen Benutzer sich mit dem Internet verbinden. Mobile Endgeräte sind heutzutage weit verbreitet und ermöglichen es, immer “online” zu sein – egal wo man sich gerade befindet. Die Verkehrseigenschaften von solchen

Geräten sind aber bisher kaum erforscht, insbesondere wenn sie zu Hause über WiFi mit dem Internet verbunden sind. Die Kenntnis der Verkehrseigenschaften von derartigen Geräten kann Netzbetreibern helfen, zukünftige Anforderungen an ihre Netze zu erkennen.

Außerdem besagt eine weit verbreitete Ansicht, dass Privatnutzer für einen Großteil der "Unsicherheit" im Internet verantwortlich sind. Allerdings haben sich bisher nur wenige systematische Studien mit der Frage beschäftigt, ob solche Ansichten der Wahrheit entsprechen. Um Sicherheitsprobleme lösen zu können, müssen sowohl ihre Verbreitung, als auch Faktoren, die diese Probleme beeinflussen, bekannt sein.

In dieser Dissertation beantworten wir derartige Fragen. Wir präsentieren ein Softwaretool für effiziente retrospektive Verkehrsanalysen. Des Weiteren charakterisieren wir DSL-Verbindungen und die Verwendung von NAT. Zu unser Überraschung stellen wir fest, dass Onlinezeiten im Allgemeinen sehr kurz sind, was wiederum zu einem häufigem Wechsel der IP-Adressen führt. Wir zeigen auch, dass NAT-Gateways von über 90 % der DSL-Anschlüsse verwendet werden und dass an mehr als 10 % der Anschlüsse mehrere Computer gleichzeitig aktiv sind.

Wenn wir die Zusammensetzung des Netzverkehrs analysieren, stellen wir fest, dass HTTP dominiert. Über 57 % des Datenvolumens werden von HTTP verursacht, wohingegen Peer-to-Peer-Protokolle (P2P) nur 14–25 % zum Gesamtvolumen beitragen. Zur Jahrhundertwende hat HTTP dominiert. Durch die Einführung von P2P-Netzen hat sich das grundlegend verändert. Seitdem dominierte P2P. Unsere Untersuchung zeigt, dass das Pendel wieder zurück schwingt und dass HTTP-Verkehr zunimmt, während P2P-Verkehr abnimmt.

Um Sicherheitsprobleme zu finden, entwickeln wir mehrere Metriken – sowohl Signaturen von bekannter Schadsoftware, als auch verhaltensbasierte Methoden. Wir analysieren den Einfluss des Sicherheitsbewusstseins von Benutzern (z.B. Verwendung von Antivirensoftware) auf Probleme, die von diesen Metriken erkannt werden. Wir führen dieselben Analysen auch in einem Gemeinschaftsnetz in einer ländlichen Region Indiens durch. Zu unserer Überraschung stellen wir fest, dass wir in beiden Netzen eine ähnliche Menge an Sicherheitsproblemen finden – in beiden Fällen finden wir deutlich weniger infizierte Computer als man erwarten würde. Andererseits mussten wir feststellen, dass riskantes Verhalten relativ weit verbreitet ist und dass übliche Gegenmaßnahmen, wie Antivirensoftware, nicht mit einer geringeren Infektionswahrscheinlichkeit korreliert.

Pre-published Papers

Parts of this thesis are based on the following peer-reviewed papers that have already been published. All my collaborators are among my co-authors.

International Conferences

Maier, G., Schneider, F., and Feldmann, A. A first look at mobile handheld device traffic. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2010)

Maier, G., Feldmann, A., Paxson, V., and Allman, M. On dominant characteristics of residential broadband internet traffic. In *Proceedings of the Internet Measurement Conference (IMC)* (2009)

Maier, G., Sommer, R., Dreger, H., Feldmann, A., Paxson, V., and Schneider, F. Enriching network security analysis with time travel. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)* (2008)

Contents

Chapter 1

Introduction

Residential broadband Internet connectivity is a mature service in many countries. According to the Organization for Economic Co-operation and Development (OECD) there are more than 260 million broadband customers world-wide. This foundation of rich access allows users to tightly integrate network use into their lives—from checking the weather or sports scores to shopping and banking to communicating with family and friends in myriad ways. However, the nature of the connectivity differs from previously studied environments such as campus networks and enterprises in salient ways. First, users of residential broadband connections will often have different goals than those in other environments, and are not subject to the same sorts of strict acceptable use policies that may regulate their access at work or at school, such as prohibitions against accessing certain Web sites or employing certain applications. In addition, we expect that the users who set up hosts and ancillary equipment in residences often have no expertise in system administration, nor much desire to understand any more than is necessary to "make it work". Unlike for campuses (and to a lesser extent, enterprises), researchers rarely have large-scale access to residential traffic, and thus its makeup, dynamics, and variations remain underexamined. Understanding traffic characteristics and user behavior of residential networks is important to accurately design, plan, and provision future network architectures.

Furthermore, the ever increasing minituarization has given rise to new classes of devices that users utilize to connect to the Internet. Mobile hand-held devices (MHDs, e. g., iPhones or BlackBerrys) are ubiquitous today. However, little is known about how they are used—especially at home. Understanding characteristics of such novel device traffic can help network operators to anticipate future networking demands.

Finally, conventional wisdom says that residential users or users of community networks are responsible for much of today's Internet insecurity as they typically lack the means to maintain and secure their systems to the necessary degree. In particular, this must be the case for rural or developing regions, where the lack of infrastructure and technical expertise further limits the sophistication of their

protection. However, so far few systematic studies have examined whether this presumption reflects reality. In addition, no studies have examined the influence of security awareness, such as using anti-virus software or risky behavior (e. g., visiting potentially dangerous websites) on the level of malicious activity.

We, in this thesis, undertake a study to answer these questions. We base our analysis on observations of more than 20,000 residential DSL lines from a major European ISP. To ensure user privacy and confidentiality, all data is immediately anonymized. Furthermore, for some of our findings, we also compare the results from the European ISP to other network environments, including a university network, a large research lab, and a community network in rural India that connects several thousand users to the Internet.

These unique vantage points provide a broad view of Internet traffic—in particular residential Internet traffic—enabling more comprehensive and detailed characterizations than was possible in previous work, such as Cho et al.'s studies based on backbone traces [24, 25, 46], other work that examined specific applications like P2P-assisted content distribution [60] and Skype [19], or studies using active measurements [35].

1.1 Goals

A first step towards this goal is understanding DSL session and network level characteristics. We want to understand whether users are always connected to the Internet or whether they connect on-demand and then quickly disconnect once they are done. Furthermore, network address translation (NAT) is commonly used to connect to the Internet. We want to detect NAT gateways and investigate whether users connect using such gateways and if they do whether they connect only a single host or multiple hosts. Answering these questions will also shed light on how well IP addresses can be used as end-host identifiers.

For ISPs and content providers it is important to know which kind of contents and applications are popular, since different content types and application protocols require different traffic engineering approaches. We analyze the application layer protocol mix and investigate the most popular protocol, HTTP, in more depth. Next, we investigate network performance characteristics to assess whether current transport protocols and their default settings are sufficient to utilize the available network resources, or whether these settings limit the achievable quality-of-experience of users.

Finally, we examine malicious activity and risky behavior of residential users to understand whether residential users are indeed responsible for most of today's Internet insecurity. We also assess the influence of security awareness and risky behavior on malicious activity.

In summary, we, in this thesis, answer the following questions with regard to residential networks:

- How can one efficiently record, index, and retrieve network traffic over longer periods of time in order to facilitate retrospective analyses.
- Are well established properties of network traffic, such as the heavy-tailedness of flow sizes and volume per host still applicable?
- How persistent are IP addresses and over what time-scales can they be used as host (or DSL line) identifier?
- How can we detect NAT usage? What is the current deployment level of NAT and how many hosts are connected behind such a NAT gateway?
- How can one reliably identify application layer protocols? What are the popular application layer protocols used by residential customers and what are their characteristics?
- Can we rely on transport layer port numbers to classify traffic?
- What level of network performance do users experience and what are the limiting factors? To what degree are performance enhancing options (e. g., TCP window scaling) deployed?
- What is the traffic share of mobile devices and how does mobile device traffic differ from overall residential traffic?
- How can we detect hosts that are infected with malware and how prevalent are they? Do common counter-measures, such as regular software updates, or risky behavior influence the likelihood of infections?
- How does malicious activity differ between residential networks in Europe, security conscious enterprise networks, and rural community networks in developing countries?

1.2 Unexpected Results

Our study discovered a number of results we find surprising in terms of the standard "mental models" one develops from the Internet measurement and security literature and by talking with operators and colleagues. For example:

- In our residential network HTTP traffic, not peer-to-peer, dominates. Overall, HTTP makes up nearly 60 % of traffic by bytes while peer-to-peer contributes roughly 14 %. Even if we assume that all unclassified traffic is peer-to-peer, this latter figure only rises to one-quarter, confirming contemporaneous observations [41, 69, 98] from other network environments.

- DSL sessions run quite short in duration, with a median length of only 20–30 min. The short lifetime affects the rate of IP address reassignments, and we find 50 % of addresses are assigned at least twice in 24 h, and 1–5 % of addresses more than 10 times, with significant implications for IP address aliasing. NAT gateways are widely deployed and up to 49 % of DSL lines connect multiple hosts to the Internet. Furthermore, 10 % of DSL lines connect multiple *concurrently active* hosts.

- Delays experienced from a residence to the ISP's Internet gateway often exceed those over the wide-area path from the gateway to the remote peer. We find a median local component of 46 ms (due to DSL interleaving), versus a median remote component of 17 ms.

- Users rarely employ the full capacity of their lines, confirming observations by Siekkinen et al. [105]. 802.11 wireless networking in customers' homes, and TCP settings on the residential systems, appear to limit the achievable throughput.

- We find only a small fraction of actively malicious hosts both at a European ISP as well as a rural community network in India. Indeed, a comparison of these two environments shows that there is no significant difference in the levels of malicious activity.

- While OS software updates and anti-virus technology are widely deployed we do not find a correlation with a lower degree of malicious activity. Likewise, while we observe frequent risky behavior we find that such behavior does not increase the probability of malicious activity as much as one might presume.

1.3 Terminology

For clarity of exposition, we define the following terms. A *line* denotes a physical DSL line as identified by a line-card identifier. We define a DSL-level *session* as the period when the DSL modem and the line-card are together in operation. We refer to the network between the monitoring point and the customer as the *local side*, as opposed to the *remote side* (remainder of the Internet). Similarly, the customer sends *upstream* traffic and receives *downstream* traffic. A *flow* refers to unidirectional data transmission at the usual 5-tuple granularity (IP addresses, transport protocol, transport ports). A *connection* is a bi-directional transport-level communication channel, demarked for TCP by the usual control packets (SYN, FIN/RST) and for UDP by the the arrival of the first packet and the absence of activity detected using an idle timeout (20 sec). For TCP we further consider the connection as having terminated if we do not observe any packets for 180 sec. Finally, the *originator* endpoint actively initiated the connection, as opposed to the *responder*, which passively awaited the connection request.

We commonly present the distribution of measurement data as empirical probability density functions (PDFs), empirical cumulative density functions (CDFs), and empirical complimentary cumulative density functions (CCDFs). We refer to these empirical functions using the abbreviations PDF, CDF, and CCDF. PDFs are obtained by the density estimators of R and Splus.

1.4 Structure of This Thesis

The remainder of this thesis is structured as follows: In Chapter 2 we present our data sets and vantage points we use in this thesis.

In Chapter 3 we introduce the Time Machine (TM), a tool that enables efficient retrospective network traffic analysis. In many situations it can be important to archive the raw contents of a network traffic stream to disk to enable later inspection. Yet, traffic volumes can total several TB per day, rendering wholesale recording infeasible in many environments. We develop a TM for efficient recording, indexing and retrieval of historic network data, that can be used in high-throughput network environments. Using a simple heuristic we are able to reduce traffic volume by more than 90 % while still retaining over 90 % of connections in their entirety.

The thesis continues in Chapter 4 with observations from the network activity of more than 20,000 residential DSL lines. We show that online times of DSL lines

are short, resulting in high IP address churn. We also develop a novel methodology to detect network address translation (NAT) and find that 90 % of DSL lines use NAT gateways and that up to 49 % of lines connect more than one host via such NAT gateways.

To understand popular applications among the user population, we examine the application protocol mix in Chapter 5. Surprisingly, we find that HTTP accounts for more than 57 % of all bytes, with P2P only contributing about 14 % of bytes. In order to understand why HTTP is the most popular protocol we conduct a more in-depth analysis of HTTP and find that 25 % of HTTP traffic is caused by flash-video downloads (e.g., from YouTube).

In Chapter 6 we investigate performance characteristics in terms of TCP option usage, observed round-trip-times, and achievable throughput per DSL line and application. We show that DSL lines hardly ever utilize their available bandwidth, mostly due to settings on the local host.

The next part of this thesis, Chapter 7, investigates the use of mobile hand-held devices (MHDs, e.g., iPhones). MHDs are ubiquitous today and enable people to be "online" wherever they are. We investigate MHD usage when the MHD is connected via WiFi at home. The key findings are that iPhones and iPods are the most commonly observed MHDs and that the largest fractions of MHD HTTP volume are multimedia and application downloads.

The final part of this thesis, Chapter 8, analyzes malicious activity and risky behavior of residential users. We use the Time Machine to record traces of sufficient length for security analyses without exceeding the available disk space. Conventional wisdom holds that residential users are responsible for much of today's Internet insecurity. To assess malicious activity, we develop a set of metrics—including signatures for known malware and behavioral techniques—and analyze the relationship between problems flagged by these metrics and security awareness (e. g., using anti-virus software). Furthermore, we compare our results with a rural community network in India. To our surprise, we find that both environments have similar levels of problematic behavior, in both cases indicating only a small fraction of malicious hosts. However, we also find that risky behavior is quite widespread and that security awareness steps, such as using anti-virus updates, do not correlate with a lower degree of malicious activity.

Finally, we conclude this thesis in Chapter 9.

Chapter 2

Data Sets and Vantage Points

We base our study on a variety of passive network measurements. We use passive, anonymized packet-level traces and live measurements at various vantage points. Live measurements are important in order to assess performance characteristics of deployed measurement tools and security systems, such as Network Intrusion Detection Systems (NIDS). Packet traces are invaluable for performing analyses that cannot be performed using live traffic due to resource constraints. Furthermore, packet traces facilitate *reproduceability* and enable us to investigate new aspects of network traffic that only became relevant in retrospect. We first describe the different vantage points and network environments before listing the traces we use for this study.

2.1 Residential Broadband Network

Many analyses in this thesis are based on anonymized packet level traces of residential DSL connections collected at an aggregation point within a large European ISP. Overall, the ISP has roughly 10 million (4 %) of the 260 million worldwide broadband subscribers [82]. They predominantly use DSL. The access bandwidth of the monitored lines varies between 1,200/200 Kbps (downstream/upstream) and 17,000/1,200 Kbps, with the exact rate depending on both the customer's contract and their distance from the DSLAM (the ISP's line-card). In the portion of the network we monitored most users had distances low enough to in principle support 17 Mbps. The ISP does not use any traffic filter or traffic shaper.

Our monitoring vantage point allows us to observe more than 20,000 DSL lines from one urban area, connected to one access router, for which we employed Endace DAG network monitoring cards [40] for traffic capture. The data anonymization, classification, as well as application protocol specific header extraction is performed immediately on the secured measurement infrastructure using the Bro NIDS [86] with dynamic protocol detection [36]. We label traces from this ISP as ISP.

In addition, we gathered anonymized DSL session information, including the session start and end times, anonymized IP address, anonymized line-card identifier, and the configured access-bandwidth. These DSL session traces allow us to uniquely identify DSL lines as IP addresses are subject to re-assignment and churn.

2.2 AirJaldi Network in India

The AirJaldi [5] wireless network is a non-profit community network in the Himalayas of India. Using approximately 400 wireless routers, it covers a radius of 80 km in and around the city of Dharamsala. AirJaldi connects up to 10,000 users and several thousand machines with two leased lines from broadband ISPs, which provide a total uplink capacity of 10 Mbps.

The majority of the rural population accesses the Internet via publicly shared machines in cyber cafes and at Non Government Organizations (NGOs). In addition, several hundred residential users connect to the network with individually administrated machines.

AirJaldi uses a multi-tiered network address translation (NAT) architecture, where NAT gateways are connected through other NAT gateways. Due to this architecture we cannot distinguish individual end-systems at our monitoring point, which is located at the central uplink router. Our NAT detection approach, which is based on OS diversity, see Section 4.1.3, can likewise not reliably estimate the number of hosts behind such NAT gateways with many hosts.

2.3 Lawrence Berkeley National Laboratory

The *Lawrence Berkeley National Laboratory* (LBNL) is a large research institute with more than 12,000 registered hosts connected to the Internet via a 10 Gbps uplink. LBNL's traffic amounts to several TB per day. Our monitoring link here is a 10 Gbps tap into the upstream traffic. Since LBNL offers an open research environment LBNL's security policy is very liberal and defaults to unrestricted access. LBNL has however a staff security team actively monitoring its network for malicious activity. Systems found to be compromised are taken off the network immediately.

2.4 University Networks

The *Münchner Wissenschaftsnetz (Munich Scientific Research Network, MWN)* connects two major universities and affiliated research institutes to the Internet (roughly 50,000 hosts). MWN has a 10 Gbps uplink, and its traffic totals 3–6 TB/day. Since our monitoring comes from a 1 Gbps SPAN port, data rates can reach this limit during peak hours, leading to truncation. The *University of California, Berkeley* (UCB) has about 45,000 hosts. It is connected to the Internet by two 1 Gbps links and has several TB of traffic per day. As SPAN ports of the two upstream routers are aggregated into one 1 Gbps monitoring link, we can again reach capacity limits during peak times. We use these university environments only to assess the performance of the Time Machine in Chapter 3.

2.5 Anonymized Packet-Traces

For our analysis we use the following anonymized packet-level traces collected at the environments mentioned above. Table 2.1 provides an overview of the data traces from the European ISP we use for characterizing residential broadband traffic (see Chapter 4, Chapter 5, Chapter 6, and Chapter 7). The table includes the time when the trace was gathered and overall size. While we typically do not experience any packet loss, there are several multi-second periods (< 5 min overall per trace) with no packets to due OS/file-system interactions. WEEK reflects 14 different traces of 90 minutes each collected over the course of one week. These traces were gathered twice per day: during the busy-hour and during the night-time "off-hour".

In addition, we gathered anonymized DSL session information (see Table 2.2), including the session start and end times, anonymized IP address, anonymized line-card identifier, and the configured access-bandwidth. Along with DSL session traces for each of our packet measurements, we obtained a 10-day DSL session-only trace from Jan 2009 (TEN), as well as six separate 24 h session-only traces recorded every 4^{th} day in Jan–Feb 2009 (EVERY4).

For analyzing malicious activity and risky behavior, we want a longer observation period in order to be able to analyze how such activity changes over the course of several days. However, due to the high volume it is not feasible to record the full data stream for a 14-day period, and we therefore take advantage of the "Time Machine" (TM) (see Chapter 3) to reduce the stored amount of data. The Time Machine records only the first n KB of each connection and discards the remainder. We use $n = 50$ KB. Since we employ Endace monitoring cards and

Table 2.1: **Summary of anonymized packet traces for the European ISP.**

Name	Start date	Duration	Size
WEEK	Sat Aug 02, 2008 4am	14× 90 min	≈ 100–600 GB ea.
SEP08	Thu Sep 18, 2008 4am	24 h	≈ 4 TB
APR09	Wed Apr 01, 2009 2am	24 h	≈ 4 TB
AUG09a (day1)	Fri Aug 21, 2009 2am	24 h	≈ 6 TB
AUG09b (day2)	Sat Aug 22, 2009 2am	24 h	≈ 5 TB

Table 2.2: **Summary of additional anonymized DSL session information for the European ISP.**

Name	Start date	Duration	Loss
TEN	Tue Jan 08, 2009	10 days	none
EVERY4	Thu Jan 20, 2009	6× 24 h; every 4 days	none

Table 2.3: **Trace summaries for the European ISP and LBNL collected via Time Machine (TM).**

			Size		
Location	Start date	Duration	unfiltered	after TM	Loss
ISP	Sat Mar 13, 2010	14 days	≈ 88 TB	≈ 8 TB	0 %
LBNL	Thu Apr 29, 2010	4 days	≈ 24 TB	≈ 330 GB	0.2 %

Table 2.4: **Trace summaries for the Indian AirJaldi network.**

Name	Start date	# obs. IPs	Duration	Size	Loss
AirJaldi1	Wed Mar 10, 2010	≈ 260	40 h	≈ 60 GB	0.35 %
AirJaldi2	Sat Mar 13, 2010	≈ 180	34 h	≈ 30 GB	1.07 %
AirJaldi3	Thu Apr 22, 2010	≈ 260	34 h	≈ 50 GB	0.44 %

discard the bulk of the data, we do not experience any packet loss. The traces' specifics are summarized in Table 2.3.

At LBNL we use one 4 day long packet-level trace collected using the Time Machine for security analysis. The trace covers two weekdays and a weekend, with a total of 7,000 active hosts during that period. As individual large flows are common in LBNL's traffic, the Time Machine is generally able to skip a larger percentage of the total byte volume than in a typical ISP environment. In addition, the LBNL Time Machine also uses a smaller cut-off value (25 KB) than we configured at the European ISP. During our recording interval, 0.2 % of the packets were reported dropped by the capture mechanism.

Furthermore, we compare malicious activity to the rural AirJaldi community network. We gathered three full packet traces at AirJaldi in March and April 2010. The duration of each trace is between 34–40 h and we observe 180–260 distinct local IP addresses per trace. The traces for AirJaldi are summarized in Table 2.4.

Chapter 3

Enriching Network Security Analysis with Time Travel

In many situations it can be enormously helpful to archive the raw contents of a network traffic stream to disk, to enable later inspection of activity that becomes interesting only in retrospect. We present a *Time Machine (TM)* for network traffic that provides such a capability. The TM leverages the heavy-tailed nature of network flows to capture nearly all of the likely-interesting traffic while storing only a small fraction of the total volume. An initial proof-of-principle prototype established the forensic value of such an approach, contributing to the investigation of numerous attacks at a site with thousands of users. Based on these experiences, a rearchitected implementation of the system provides flexible, high-performance traffic stream capture, indexing and retrieval, including an interface between the TM and a real-time network intrusion detection system (NIDS). The NIDS controls the TM by dynamically adjusting recording parameters, instructing it to permanently store suspicious activity for offline forensics, and fetching traffic from the past for retrospective analysis. We present a detailed performance evaluation of both stand-alone and joint setups, and report on experiences with running the system live in high-volume environments.

3.1 Motivation and Background

When investigating security incidents or trouble-shooting performance problems, network packet traces—especially those with full payload content—can prove invaluable. Yet in many operational environments, wholesale recording and retention of entire data streams is infeasible. Even keeping small subsets for extended time periods has grown increasingly difficult due to ever-increasing traffic volumes. However, almost always only a very small subset of the traffic turns out to be relevant for later analysis. The key difficulty is how to decide *a priori* what data will be crucial when subsequently investigating an incident *retrospectively*.

For example, consider the Lawrence Berkeley National Laboratory (LBNL), a security-conscious research lab ($\approx$ 12,000 hosts, 10 Gbps Internet connectivity). The operational cybersecurity staff at LBNL has traditionally used bulk-recording with `tcpdump` to analyze security incidents retrospectively. However, due to the high volume of network traffic, the operators cannot record the full traffic volume, which averages 1.5 TB/day. Rather, the operators configure the tracing to omit 10 key services, including HTTP and FTP data transfers, as well as myriad high-volume hosts. Indeed, as of this writing the `tcpdump` filter contains 72 different constraints. Each of these omissions constitutes a blind spot when performing incident analysis, one very large one being the lack of records for any HTTP activity.

In this chapter we develop a system that uses dynamic packet filtering and buffering to enable effective bulk-recording of large traffic streams, coupled with interfaces that facilitate both manual (operator-driven) and automated (NIDS-driven) retrospective analysis. As this system allows us to conveniently "travel back in time," we term the capability it provides *Time Travel*, and the corresponding system a *Time Machine* (TM)[1]. The key insight is that due to the "heavy-tailed" nature of Internet traffic [84, 87], one can record most connections in their entirety, yet skip the bulk of the total volume, by only storing up to a (customizable) cutoff limit of bytes for each connection. We show that due to this property it is possible to buffer several days of raw high-volume traffic using commodity hardware and a few hundred GB of disk space, by employing a cutoff of 10–20 KB per connection—which enables retaining a *complete* record of the vast majority of connections.

Preliminary work explored the feasibility of this approach and presented a prototype system that included a simple command-line interface for queries [63]. In this chapter we build upon experiences derived from ongoing operational use at LBNL of that prototype, which led to a complete reimplementation of the system for much higher performance and support for a rich query-interface. This operational use has also proven the TM approach as an invaluable tool for network forensics: the security staff of LBNL now has access to a comprehensive view of the network's activity that has proven particularly helpful with tracking down the ever-increasing number of attacks carried out over HTTP.

At LBNL, the site's security team uses the original TM system on a daily basis to verify reports of illegitimate activity as reported by the local NIDS installation or received via communications from external sites. Depending on the type of activity under investigation, an analyst needs access to traffic from the past few

[1] For what it's worth, we came up with this name well before its use by Apple for their backup system, and it appeared in our 2005 IMC short paper [63].

hours or past few days. For example, the TM has enabled assessment of illegitimate downloads of sensitive information, web site defacements, and configuration holes exploited to spam local Wiki installations. The TM also proved crucial in illuminating a high-profile case of compromised user credentials [27] by providing evidence from the past that was otherwise unavailable.

Over the course of operating the original TM system within LBNL's production setup (and at experimental installations in two large university networks), several important limitations of the first prototype became apparent and led us to develop a new, much more efficient and feature-enhanced TM implementation that is currently running there in a prototype setup. First, while manual, analyst-driven queries to the TM for retrieving historic traffic are a crucial TM feature, the great majority of these queries are triggered by external events such as NIDS alerts. These alerts occur in significant volume, and in the original implementation each required the analyst to manually interact with the TM to extract the corresponding traffic prior to inspecting it to assess the significance of the event. This process becomes wearisome for the analyst, leading to a greater likelihood of overlooking serious incidents; the analyst chooses to focus on a small subset of alerts that *appear* to be the most relevant ones. In response to this problem, our current system offers a direct interface between the NIDS and the TM: once the NIDS reports an alert, it can ask the TM to *automatically* extract the relevant traffic, freeing the analyst of the need to translate the notification into a corresponding query.

In addition, we observed that the LBNL operators still perform their traditional bulk-recording in parallel to the TM setup,[2] as a means of enabling occasional access to more details associated with problematic connections. Our current system addresses this concern by making the TM's parameterization dynamically adaptable: for example, the NIDS can automatically instruct the redesigned TM to suspend the cutoff for hosts deemed to be malicious.

We also found that the operators often extract traffic from the TM for additional processing. For example, LBNL's analysts do this to assess the validity of NIDS notifications indicating that a connection may have leaked personally identifiable information (PII). Such an approach reflects a two-tiered strategy: first use cheap, preliminary heuristics to find a pool of possibly problematic connections, and then perform much more expensive analysis on just that pool. This becomes tenable since the volume is much smaller than that of the full traffic stream. Our current system supports such an approach by providing the means to redirect the relevant traffic *back to the NIDS*, so that the NIDS can further inspect it automatically. By coupling the two systems, we enable the NIDS to perform *retrospective* analysis.

[2] One unfortunate side-effect of this parallel setup is a significantly reduced disk budget available to the TM.

Finally, analysis of the initial TM prototype in operation uncovered a key performance challenge in structuring such a system, namely the interactions of indexing and recording packets to disk while simultaneously handling random access queries for historic traffic. Unless we carefully structure the system's implementation to accommodate these interactions, the rigorous real-time requirements of high-volume packet capture can lead to packet drops even during small processing spikes.

Our contribution is the development of a system that both supports such capture and provides the capabilities required to use it effectively in operational practice, namely dynamic configuration, and automated querying for retrospective analysis. We provide the latter in the context of interfacing the TM with the open-source Bro NIDS, and present and evaluate several scenarios for leveraging the new capability to improve the detection process.

The remainder of this chapter is structured as follows. In Section 3.2 we introduce the basic filtering structure underlying the TM. We present a design overview of the TM, including its architecture and remote control capabilities, in Section 3.3. In Section 3.4 we evaluate the performance of the TM when deployed in high-volume network environments. In Section 3.5 we couple the TM with a NIDS. We discuss deployment trade-offs in Section 3.6 and related work in Section 3.7. We finish with a summary in Section 3.8.

3.2 Exploiting Heavy-Tails

The key strategy for efficiently recording the contents of a high-volume network traffic stream comes from exploiting the heavy-tailed nature of network traffic: most network connections are quite short, with a small number of large connections (the heavy tail) accounting for the bulk of total volume [84, 87]. Thus, by recording only the first N bytes of each connection (the *cutoff*), we can record most connections in their entirety, while still greatly reducing the volume of data we must retain. For large connections, we keep only the beginning; however, for many uses the beginning of such connections is the most interesting part (containing protocol handshakes, authentication dialogs, data items names, etc.). Faced with the choice of recording some connections completely versus recording the beginning of *all* connections, we generally prefer the latter. (We discuss the evasion risk this trade-off faces, as well as mitigation strategies, in Section 3.6.)

To directly manage the resources consumed by the TM, we configure the system with disk and memory *budgets*, which set upper bounds on the volume of data retained. The TM first stores packets in a memory buffer. When the budgeted buffer fills up, the TM migrates the oldest buffered packets to disk, where they

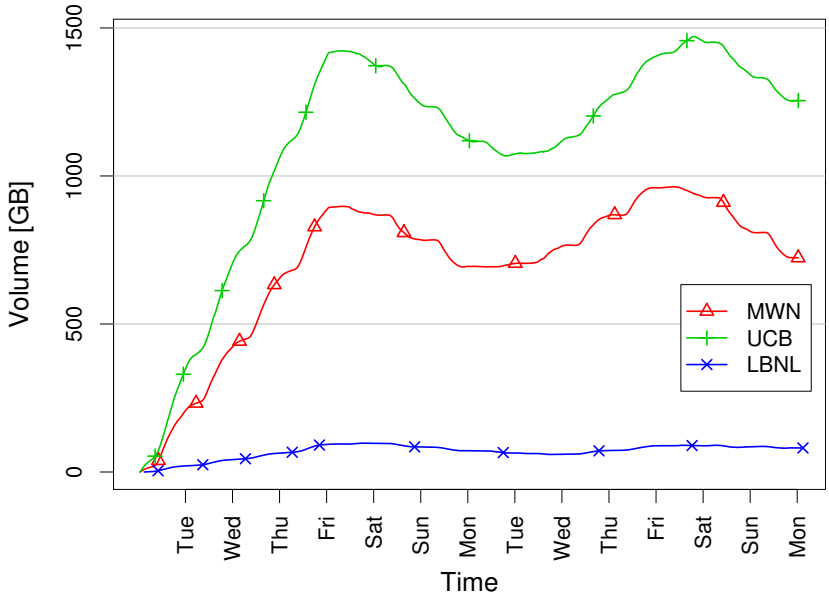

Figure 3.1: **Required Time Machine buffer size with $t_r = 4d$ and 10 KB cutoff.**

reside until the TM's total disk consumption reaches its budgeted limit. After this point, the TM begins discarding the oldest stored packets in order to stay within the budget. Thus, in steady-state the TM will consume a fixed amount of memory and disk space, operating continually (months at a time) in this fashion, with always the most recent packets available, subject to the budget constraints.

As described above, the cutoff and memory/disk budgets apply to all connections equally. However, the TM also supports defining *storage classes*, each characterized by a BPF filter expression, and applying different sets of parameters to each of these. Such classes allow, for example, traffic associated with known-suspicious hosts to be captured with a larger cutoff and retained longer (by isolating its budgeted disk space from that consumed by other traffic).

We now turn to validating the effectiveness of the cutoff-based approach in reducing the amount of data we have to store. To assess this, we use a simulation driven off connection-level traces. The traces record the start time, duration, and volume of each TCP connection seen at a given site. Such traces capture the nature of their environment in terms of traffic volume, but with much less volume than would full packet-level data, which can be difficult to record for extended periods of time.

Since we have only connection-level information for the simulation, we approximate individual packet arrivals by modeling each connection as generating packets at a constant rate over its duration, such that the total number of (maximum-sized) packets sums to the volume transferred by the connection. Clearly, this is an oversimplification in terms of packet dynamics; but because we consider traffic at very large aggregation, and at time scales of hours/days, the inaccuracies it introduces are negligible [122].

For any given cutoff N, the simulation allows us to compute the volume of packet data currently stored. We can further refine the analysis by considering a specific *retention time* t_r, defining how long we store packet data. While the TM does not itself provide direct control over retention time, with our simulation we can compute the storage the system would require (i.e., what budget we would have to give it) to achieve a retention time of at least t_r.

For our assessment, we used a set of connection-level logs gathered between November 5–18, 2007, at three institutions: Münchner Wissenschaftsnetz (Munich Scientific Research Network, MWN), Lawrence Berkeley National Laboratory (LBNL), and University of California, Berkeley (UCB). See Chapter 2 for details of these vantage points.

The connections logs contain 3120M (UCB), 1898M (MWN), and 218M (LBNL) entries respectively. The logs reveal that indeed 91–94% of all connections at the three sites are shorter than a cutoff value of $N = 10\,\text{KB}$. With a cutoff of 20 KB, we can record 94–96% of all connections in their entirety. (Of all connections, only 44–48% have any payload. Of those, a cutoff value of $N = 10\,\text{KB}$ truncates 14–19%; $N = 20\,\text{KB}$ truncates 9–13%.)

Figure 3.1 plots the disk budget required for a target retention time $t_r = 4\,\text{days}$, when employing a 10 KB cutoff. During the first 4 days we see a ramp-up phase, during which no data is evicted because the retention time t_r has not yet passed. After the ramp-up, the amount of buffer space required stabilizes, with variations stemming from diurnal patterns. For LBNL, a quite modest buffer of 100 GB suffices to retain 4 days of network packets. MWN and UCB have higher buffer requirements, but even in these high-volume environments buffer sizes of 1–1.5 TB suffice to provide days of historic network traffic, volumes within reach of commodity disk systems, and an order of magnitude less than required for the complete traffic stream.

3.3 The Time Machine Design

In this section we give an overview of the design of the TM's internals, and its query and remote-control interface, which enables coupling the TM with a real-time NIDS (Section 3.5). What we present reflects a complete reworking of the original approach framed in [63], which, with experience, we found significantly lacking in both necessary performance and operational flexibility.

3.3.1 Architecture

While in some ways the TM can be viewed as a database, it differs from conventional databases in that *(i)* data continually streams both into the system and *out* of it (expiration), *(ii)* it suffices to support a limited query language rather than full SQL, and *(iii)* it needs to observe real-time constraints in order to avoid failing to adequately process the incoming stream.

Consequently, we base the TM on the multi-threaded architecture shown in Figure 3.2. This structure can leverage multiple CPU cores to separate recording and indexing operations as well as external control interactions. The *Capture Thread* is responsible for: capturing packets off of the network tap; classifying packets; monitoring the cutoff; and assigning packets to the appropriate storage class. *Index Threads* maintain the index data to provide the *Query Threads* with the ability to efficiently locate and retrieve buffered packets, whether they reside in memory or on disk. The *Index Aggregation Thread* does additional bookkeeping on index files stored on disk (merging smaller index files into larger ones), and *User Interface Threads* handle interaction between the TM and users or remote applications like a NIDS.

Packet Capture

The *Capture Thread* uses `libpcap` to access the packets on the monitored link and potentially prefilter them. It passes the packets on to Classification.

Classification

The classification stage maps packets to *connections* by maintaining a table of all currently active connections, as identified by the usual 5-tuple. For each connection, the TM stores the number of bytes already seen. Leveraging these counters, the classification component enforces the cutoff by discarding all further packets once a connection has reached its limit. In addition to cutoff management, the classification assigns every connection to a *storage class*. A storage class defines

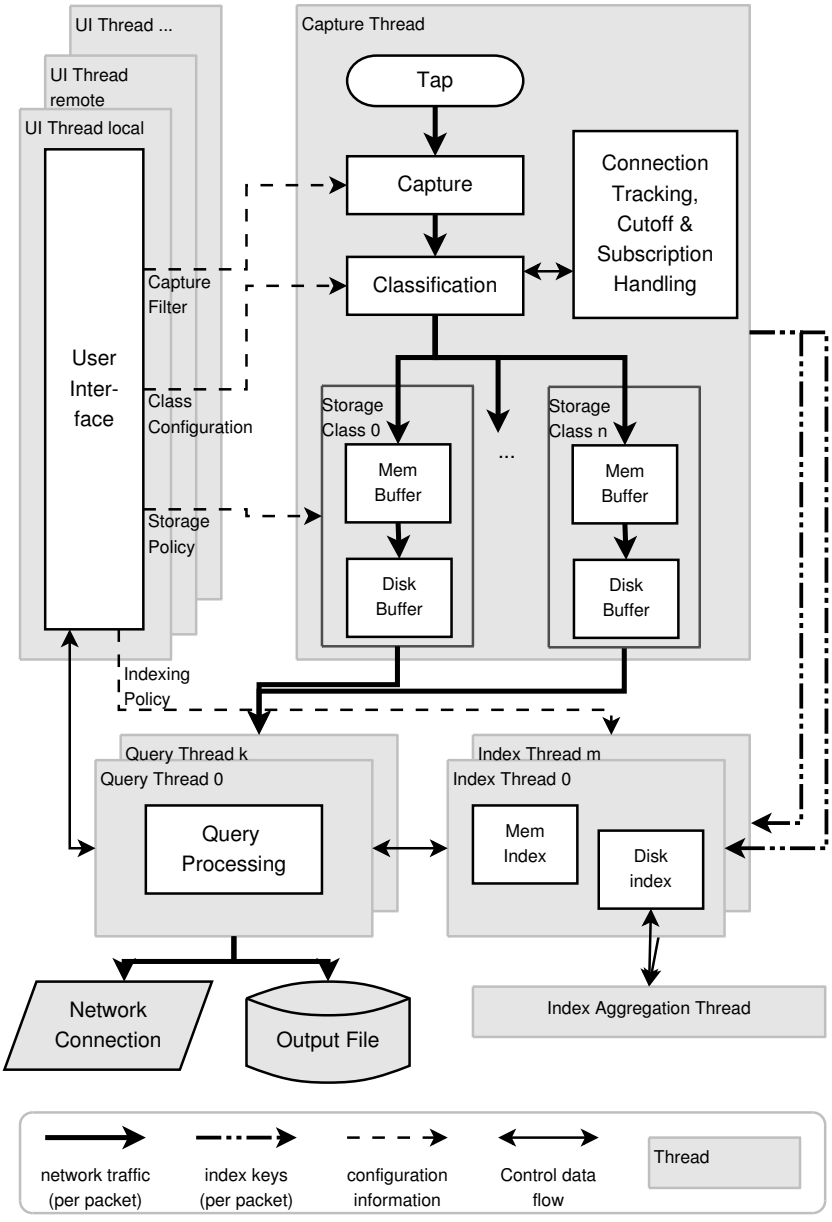

Figure 3.2: **Architecture of the Time Machine.**

```
# Query. Results are sent via network connection.
query feed nids-61367-0 tag t35654 index conn4 "tcp 1.2.3.4:42 5.6.7.8:80"
     subscribe

# In-memory query. Results are stored in a file.
query to_file "x.pcap" index ip "1.2.3.4" mem_only start 1200253074 end
     1200255474 subscribe

# Dynamic class assignment.
set_dyn_class 5.6.7.8 alarm
```

Figure 3.3: **Example query and control commands of the Time Machine.**

which TM parameters (cutoff limit and budgets of in-memory and on-disk buffers) apply to the connection's data.

Storage Classes

Each storage class consists of two buffers organized as FIFOs. One buffer is located within the main memory; the other is located on disk. The TM fills the memory buffer first. Once it becomes full, the TM migrates the oldest packets to the disk buffer. Buffering packets in main memory first allows the TM *(i)* to better tolerate bandwidth peaks by absorbing them in memory before writing data to disk, and *(ii)* to rapidly access the most recent packets for short-term queries, as we demonstrate in Section 3.5.4.

Indexing

The TM builds indexes of buffered packets to facilitate quick access to them. However, rather than referencing individual packets, the TM indexes all *time intervals* in which the associated index key has been seen on the network. Indexes can be configured for any subset of a packet's header fields, depending on what kind of queries are required. For example, setting up an index for the 2-tuple of source and destination addresses allows efficient queries for all traffic between two hosts. Indexes are stored in either main memory or on disk, depending on whether the indexed data has already been migrated to disk.

3.3.2 Control and Query Interface

The TM provides three different types of interfaces that support both queries requesting retrieval of stored packets matching certain criteria, and control of the

TM's operation by changing parameters like the cutoff limit. For interactive usage, it provides a command-line console into which an operator can directly type queries and commands. For interaction with other applications, the TM communicates via remote network connections, accepting statements in its language and returning query results. Finally, combining the two, we developed a stand-alone client-program that allows users to issue the most common kinds of queries (e.g, all traffic of a given host) by specifying them in higher-level terms.

Processing of queries proceeds as follows. Queries must relate to one of the indexes that the TM maintains. The system then looks up the query key in the appropriate index, retrieves the corresponding packet data, and delivers it to the querying application. Our system supports two delivery methods: writing requested packets to an *output file* and sending them via a *network connection* to the requester. In both cases, the TM returns the data in `libpcap` format. By default, queries span all data managed by the system, which can be quite time-consuming if the referenced packets reside on disk. The query interface thus also supports queries confined to either specific time intervals or memory-only (no disk search).

In addition to supporting queries for already-captured packets, the query issuer can also express interest in receiving *future* packets matching the search criteria (for example because the query was issued in the middle of a connection for which the remainder of the connection has now become interesting too). To handle these situations, the TM supports query *subscriptions*, which are implemented at a per-connection granularity.

Queries and control commands are both specified in the syntax of the TM's interaction language; Figure 3.3 shows several examples. The first query requests packets for the TCP connection between the specified endpoints, found using the connection four-tuple index `conn4`. The TM sends the packet stream to the receiving system `nids-61367-0` ("feed"), and includes with each packet the opaque tag `t35654` so that the recipient knows with which query to associate the packets. Finally, `subscribe` indicates that this query is a *subscription* for future packets relating to this connection, too.

The next example asks for all packets associated with the IP address `1.2.3.4` that reside in memory, instructing the TM to copy them to the local file `x.pcap`. The time interval is restricted via the `start` and `end` options. The final example changes the traffic class for any activity involving IP `5.6.7.8` to now be in the "`alarm`" class.

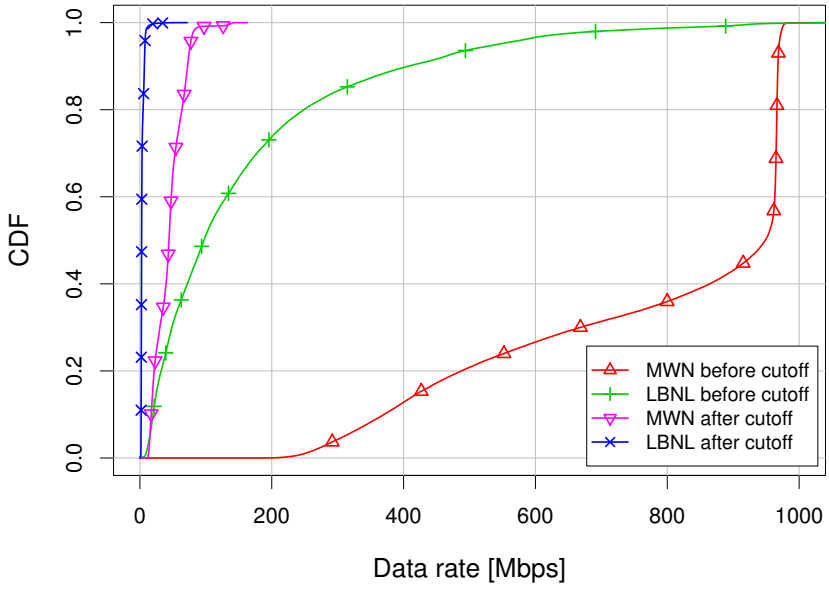

Figure 3.4: **CDF of bandwidth before/after applying a 15 KB cutoff.**

3.4 Performance Evaluation

We evaluate the performance of the TM in both controlled environments and live deployments at MWN and LBNL. The MWN deployment uses a 15 KB cutoff, a memory buffer size of 750 MB, a disk buffer size of 2.1 TB, and four different indexes (`conn4`, `conn3`, `conn2`, `ip`).[3] The TM runs on a dual-CPU AMD Opteron 244 (1.8 GHz) with 4 GB of RAM, running a 64-bit Gentoo Linux kernel (version 2.6.15.1) with a 1 Gbps Endace DAG network monitoring card [40] for traffic capture. At LBNL we use a 15 KB cutoff, 150 MB of memory, and 500 GB of disk storage, with three indexes (`conn4`, `conn3`, `ip`). The TM runs on a system with FreeBSD 6.2, two dual-core Intel Pentium D 3.7 GHz CPUs, a 3.5 TB RAID-storage system, and a Neterion 10 Gbps NIC.

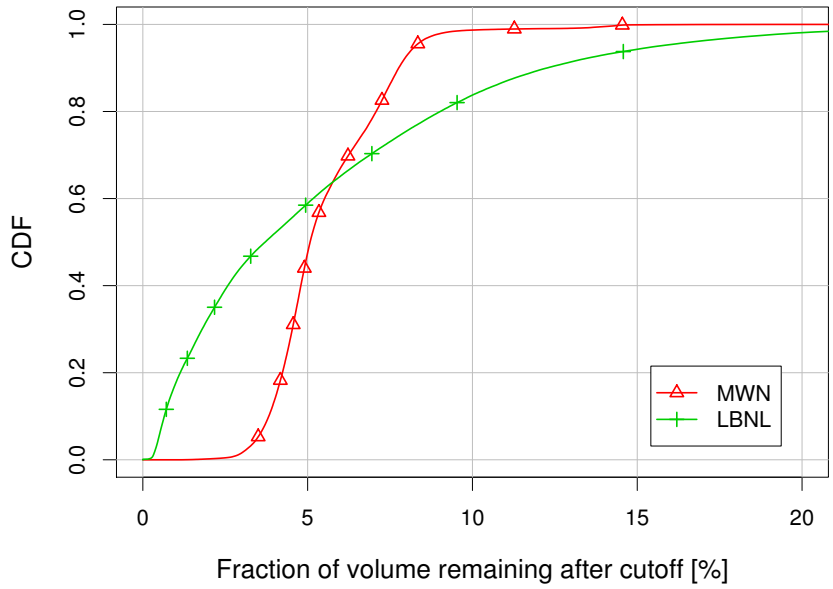

Figure 3.5: **CDF of fraction traffic remaining after applying a 15 KB cutoff.**

3.4.1 Recording

We began operation at MWN at 7 PM local time, Jan. 11, 2008, and continued for 19 days. At LBNL the measurement started at Dec. 13, 2007 at 7 AM local time and ran for 26 days. While the setup at MWN ran stand-alone, the TM at LBNL is coupled with a NIDS that sends queries and controls the TM's operation as outlined in Section 3.5.1.[4] During the measurement period, the TM setup experienced only rare packet drops. At MWN the total packet loss was less than 0.04% and at LBNL less than 0.03%. Our investigation shows that during our measurement periods these drops are most likely caused by computation spikes and scheduling artifacts, and do not in fact correlate to bandwidth peaks or variations in connection arrival rates.

We start by examining whether the cutoff indeed reduces the data volume sufficiently, as our simulation predicted. Figure 3.4 plots the original input data rates, averaged over 10 sec intervals, and the data rates after applying the cutoff for MWN and LBNL. (One can clearly see that at MWN the maximum is limited by the 1 Gbps monitoring link.) Figure 3.5 shows the fraction of traffic, the reduction rate, that remains after applying the cutoff, again averaged over 10 sec intervals. While the original data rate reaches several hundred Mbps, after the cutoff less than 6% of the original traffic remains at both sites. Hereby, the reduction rate at LBNL exhibits a higher variability. The reduction ratio shows a diurnal variation: it decreases less during daytime than during nighttime. Most likely this is due to the prevalence of interactive traffic during the day which causes short connections while bulk-transfer traffic is more prevalent during the night due to backups and mirroring.

Next, we turn to the question whether the TM has sufficient resources to leave head-room for query processing. We observe that the CPU utilization (aggregated over all CPU cores, i.e., 100% reflects saturation of all cores) measured in 10 sec intervals, shown in Figure 3.6, averages 25% (maximum $\approx$85%) for MWN indicating that there is enough head room for query processing even at peak times. For LBNL, the CPU utilization is even lower, with an average of 5% (maximum $\approx$50%). (The two local maxima for MWN in Figure 3.6 are due to the diurnal effects.)

Figure 3.7 shows how the *retention time* changes during the run at MWN. The

[3] `conn4` uses the tuple (transport protocol, ip_1, ip_2, $port_1$, $port_2$); `conn3` drops one port; `conn2` uses just the IP address pair; and `ip` a single ip address. Note, each packet leads to *two* `conn3` keys and two `ip` keys.

[4] During two time periods (one lasting 21 h, the other 4 days) the NIDS was not connected to the TM and therefore did not send any queries.

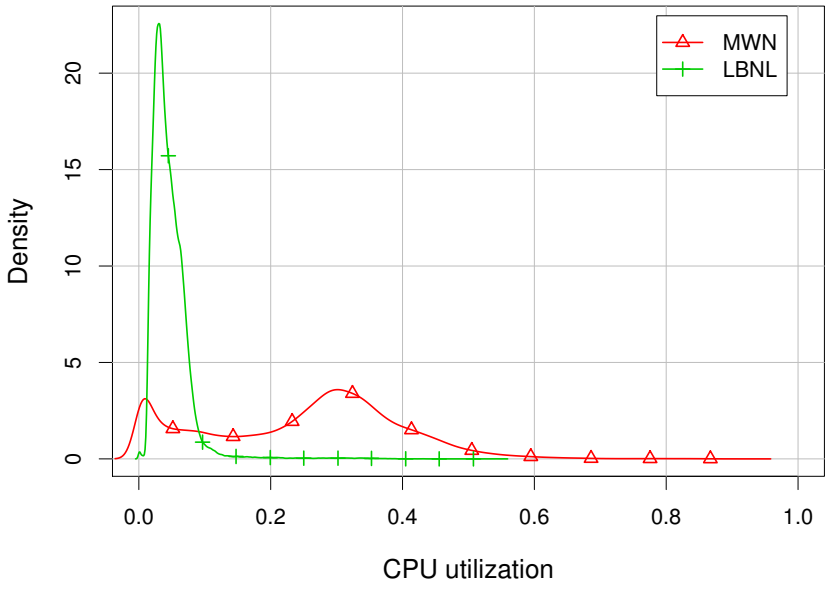

Figure 3.6: **PDF of Time Machine's CPU utilization (across all cores).**

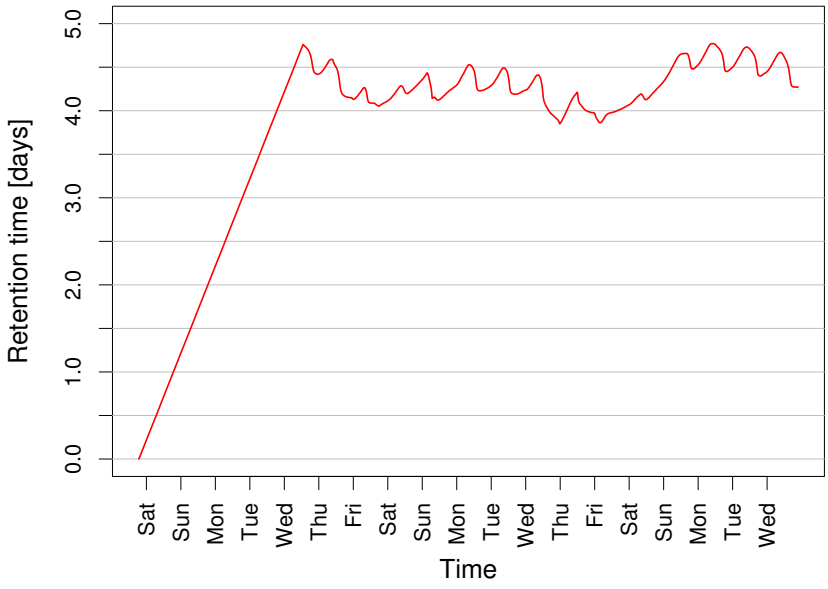

Figure 3.7: **Time Machine retention time with 2.1 TB disk buffer at MWN.**

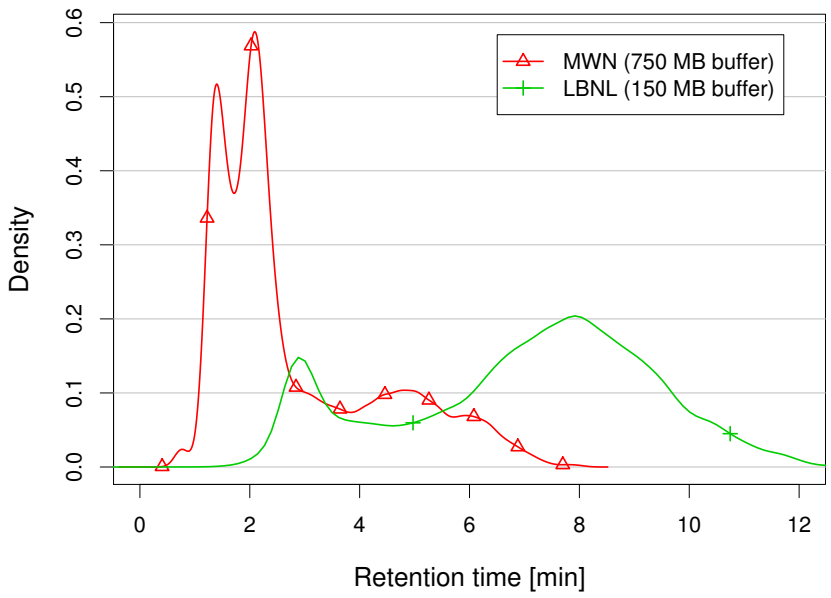

Figure 3.8: **PDF of Time Machine retention time in memory buffer.**

2.1 TB disk buffer provides ≈ 4 days during a normal work week, as one would expect given a ≈ 90% reduction in capture volume starting from 3–6 TB/day. After an initial ramp-up phase, the system retains an average of 4.3 days of network packets. As depicted in Figure 3.8, the retention time in the *memory* buffer is significantly shorter: 169 sec of network traffic on average (41 sec minimum) for MWN. The local maxima are at 84 sec, and 126 sec respectively, due to the diurnal effects. At LBNL we achieve larger retention times. The 500 GB disk buffer retained a maximum of more than 15 days, and the 150 MB memory buffer (Figure 3.8) was able to provide 421 sec on average (local maxima at 173 sec, and 475 sec).

Overall, our experience from these deployments is that the TM can satisfy queries for packets observed within the last days (weeks), providing that these are within the connection's cutoff. Moreover, the TM can answer queries for packets within the past couple of minutes very quickly as it stores these in memory.

3.4.2 Querying

As we plan to couple the TM with other applications, e.g., an intrusion detection system, that automatically generates queries it is important to understand how much load the TM can handle. Accordingly, we now examine the *query* performance of the TM with respect to *(i)* the number of queries it can handle, and *(ii)* the latency between issuing queries and receiving the corresponding replies. For these benchmarks, we ran the TM at LBNL on the same system as described above. For all experiments, we configured the TM with a memory buffer of 150 MB and a cutoff of 15 KB.

We focus our experiments on in-memory queries, since according to our experience these are the ones that are issued both at high rates and with the timeliness requirements for delivering the replies. In contrast, the execution of disk-based queries is heavily dominated by the I/O time it takes to scan the disk. They can take seconds to minutes to complete and therefore need to be limited to a very small number in any setup; we discuss this further in Section 3.6.

Load

We first examine the number of queries the TM can support. To this end, we measure the TM's ability to respond to queries that a simple benchmark client issues at increasing rates. All queries request connections for which the TM has data, so it can extract the appropriate packets and send them back in the same way as it would for an actual application.

To facilitate reproducible results, we add an *offline* mode to the TM: rather than reading live input, we preload the TM with a previously captured trace. In this mode, the TM processes the packets in the trace just as if it had seen them live, i.e., it builds up all of its internal data structures in the same manner. Once it finishes reading the trace, it only has to respond to the queries. Thus, its performance in this scenario may exceed its performance in a live setting during which it continues to capture data thus increasing its head-room for queries. (We verified that a TM operating on live traffic has head-room to sustain a reasonable query load in realistic settings, see Section 3.5.3.)

We use a 5.3 GB full trace captured at LBNL's uplink, spanning an interval of 3 min. After preloading the TM, the cutoff reduces the buffered traffic volume to 117 MB, which fits comfortably into the configured memory buffer. We configure the benchmark client to issue queries from a separate system at increasing rates: starting from one query every two seconds, the client increases the rate by 0.5 queries/sec every 10 seconds. To ensure that the client only issues requests for

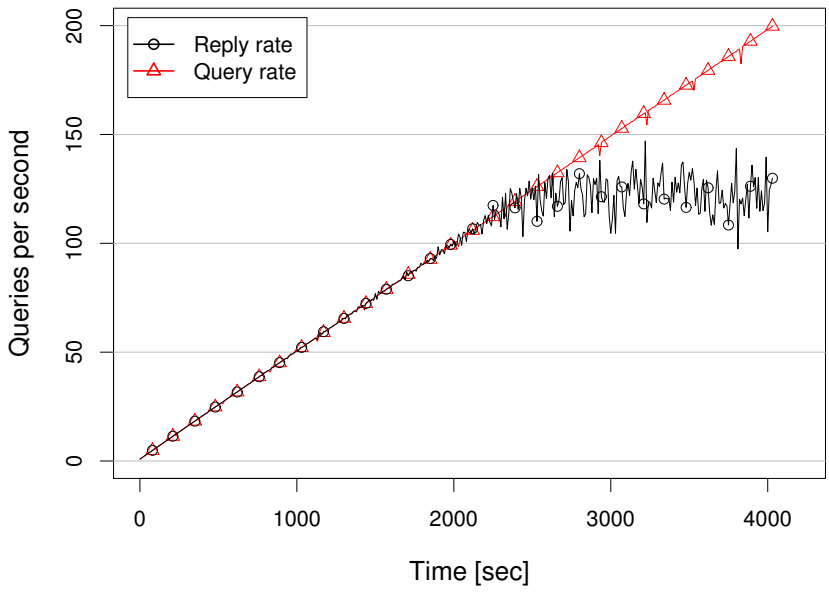

Figure 3.9: **Issuing queries to the Time Machine at increasing rates.**

packets in the TM's memory buffer, we supplied it with a sample of 1% of the connections from the input trace. Each time the client requests a connection, it randomly picks one from this list to ensure that we are not unfairly benefiting from caching.

On the TM, we log the number of queries processed per second. As long as the TM can keep up, this matches the client's query rate. Figure 3.9 plots the outcome of the experiment. Triangles show the rate at which queries were issued, and circles reflect the rate at which the TM responded, including sending the packets back to the client. We see that the TM can sustain about 120 queries/secs. Above that point, it fails to keep up. Overall, we find that the TM can handle a high query rate. Moreover, according to our experience the TM's performance suffices to cope with the number of automated queries generated by applications such as those discussed in Section 3.5.

Latency

Our next experiment examines query latency, i.e., the time between when a client issues a query and its reception of the first packet of the TM's reply. Naturally, we wish to keep the latency low, both to provide timely responses and to ensure

accessibility of the data (i.e., to avoid that the TM has expunged the data from its in-memory buffer).

To assess query latency in a realistic setting, we use the following measurement with *live* LBNL traffic. We configure a benchmark client (the Bro NIDS) on a separate system to request packets from one of every n fully-established TCP connections. For each query, we log when the client sends it and when it receives the first packet in response. We run this setup for about 100 minutes in the early afternoon of a work-day. During this period the TM processes 73 GB of network traffic of which 5.5 GB are buffered on disk at termination. The TM does not report any dropped packets. We choose $n = 100$, which results in an average of 1.3 connections being requested per second ($\sigma = 0.47$). Figure 3.10 shows the probability density of the observed query latencies. The mean latency is 125 ms, with $\sigma = 51$ ms and a maximum of 539 ms (median 143 ms). Of the 7881 queries, 1205 are answered within less than 100 ms, leading to the notable peak "(a)" in Figure 3.10. We speculate that these queries are most likely processed while the TM's capture thread is not performing any significant disk I/O (indeed, most of them occur during the initial ramp-up phase when the TM is still able to buffer the network data completely in memory). The second peak "(b)" would then indicate typical query latencies during times of disk I/O once the TM has reached a steady-state.

Overall, we conclude that the query interface is sufficiently responsive to support automatic Time Travel applications.

3.5 Coupling TM with a NIDS

Network intrusion detection systems analyze network traffic in real-time to monitor for possible attacks. While the real-time nature of such analysis provides major benefits in terms of timely detection and response, it also induces a significant constraint: the NIDS must immediately decide when it sees a network packet whether it might constitute part of an attack.

This constraint can have major implications, in that while at the time a NIDS encounters a packet its content may appear benign, *future* activity can cast a different light upon it. For example, consider a host scanning the network. Once the NIDS has detected the scanning activity, it may want to look more closely at connections originating from that source—*including those that occurred in the past*. However, any connection that took place prior to the time of detection has now been lost; the NIDS cannot afford to remember the details of everything it has

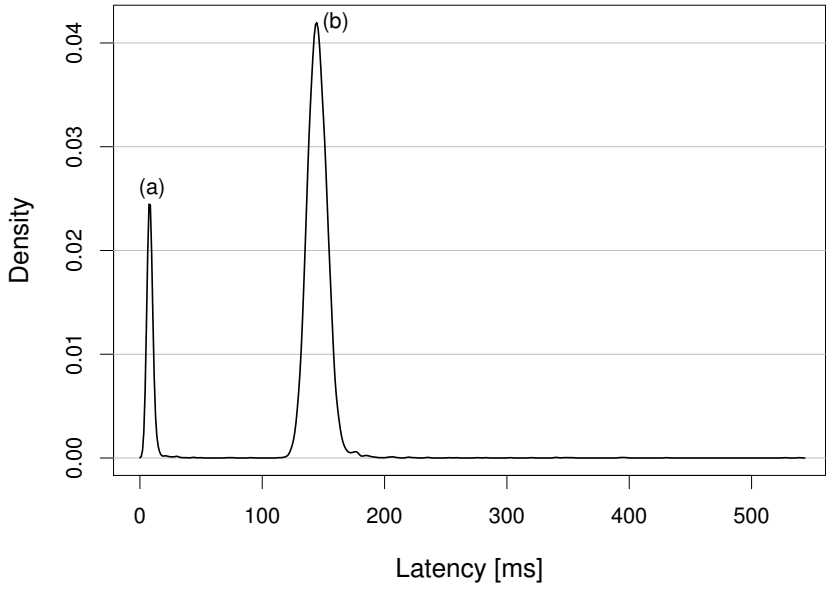

Figure 3.10: **PDF of latency between Time Machine queries and replies.**

ever seen, on the off chance that at some future point it might wish to re-inspect the activity.

The TM, on the other hand, effectively provides a very large buffer that stores network traffic in its most detailed form, i.e., as packets. By *coupling* the two systems, we allow the NIDS to access this resource pool. The NIDS can then tell the TM about the traffic it deems interesting, and in turn the TM can provide the NIDS with historic traffic for further analysis.

Given the TM capabilities developed in the previous section, we now explore the operational gains achievable by closely coupling the TM with a NIDS. We structure the discussion in five parts: *(i)* our prototype deployment at LBNL; *(ii)* experiences with enabling the NIDS to control the operation of the TM; *(iii)* the additional advantages gained if the NIDS can retrieve historic data from the TM; *(iv)* the benefits of tightly coupling the two systems; and *(v)* how we implemented these different types of functionality.

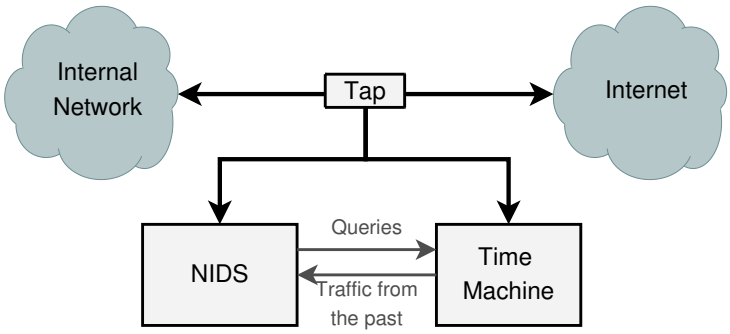

Figure 3.11: **Coupling Time Machine and NIDS at LBNL.**

3.5.1 Prototype Deployment

Figure 3.11 shows the high-level structure of coupling the TM with a NIDS. Both systems tap into the monitored traffic stream (here, a site's border) and therefore see the same traffic. The NIDS drives communication between the two, controlling the operation of the TM and issuing queries for past traffic. The TM then sends data back to the NIDS for it to analyze.

We install such a dual setup in the LBNL environment, using the open-source *Bro* NIDS [86]. Bro has been a primary component of LBNL's network monitoring infrastructure for many years, so using Bro for our study as well allows us to closely match the operational configuration.

The TM uses the same setup as described in Section 3.4: 15 KB cutoff, 500 GB disk budget, running on a system with two dual-core Pentium Ds and 4 GB of main memory. We interface the TM to the site's experimental "Bro Cluster" [119], a set of commodity PCs jointly monitoring the institute's full border traffic in a configuration that shadows the operational monitoring (along with running some additional forms of analysis). The cluster consists of 12 nodes in total, each a 3.6 GHz dual-CPU Intel Pentium D with 2 GB RAM.

We conducted initial experiments with this setup over a number of months, and in Dec. 2007 ran it continuously through early Jan. 2008 (see Section 3.4.1). The experiences reported here reflect a subsequent two-week run in Jan. 2008. During this latter period, the systems processed 22.7 TB of network data, corresponding to an average bitrate of 155 Mbps. The TM's cutoff reduced the total volume to 0.6 TB. It took a bit over 11 days until the TM exhausted its 500 GB disk budget

for the first time and started to expire data. The NIDS reported 66,000 operator-level notifications according to the configured policy, with 98% of them referring to scanning activity.

3.5.2 NIDS Controls the Time Machine

The TM provides a network-accessible control interface that the NIDS can use to dynamically change operating parameters based on its analysis results such as cutoffs, buffer budgets, and timeouts. In our installation, we instrument the NIDS so that for every operator notification[5], it instructs the TM to *(i)* disable the cutoff for the affected connection for non-scan notifications, and *(ii)* change the storage class of the IP address the attacker is coming from to a more conservative set of parameters (higher cutoffs, longer timeouts), and also assign it to separate memory and buffer pools. The latter significantly increases the retention time for the host's activity, as it now no longer shares its buffer space with the much more populous benign traffic.

In concrete terms, we introduce two new TM storage classes: *scanners*, for hosts identified as scanners, and *alarms*, for hosts triggering operator notifications other than scan reports. The motivation for this separation is the predominance of Internet-wide scanning: in many environments, scanning alerts heavily dominate the reporting. By creating a separate buffer for scanners, we increase the retention time for notifications not related to such activity, which are likely to be more valuable. The classes *scanners* and *alarms* are provided with a memory budget of 75 MB and a disk budget of 50 GB each. For scanners, we increase the cutoff from 15 KB to 50 KB; for all other offenders we disable the cutoff altogether. Now, whenever the NIDS reports an operator notification, it first sends a `suspend_cutoff` command for the triggering connection to the TM. It then issues a `set_class` command for the offending host, putting the address into either *scanners* or *alarms*.

Examining the commands issued by the NIDS during the two-week period, we find that it sent 427 commands to suspend the cutoff for individual connections. Moreover, it moved 12,532 IP addresses into the *scanners* storage class and 592 into the *alarms* storage class.[6]

[5] We note that the specifics of what constitutes an operator notification vary from site to site, but because we cannot report details of LBNL's operational policy we will refer only to broad *classes* of notifications such as "scans".

[6] We note that the number of issued commands does not directly correspond to the number of operator notifications generated by the NIDS. The NIDS often reports hosts and connections multiple times, but only sends the corresponding command once. Furthermore, the NIDS sometimes issues commands to change the storage class for activity which does not generate a notification.

3.5.3 NIDS Retrieves Data from Time Machine

Another building block for better forensics support is automatic preservation of incident-related traffic. For all operator notifications in our installation, the NIDS queries the TM for the relevant packets, which are then permanently stored for later inspection.

Storage

The NIDS issues up to three queries for each major (non-scan) notification. Two `to_file` queries instruct the TM to store *(i)* all packets of the relevant connection and *(ii)* all packets involving the offender's IP address within the preceding hour. For TCP traffic, the NIDS issues a `feed` query asking it to also *return* the connection's packets to the NIDS. The NIDS then stores the *reassembled* payload stream on disk. For many application protocols, this eases subsequent manual inspection of the activity. We restrict connection queries to in-memory data, while host queries include disk-buffered traffic as well. Our motivation is that connection queries are time-critical while host queries are related to forensics.

During the examined two-week period, the NIDS issued queries for 427 connections (after duplicate elimination) and 376 individual hosts. As queries for connections were limited to in-memory data, their mean processing time was 210 ms ($\sigma = 510$ ms). Among the queries, there was one strong outlier that took 10.74 sec to complete: it yielded 299,002 packets in response. Manual inspection of the extracted traffic showed that this was a large DNS session. Excluding this query, the mean time was 190 ms ($\sigma = 100$ ms). Queries for individual hosts included on-disk data as well, and therefore took significantly longer; 25.7 sec on average. Their processing times also varied more (median 10.2 sec, $\sigma = 54.1$ sec).

Interactive Access

To further reduce the turnaround time between receiving a NIDS notification and inspecting the relevant traffic, we developed a Web-based interface that enables browsing of the data associated with each notification; Figure 3.12 shows a snapshot. The prototype interface presents the list of notifications and indicates which kind of automatically extracted TM traffic is available. The operator can then inspect relevant packets and payload using a browser, including traffic that occurred prior to the notification.

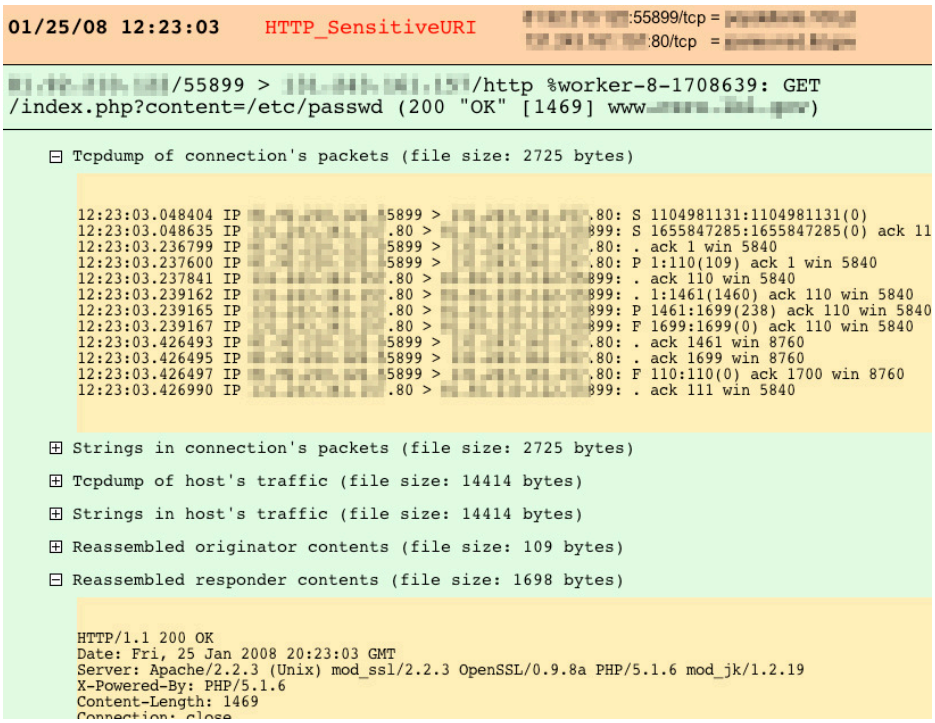

Figure 3.12: **Web-interface to security notifications and their corresponding network traffic (packets and payload).**

Experiences

We have been running the joint TM/NIDS setup at LBNL for two months, and have used the system to both analyze packet traces and reassembled payload streams for more detailed analysis. During this time, the TM has proven to be extremely useful. First, one often just cannot reliably tell the impact of a specific notification without having the actual traffic at hand. Second, it turns out to be an enormous timesaver to always have the traffic related to a notification available for *immediate* analysis. This allows the operator to inspect a significantly larger number of cases in depth than would otherwise be possible, even those that appear to be minor on first sight. Since with the TM/NIDS setup double-checking even likely false-positives comes nearly for free, the overall quality of the security monitoring can be significantly improved.

Our experience from the deployment confirms the utility of such a setup in several ways. First, the TM enables us to assess whether an attack succeeded. For example, a still very common attack includes probing web servers for vulnerabilities. Consider Web requests of the form `foo.php?arg=../../../etc/passwd` with which the attacker tries to trick a CGI script into returning a list of passwords. Since many attackers scan the Internet for vulnerable servers, simply flagging such requests generates a large number false positives, since they very rarely succeed. If the NIDS reports the server's response code, the operator can quickly weed out the cases where the server just returned an error message. However, even when the server returns an *200 OK*, this does *not* necessarily indicate a successful attack. Often the response is instead a generic, harmless page (e.g., nicely formatted HTML explaining that the request was invalid). Since the TM provides the served web page in its raw form, we can now quickly eliminate these as well. To further automate this analysis, we plan to extend the setup so that the NIDS *itself* checks the TM's response for signs of an actual password list, and suppresses the notification unless it sees one. Similar approaches are applicable to a wide range of probing attacks.

For applications running on non-standard ports the TM has the potential to significantly help with weeding out false-positives. Bro, for example, flags outgoing packets with a destination port 69/udp as potential "Outbound TFTP" (it does not currently include a TFTP protocol analyzer). Assessing the significance of this notification requires looking at the payload. With the TM recordings we were able to quickly identify in several instances that the reported connection reflected BitTorrent traffic rather than TFTP. In another case, Bro reported parsing errors for IRC traffic on 6667/tcp; inspection of the payload quickly revealed that a custom protocol was using the port.

```
XXX.XXX.XXX.XXX/57340 > XXX.XXX.XXX.XXX/smtp same gap on
    link/time-machine (> 124/6296)
XXX.XXX.XXX.XXX/55529 > XXX.XXX.XXX.XXX/spop same gap on
    link/time-machine (> 275/165)
XXX.XXX.XXX.XXX/2050  > XXX.XXX.XXX.XXX/pop-3 same gap on
    link/time-machine (> 17/14)
```

Figure 3.13: **Example of drops confirmed by the Time Machine.**

The information captured by the TM can also shed light on how attacks work. In one instance, a local client downloaded a trojan via HTTP. The NIDS reported the fact and instructed the TM to return the corresponding traffic. Once the NIDS had reassembled the payload stream, the trojan's binary code was available on disk for further manual inspection (though truncated at the 15 KB cutoff).

Finally, the TM facilitates the extraction of packet traces for various interesting network situations, even those not necessarily reflecting attacks. Among others, we collected traces of TCP connections opened simultaneously by both sides; sudden FIN storms of apparently misconfigured clients; and packets that triggered inaccuracies in Bro's protocol processing.

3.5.4 Retrospective Analysis

In the following, we demonstrate the potential of a tighter integration of TM and NIDS by examining forms of *retrospective analysis* this enables.

Recovering from Packet Drops

Under heavy load, a NIDS can lack the processing power to capture and analyze the full packet stream, in which case it will incur *measurement drops* [37]. Working in conjunction with the TM, however, a NIDS can query for connections that are missing packets and reprocess them. If the same gap also occurs in the response received from the TM, the NIDS knows that most likely the problem arose external to the NIDS device (e.g., in an optical tap shared by the two systems, or due to asymmetric routing).

We implemented this recovery scheme for the Bro NIDS. With TCP connections, Bro infers a packet missing if it observes a sequence gap purportedly covered by a TCP acknowledgment. In such cases we modified Bro to request the affected connection from the TM. If the TM connection is complete, Bro has recovered from the gap and proceeds with its analysis. If the TM connection is however also

missing the packet, Bro generates a notification (see Figure 3.13). In addition to allowing Bro to correctly analyze the traffic that it missed, this also enables Bro to differentiate between drops due to overload and packets indeed missing on the link.

Offloading the NIDS

NIDS face fundamental trade-offs between depth of analysis and resource usage [107]. In a high-volume environment, the operator must often choose to forego classes of analysis due to limited processing power. However, by drawing upon the TM, a NIDS can make fine-grained exceptions to what would otherwise be analysis omissions. It does so by requesting initially excluded data once the NIDS recognizes its relevance because of some related analysis that is still enabled.

For example, the bulk of HTTP traffic volume in general originates from HTTP servers, rather than clients. Thus, we can significantly offload a NIDS by restricting its analysis to client-side traffic, i.e., only examine URLs and headers in browser requests, but not the headers and items in server replies. However, once the NIDS observes a suspicious request, it can query the TM for the complete HTTP connection, which it then analyzes with full server-side analysis. The benefit of this setup is that the NIDS can now save significant CPU time as compared to analyzing *all* HTTP connections, yet sacrificing little in the way of detection quality.

FTP data transfers and portmapper activity provide similar examples. Both of these involve dynamically negotiated secondary connections, which the NIDS can discern by analyzing the (lightweight) setup activity. However, because these connections can appear on arbitrary ports, the NIDS can only inspect them directly if it foregoes port-level packet filtering. With the TM, however, the NIDS can request subscriptions (Section 3.3.2) to the secondary connections and inspect them in full, optionally also removing the cutoff if it wishes to ensure that it sees the entire contents.

We explore the HTTP scenario in more detail to understand the degree to which a NIDS benefits from offloading some of its processing to the TM. For our assessment, we need to compare two different NIDS configurations (with and without the TM) while processing the same input. Thus, we employ a trace-based evaluation using a 75 min full-HTTP trace captured on LBNL's upstream link (21 GB; 900,000 HTTP sessions), using a two-machine setup similar to that in Section 3.4.2. The evaluation requires care since the setup involves communication with the TM: when working offline on a trace, both the NIDS and the TM can process their input more quickly than real-time, i.e., they can consume 1 sec worth of measured traffic in less than 1 sec of execution time. However, the NIDS and the TM differ in the

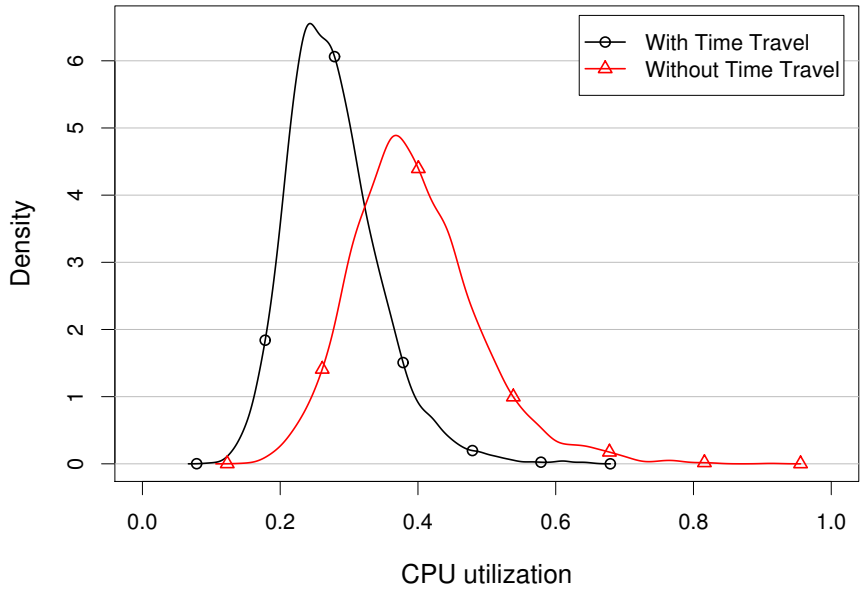

Figure 3.14: **PDF of Bro's CPU load with and without Time Travel.**

rate at which they outpace network-time, which can lead to a desynchronization between them.

To address these issues, the Bro system provides a *pseudo-realtime* mode [108]: when enabled, it inserts delays into its execution to match the inter-packet gaps observed in a trace. When using this mode, Bro issues queries at the same time intervals as it would during live execution. Our TM implementation does not provide a similar facility. However, for this evaluation we wish to assess the NIDS's operation, rather than the TM's, and it therefore suffices to ensure that the TM correctly replies to all queries. To achieve this, we preload the TM with just the relevant subset of the trace, i.e., the small fraction of the traffic that the Bro NIDS will request from the TM. The key for preloading the TM is predicting which connections the NIDS will request. While in practice the NIDS would trigger HTTP-related queries based on URL patterns, for our evaluation we use an approach independent of a specific detection mechanism: Bro requests each HTTP connection with a small, fixed probability p.

Our first experiment measures the performance of a stand-alone NIDS. We configure Bro to perform full HTTP processing. To achieve a fair comparison, we modify Bro to ignore all server payload after the first 15 KB of each connection, simulating the TM's cutoff. We then run Bro in pseudo-realtime mode on the trace and log

the CPU usage for each 1 sec interval. Figure 3.14 shows the resulting probability density.

With the baseline established, we then examine the TM/NIDS hybrid. We configure Bro to use the same configuration as in the previous experiment, except with HTTP response processing disabled. Instead, we configure Bro to issue queries to the TM for a pre-computed subset of the HTTP sessions for complete analysis. We choose $p = 0.01$, a value that from our experience requests full analysis for many more connections than a scheme based on patterns of suspicious URLs would. We supply Bro with a prefiltered version of the full HTTP trace with all server-side HTTP payload packets excluded.[7] As described above, we provide the TM with the traffic which Bro will request.

We verify that the TM/NIDS system matches the results of the stand-alone setup. However, Figure 3.14 shows a significant reduction in CPU load. In the stand-alone setup, the mean per-second CPU load runs around 40% ($\sigma = 9\%$). With TM offloading, the mean CPU load decreases to 28%, ($\sigma = 7\%$). We conclude that offloading indeed achieves a significant reduction in CPU utilization.

Broadening the analysis context

Finally, with a TM a NIDS can request historic network traffic, allowing it to perform analysis on past traffic within a context not available when the traffic originally appeared. For example, once the NIDS identifies a source as a scanner, it is prudent to examine *all* of its traffic in-depth, including its previous activity. The same holds for a local host that shows signs of a possible compromise. Such an in-depth analysis may for example include analyzers that were previously disabled due to their performance overhead. In this way the NIDS can construct for the analyst a detailed application-level record of the offender, or the NIDS might itself assess this broader record against a meta-policy to determine whether the larger view merits an operator notification.

3.5.5 Implementing Retrospective Analysis

Implementing the TM/NIDS interface for the above experiments requires solving a number of problems. The main challenge lies in that processing traffic from the past, rather than freshly captured, violates a number of assumptions a NIDS

[7] We prefilter the trace, rather than installing a Bro-level BPF filter, because in a live setting the filtering is done by the kernel, and thus not accounted towards the CPU usage of the Bro process.

typically makes about packets appearing in real-time with a causal order reflecting a monotonic passage of time.

A simple option is to special-case the analysis of resurrected packets by introducing a second data path into the NIDS exclusively dedicated to examining TM responses. However, such an approach severely limits the power of the hybrid system, as we in this case cannot leverage the extensive set of tools the NIDS already provides for live processing. For example, offloading applications, as described in Section 3.5.4, would be impossible to realize without duplicating much of the existing code. Therefore, our main design objective for our Bro implementation is to process all TM-provided traffic inside the NIDS's *standard* processing path, the same as for any live traffic—and in parallel with live traffic. In the remainder of this section, we discuss the issues that arose when adding such a TM interface to the Bro NIDS.

Bro Implementation

Bro provides an extensive, domain-specific scripting language. We extend the language with a set of predefined functions to control and query the TM, mirroring the functionality accessible via the TM's remote interface (see Section 3.3.2), such as changing the TM class associated with a suspect IP address, or querying for packets based on IP addresses or connection 4-tuples. One basic requirement for this is that the interface to the TM operates asynchronously, i.e., Bro must not block waiting for a response.

Sending commands to the TM is straight-forward and thus omitted. Receiving packets from the TM for processing, however, raises subtle implementation issues: the timestamp to associate with received query packets, and how to process them if they are replicates of ones the NIDS has already processed due to direct capture from the network, or because the same packet matches multiple streams returned for several different concurrent queries.

Regarding timestamps, retrieved packets include the time when the TM recorded them. However, this time is in the past and if the NIDS uses it directly, confusion arises due to its assumptions regarding time monotonicity. For example, Bro derives its measure of time from the timestamps of the captured packets. For example it uses these timestamps to compute timer expirations and to manage state. The simple solution of rewriting the timestamps to reflect the current time confounds any analysis that relies on either absolute time or on relative time between multiple connections. Such an approach also has the potential to confuse the analyst that inspects any timestamped or logged information.

The key insight for our solution, which enables us to integrate the TM interface into Bro with minimal surgery, is to restrict Bro to always request *complete connections* from the TM rather than individual packets. Such a constraint is tenable because, like all major NIDS, connections form Bro's main unit of analysis.

We implement this constraint by ensuring that Bro only issues queries in one of two forms: *(i)* for all packets with the same 4-tuple $(\text{address}_1, \text{port}_1, \text{address}_2, \text{port}_2)$, or *(ii)* for all packets involving a particular address. In addition, to ensure that Bro receives all packets for these connections, including future ones, it *subscribes* to the query (see Section 3.3.2).

Relying on complete connections simplifies the problem of timestamps by allowing us to introduce the use of *per-query network times*: for each TM query, Bro tracks the most recently received packet in response to the query and then maintains separate per-query *timelines* to drive the management of any timer whose instantiation stems from a retrieved packet. Thus, TM packets do not perturb Bro's global timeline (which it continues to derive from the timestamps of packets in its direct input stream).

We also rely on complete connections to address the issue of replicated input. When retrieved packets for a connection begin to arrive while Bro is processing the same connection via its live feed, it discards the live version and starts afresh with the TM version. (It also discards any future live packets for such connections, since these will arrive via its TM subscription.) Moreover, if Bro is processing packets of a connection via the TM and then receives packets for this same connection via its live feed (unlikely, but not impossible if the system's packet capturing uses large buffers), then Bro again ignores the live version. Finally, if Bro receives a connection multiple times from the TM (e.g., because of multiple matching queries), it only analyzes the first instance.

Our modifications to Bro provide the NIDS with a powerful interface to the TM that supports forensics as well as automatic, retrospective analysis. The additions introduce minimal overhead, and have no impact on Bro's performance when it runs without a TM.

3.6 Deployment Trade-Offs

In an actual deployment, the TM operator faces several trade-offs in terms of CPU, memory, and disk requirements. The most obvious trade-off is the design decision of foregoing complete storage of high-volume connections in order to reduce memory/disk consumption. There are others as well, however.

Risk of Evasion

The TM's cutoff mechanism faces an obvious risk for evasion: if an attacker delays his attack to occur after the cutoff, the TM will not record the malicious actions. This is a fundamental limitation of our approach. However, short of comprehensively storing *all* packets, any volume reduction heuristic faces such a blind spot.

The cutoff evasion problem is similar in risks to the problem NIDS face when relying on timeouts for state management. If a multi-step attack is stretched over a long enough time period such that the NIDS is forced to expire its state in the interim the attack can go undetected. Yet, to avoid memory exhaustion state must be expired eventually. Therefore, NIDS rely on the fact that an attacker cannot predict when *exactly* a timeout will take place [37].

Similarly, the TM has several ways for reducing the risk of evasion by making the cutoff mechanism less predictable. One approach is to use different storage classes (see Section 3.3.1) with different cutoffs for different types of traffic, e.g., based on applications (for some services, delaying an attack to later stages of a session is harder than for others). As discussed in Section 3.5.2, we can also leverage a NIDS's risk assessment to dynamically adjust the cutoff for traffic found more likely to pose a threat. Finally, we plan to examine *randomizing* the cutoff so that *(i)* an attacker cannot predict at which point it will go into effect, and *(ii)* even when the cutoff has been triggered, the TM may continue recording a random subset of subsequent packets.

Network Load

When running in high-volume 10 Gbps environments, the TM can exceed the limits of what commodity hardware can support in terms of packet-capture and disk utilization. We can alleviate this impact with use of more expensive, special-purpose hardware (such as the Endace monitoring card at MWN), but at added cost and for limited benefit. We note, however, that the TM is well-suited for *clustering* in the same way as a NIDS [119]: we can deploy a set of PCs, each running a separate TM on a slice of the total traffic. In such a distributed setting, an additional front-end system can create the impression to the user of interacting with a single TM by relaying to/from all backend TMs.

Floods

Another trade-off concerns packet floods, such as encountered during high-volume DoS attacks. Distributed floods stress the TM's connection-handling, and can thus undermine the capture of useful traffic. For example, during normal operation

at MWN an average of 500,000 connections are active and stored in the TM's connection table. However, we have experienced floods during which the number of connections increased to 3–4 million within 30 seconds. Tracking these induced massive packet drops and eventually exhausted the machine's physical memory.

In addition, adversaries could attack the TM directly by exploiting its specific mechanisms. They could for example generate large numbers of small connections in order to significantly reduce retention time. However, such attacks require the attacker to commit significant resources, which, like other floods, will render them vulnerable to detection.

To mitigate the impact of floods on the TM's processing, we plan to augment the TM with a flood detection and mitigation mechanism. For example, the system can probabilistically track per-source thresholds of connection attempts and resort to address-specific packet sampling once a source exceeds these. Alternatively, when operating in conjunction with a NIDS that includes a flood detection mechanism, the TM can rely upon the NIDS to decide when and how the TM should react.

Retrieval Time

When running a joint TM/NIDS setup, we need to consider a trade-off between the response time for answering a query versus the time range that the TM examines to find the relevant packets. As discussed in Section 3.4.2, the TM can answer queries quite quickly as long as it restricts its retrieval to in-memory data. However, once the TM needs to search its disk, queries can take seconds to minutes, even if they include a time interval to limit the scope of the search. Thus, the NIDS must issue such queries carefully so as to not exhaust the TM's resources. We do not yet have enough long-term operational experience to have determined good rules for how to manage such queries, but this is part of the near-term focus of our future work.

NIDS and Cutoff

A final issue concerns the impact of the TM's cutoff on the NIDS processing. In our NIDS implementation, we minimize the limitations resulting from the cutoff by combining each NIDS query with a request to remove the cutoff for the associated connections or addresses. This takes care of future activity via the TM. But there is little we can do about connections curtailed in the past—those for which the TM already applied the cutoff. Recall that the general premise of the TM is that we usually can operate the TM with a sufficiently large cutoff that information of interest is captured. To further reduce this problem the TM always stores all TCP control packets (SYN/FIN/RST), thus enabling a NIDS to perform its connection-level analysis.

3.7 Related Work

We develop a "Time Machine" for efficient network packet recording and retrieval, and couple the resulting system with a NIDS. The basic approach is to leverage the heavy-tailed nature of Internet traffic [84, 87] to significantly reduce the volume of bulk traffic recording. Our work builds on the proof-of-principle prototype described by Kornexl et al. [63], greatly increasing its performance and coupling it to external systems to support automated querying, live connections to a NIDS, and "subscriptions" to future packets satisfying a given query. These features have enabled us to then use the system in conjunction with the Bro NIDS in an operational high-volume setting.

Different approaches have been suggested in the literature to record high-volume network traffic. First, several systems aim to record full packet traces: Anderson et al. [8] records at kernel-level to provide bulk capture at high rates, and Antonelli et al. [10] focuses on long-term archive and stores traffic on *tapes*. However, these systems do not provide automated and efficient real-time query interfaces. Hyperion [34] employs a dedicated stream file system to store high-volume data streams, indexing stream data using Bloom filters. Hyperion bulk-records entire traffic streams, and does not a provide features for automatic or semi-automatic forensics nor coupling with a NIDS. Gigascope [31], on the other hand, supports SQL-like queries on a packet stream, but no long-term archiving. In 2007 Schneider et al. [101] analyze traffic capture performance of different commodity computer and operating systems and found that AMD Opterons with FreeBSD outperforms all other OS/CPU combinations.

Another approach is to store higher-level abstractions of the network traffic to reduce the data volume: Reiss et al. [95] record flows and provide real-time query facilities; Shanmugasundaram et al. [104] record key events of activity such as connections and scans; and [23, 76] both provide frameworks suitable for performing different data reduction techniques. ([76] is based on the CoMo platform [28]). Cooke et al. [30] aggregate data as it ages: first packets are stored; these are than transformed into flows. They focus on storage management algorithms to divide storage between the different aggregation levels. Ponec et al. [89] store traffic digests for payload attribution; the queries do not yield the actual content. Any data reduction decreases the amount of information available. We argue that for security applications, the TM's approach of archiving the head of connections at the packet-level provides an attractive degree of detail compared to such abstractions.

Reducing traffic volume by omitting parts of the traffic is employed by the Shunt [49]. The Shunt is a programmable NIC that an associated NIDS/NIPS instructs to for-

ward, drop, or *divert* packets at a per-connection granularity. The drop functionality supports intrusion prevention; the forward functionality supports offloading the NIDS for streams it has determined it will forego analyzing; and the divert functionality allows the NIDS to inspect and potentially intercede any traffic it wishes. The TM could leverage the Shunt to impose the cutoff directly on the NIC.

While intrusion detection systems such as Snort [97] and Bro [86] can record traffic, they typically keep only a small subset of the network's packets; for Snort, just those that triggered an alert, and for Bro just those that the system selected for analysis. Neither system—nor any other of which we are aware—can incorporate network traffic recorded in the past into their *live* analysis. We added this capability to the Bro system.

Commercial vendors, e.g., [26, 40, 55], offer a number of packet recorders. Due to their closed nature, it is difficult to construct a clear picture of their capabilities and performances. As far as we can tell, none of these has been coupled with a NIDS.

Finally, the notion of "time travel" has been discussed in other contexts of computer forensics. For instance, ReVirt [38] can reconstruct past states of a virtual machine at the instruction-level.

3.8 Summary

In this chapter we explore the significant capabilities attainable for network security analysis via *Time Travel*, i.e., the ability to quickly access past network traffic for network analysis and security forensics. This approach is particular powerful when integrating traffic from the past with a real-time NIDS's analysis. We support Time Travel via the Time Machine (TM) system, which stores network traffic in its most detailed form, i.e., as packets. The TM provides a remote control-and-query interface to automatically request stored packets and to dynamically adapt the TM's operation parameters. To reduce the amount of traffic stored, the TM leverages a simple but effective "cutoff" heuristic: it only stores the first N bytes of each connection (typically, $N = 10$–$20\,\mathrm{KB}$). This approach leverages the heavy-tailed nature of network traffic to capture the great majority of connections in their entirety, while omitting storage of the vast majority of the bytes in a traffic stream.

We show that the TM allows us to buffer most connections completely for minutes *in memory*, and on disk for *days*, even in 10 Gbps network environments, using

only commodity hardware. The cutoff heuristic reduces the amount of data to store to less than 10% of the original traffic. We add TM support to the open-source Bro NIDS, and examine a number of applications (controlling the TM, correlating NIDS alarms with associated packet data, and retrospective analysis) that such integration enables. In addition, we explore the technical subtleties that arise when injecting recorded network traffic into a NIDS that is simultaneously analyzing live traffic. Our evaluation using traces as well as live traffic from two large sites finds that the combined system can process up to 120 retrospective queries per second, and can potentially analyze traffic seen 4–15 days in the past, using affordable memory and disk resources.

A previous proof-of-principle TM implementation has been in operational use for several years at LBNL. The new, joint TM/NIDS installation is now running there continuously in a prototype setup, and the site's operators are planning to integrate it into their operational security monitoring.

To further improve performance in high-volume environments, we plan to develop a version of the system that implements cutoff processing in dedicated hardware (such as the Shunt FPGA [49]) or in the kernel, in order to reduce the traffic volume as early as possible. Using ideas from [30], we also plan to further extend the period we can "travel back in time" by aggregating packet data into higher level representations (e.g., flows) once evicted from the TM's buffers.

Overall, we have found that retrospective analysis requires a great deal of experience with a TM/NIDS setup in operational environments to identify the most useful applications, especially considering the trade-offs discussed in Section 3.6. Now that we have the TM/NIDS hybrid in place, the next step is to pursue a study of these possibilities. We also show the utility of the Time Machine for large scale traffic measurements and security analysis in Chapter 8.

Chapter 4

DSL Line and NAT Characteristics

We now turn to the characterization of residential broadband traffic. We begin with a look at the behavior of the users' DSL sessions (periods of connection to the ISP's network) and network address translation (NAT) usage characteristics. A first basic question concerns the durations of such connections. Network analysis studies often make the assumption that one can use IP addresses as host identifiers (for example, for studies that count the number of systems exhibiting a particular phenomenon), and previous studies have found stability in these mappings on the order of several hours to days. Moore et al. analyzed the 2001 Code Red outbreak and found that for larger timescales (days to weeks), IP addresses cannot be used as reliable host identifiers due to IP reassignment [79]; they did not examine timescales below several hours. Xie et al. observed some highly volatile dynamic IP address ranges, which they attributed mainly to dial-up hosts [132]. On the other hand, Casado et al. [21] found relatively low IP address churn. We find that DSL sessions run quite short in duration, with a median length of only 20–30 min. The short lifetime affects the rate of IP address reassignments, and we find 50% of addresses are assigned at least twice in 24 h, and 1–5% of addresses more than 10 times, with significant implications for IP address aliasing.

Network address translation (NAT) can also skew the reliability of IP addresses as host identifiers. Today, NAT is commonly used when residential users connect their computers and laptops to the Internet. Indeed, most ISPs typically offer WiFi enabled NAT home gateways to their broadband customers such that they can connect multiple wireless (or wired) devices. These NAT gateways enable customers to connect several devices to the Internet while needing only one public IP address. The prevalence of NAT devices and the number of terminals connected through a NAT gateway thus has implications on whether a public IP address can be used as a unique host identifier and if it is possible to estimate population sizes using IP addresses.

We examine the number of DSL lines using NAT and how many distinct devices or hosts are connected via such NAT gateways. Furthermore, for DSL lines showing

evidence of activity by more than one host we also study if these hosts are used concurrently.

Most previous studies on identifying NAT gateways and infering the number of hosts behind such gateways rely on information available in the packet headers, e. g., IPids, IP TTLs, or ports. Our approach takes advantage of HTTP user-agent information in addition to IP TTLs. In 2002, Bellovin [15] proposed and discussed the possibility to identify end-hosts by leveraging the fact that IPids are usually implemented as a simple counter. Nowadays an increasing number of IP-stacks implement random IPid, reducing utility of his approach. Beverly [17] evaluated several techniques to perform TCP/IP fingerprinting and found a host count inflation due to NAT by 9 % based on a one hour trace from 2004. Phaal [88] also takes advantage of the IP TTL. Furthermore, there is work in the area of OS fingerprinting, e. g., Miller [78].

Armitage [12] performed a measurement study in 2002 by offering Quake III servers at well connected Internet sites and monitoring the incoming connections. He identified NATed players by checking for non-default Quake client ports and found that 17–25 % of the players where located behind a NAT. Xie et al. [131] track IP-to-host bindings over time for counting hosts. However, they consider all host behind a NAT gateway as a single host. Casado et al. [21] use active web content to analyze NAT usage and IP address churn. By comparing local to public IP addresses they find that 5–10 % of IPs contacting *the monitored web services* have multiple hosts over a 7 month period. While their approach relies on clients contacting their web services, we can observe *all* traffic from the monitored DSL lines. We report on NAT usage from the perspective of Web service providers in Section 4.4.

Our analysis of NAT usage shows that roughly 90 % of the studied lines connect to the Internet via a NAT gateway, presenting a high potential for IP ambiguity. Indeed, for our data sets we find that 30–49 % of the lines host multiple end-hosts. Investigating the influence of shorter time-scales reveals that 20 % of lines have multiple end-hosts that are active within 1 hr of each other and 10 % of lines have activity from multiple hosts within 1 sec, i.e., on 10 % of DSL-lines the shortest observed time between activity from different hosts is less than 1 sec. These results emphasize the large error potential of techniques that rely on an IP address to uniquely identify an end-host.

The remainder of this chapter is structured as follows: We present our methodology in Section 4.1. In Section 4.2 we present observed DSL session characteristics. Next, we present our results on NAT usage and the number of hosts in Section 4.3 and the impact of shorter time-scales in Section 4.4. We then discuss limitations in Section 4.5 and conclude in Section 4.6.

4.1 Methodology

We use anonymized Radius logs (see Chapter 2) of the European ISP in order to analyze the characteristics of DSL level sessions. To analyze NAT usage among residential customers we use our 24 h traces (SEP08, APR09, and AUG09). For NAT we have to *(i)* identify lines that use a NAT gateway (e.g., a home router) to connect to the Internet and *(ii)* differentiate between the hosts behind the NAT gateway.

4.1.1 Session Characteristics

We base our analysis on Radius [96] logs, which many European ISPs use for authentication and IP address leasing. Radius supports two timeouts, *Session-Timeout* and *IdleTimeout*, though in general the monitored ISP only makes use of the first. *SessionTimeout* performs a role similar to the DHCP lease time, limiting the maximum lifetime of a session. The ISP sets it to 24 hr (a popular choice among European ISPs [81, 125]). DSL home routers generally offer an option to reconnect immediately after a session expires. However, in contrast to DHCP, Radius does not provide an option to request a particular IP address and Radius does not try to reassign the previously assigned IP address.

4.1.2 Detecting the Presence of NAT

To detect whether NAT is used on a DSL line, we utilize the fact that operating systems' networking stacks use well-defined initial IP TTL values ($initTTL$) in outgoing packets (e. g., Windows uses 128, MacOS X and Linux use 64). Furthermore, we know that our monitoring point is at a well defined hop distance (one IP-level hop) from the customers' equipment. Since NAT devices do routing they decrement the TTLs for each packet that passes through them.

These observations enable us to infer the presence of NAT based on the TTL values of packets sent by customers. If the TTL is $initTTL - 1$ the sending host is directly connected to the Internet (as the monitoring point is one hop away from the customer). If the TTL is $initTTL - 2$ then there is a routing device (i.e., a NAT gateway) in the customers' premises. Accordingly, if we observe a TTL equal to $initTTL - n$ there are multiple routing devices on the path.

We note that users could reconfigure their systems to use a different TTL. However, we do not expect this to happen often. Indeed, we do see that almost all observed TTLs are between $initTTL - 1$ and $initTTL - 3$. While there are some packets

with TTL values outside these ranges, they contribute less than 1.9 % of packets (1.7 % of bytes). Moreover, approximately half of those are due to IPSEC which uses a TTL of 255 and no other TTL has more than 0.44 % of packets. Given the low number of such packets, we discard them for our NAT detection.

A NAT gateway can come in one of two ways. It can be a dedicated gateway (e. g., a home-router) or it can be a regular desktop or notebook, that has Internet connection sharing activated. A dedicated NAT gateway will often directly interact with Internet services, e.g., by serving as DNS resolver for the local network or for synchronizing its time with NTP servers. Moreover, they generally do not use the Web or use HTTP. Some NAT gateways however use HTTP to update dynamic DNS information. We therefore define a *dedicated NAT gateway* as a device, that is directly connected to the Internet (based on TTL observations) and that does not use HTTP except for dynamic DNS updates.

4.1.3 Number of Hosts per DSL Line

We also want to count how many hosts are connected to each DSL line behind a NAT gateway to enable us to estimate the ambiguity when using IP addresses as host identifiers. A first step towards identifying a lower bound for the number of hosts per line is to count the number of distinct TTLs observed per line. Recall that Windows uses an $initTTL$ of 128 and that MacOS X and Linux use 64 and that most of the observed TTL values are within the ranges of 61–63, and 125–127. These ranges are far enough apart to make them clearly distinguishable at our monitoring point. Therefore, we can use observed TTLs to distinguish between Windows and non-Windows operating systems, yet we cannot distinguish between distinct Windows systems. This is rather unfortunate since HTTP user-agent information shows, that Windows is the dominating operating system in our user population.

However, we can use additional information to distinguish hosts. HTTP user-agent strings of regular browsers (as opposed to user-agent strings used e.g., by software update tools or media players) include information about the operating system, browser versions, etc. This can help us differentiate between hosts with the same operating system family. We find that 90 % of all active DSL lines have user-agent strings that contain such operating system and browser version information. For example, consider the following summary of all network activity of one DSL line:

Form Pkt Header		From HTTP User-Agent		
IP TTL	Proto	OS	Family	Version
63	53/DNS	–	–	–
126	80/HTTP	Win2k	Firefox	2.0.1
126	80/HTTP	WinXP	Firefox	3.0.2
126	80/HTTP	WinXP	MSIE	6
126	80/HTTP	WinXP	Firefox	2.5.1

We see a directly connected device (TTL 63 == $initTTL - 1$) that is only using DNS. According to our definition in Section 4.1.2 this device is classified as a dedicated NAT gateway. We also observe TTLs of 126, which is consistent with a Windows OS behind a NAT gateway. Examining the HTTP user-agent strings we see that both Win2k and WinXP are present. Thus, we can assume that there are at least two distinct hosts behind the NAT gateway. However, we also see that the WinXP OS uses several different browser families and versions. While it can happen that users use two different browser families on a single host (e. g., MSIE and Firefox), it seems rather unlikely that they use different *versions* of the same browser family on the same host. Using this rationale, the two different Firefox versions on WinXP indicate two distinct hosts, yielding a total of 3 end-hosts.

Or consider this example:

From Pkt Header		From HTTP User-Agent		
IP TTL	Port	OS	Family	Version
63	53/DNS	–	–	–
63	80/HTTP	Linux	Firefox	3.0.1
62	80/HTTP	Linux	Firefox	3.0.1
126	80/HTTP	WinVista	MSIE	8
126	80/HTTP	WinVista	Firefox	3.0.2

Here we also see a directly connected device (TTL 63), however there is also HTTP activity with the same TTL. We therefore classify this device as a host. We also see TTLs that are consistent with NATed Windows and Linux systems, so we conclude that the directly connected device serves a dual function: as NAT gateway and as regular computer. Moreover, we see one OS/browser combination with TTL 62—another host. For TTL 126 we see only WinVista as OS but two different browser families, which likely indicates just one host with both Firefox and MSIE installed. Overall, we infer for this example that there are 3 active hosts.

4.1.4 NAT Analysis Tool

We develop a small C program, `ttlstats`, to implement our NAT analysis. For each DSL line, the tool records whether a particular protocol was used by that line, which TTL was used in packets of this protocol, and for HTTP which user-agents were used. To identify protocols we use their well-known ports, which works well for the protocols we consider (see Section 5.1.3).

For HTTP we parse the user-agent strings and extract the operating system version (OS) and the browser version. We limit our analysis to user-agent strings from typical browsers (Firefox, Internet Explorer, Safari, and Opera), user-agents from mobile hand-held devices (see Chapter 7), and gaming consoles (Wii, Xbox, PlayStation). We do not consider other user-agents (e. g., from software update clients) since those often do not include OS information or host identifiers.

To estimate a lower bound for the number of hosts behind a NAT gateway we use two approaches:

- We only count different ⟨TTL,OS⟩ combinations as distinct hosts. We term this method OS only.
- For each ⟨TTL,OS⟩ combination we also count the number of different browser versions from the same browser family as distinct hosts. We term this method OS + browser version. Browser families are, e. g., Firefox and Internet Explorer. We do not consider different browser families as additional hosts.

In our first example above, OS only yields a host count of 2 while OS + browser version yields a host count of 3. In our second example both counting methods yield a host count of 3: one Linux system that is used as gateway and regular computer, one NATed Linux system, and one NATed computer with Windows Vista.

4.1.5 NAT Analysis for Different Data Set Types

Often the kind of data (anonymized packet-level information with HTTP) we use for this NAT analysis is not available. However (anonymized) HTTP logs might be more readily available. Vice versa IP/TCP header only traces are also common in the measurement community. Thus, we compare how well NAT analysis schemes perform when fewer information is available. For this we use several reduced information data sets, and repeat the analysis. In particular we analyze the accuracy of *(i)* using only IP TTL information (no useragent), *(ii)* using only HTTP user-agent strings (no TTL), and *(iii)* using only HTTP traffic augmented with IP TTLs (but excluding non-HTTP protocols, http). See Table 4.1 for a comparison.

Table 4.1: **Information used for different types of data sets for NAT detection.**

Information	all	http	no TTL	no useragent
HTTP User-Agent	✔	✔	✔	
IP TTLs	✔	✔		✔
Protocol Usage	✔			

4.2 DSL Session Characteristics

Looking at DSL session lengths, we expected to find typical session lengths of several hours. We analyzed the DSL session duration of the Radius logs, excluding sessions lasting under 5 minutes. Surprisingly, we find that sessions are quite short, with a median duration of only 20–30 minutes. Figure 4.1 shows the distribution of DSL session durations for those longer than 5 minutes, computed over all sessions, along with the distribution of the median session duration computed per DSL line. The data exhibits two strong modes around 20–30 minutes and 24 hr (the maximum duration given the Radius setup), partitioning the DSL lines in two large groups: always-connected lines, and lines that only connect on demand and disconnect shortly after. We do not find much in between (lines connected for several hours). While previous work found short sessions (70% lasting at most 1 hour) in the context of wireless university networks [65], we found it striking to discover such short DSL sessions in residential networks, in violation of our mental model that sessions would be significantly longer-lived.

To check if there is a significant difference in DSL session durations for P2P users vs. non-P2P users (see Section 5.1), we partitioned the DSL-lines into two groups. Overall, the characteristics of the distribution are similar, with two prevalent modes. However, we find that P2P users tend to have longer session durations and that a larger fraction of P2P users always remain connected.

To better understand the high prevalence of short sessions, we examined the Radius termination status in the logs. Radius differentiates between 18 termination causes. Figure 4.2 shows the distribution of causes for sessions longer than 5 min. We observe that more than 80% of sessions are terminated by user request (this rises to 95% for sessions under 5 min). Most likely these are caused by idle timeouts in the DSL modem on the client side. While most current broadband contracts are flat-rate, in the past time-based contracts were popular in Europe. Indeed, these latter are still offered by most European ISPs. Therefore, it is likely that consumer DSL routers come with a small idle timeout as a factory default in an

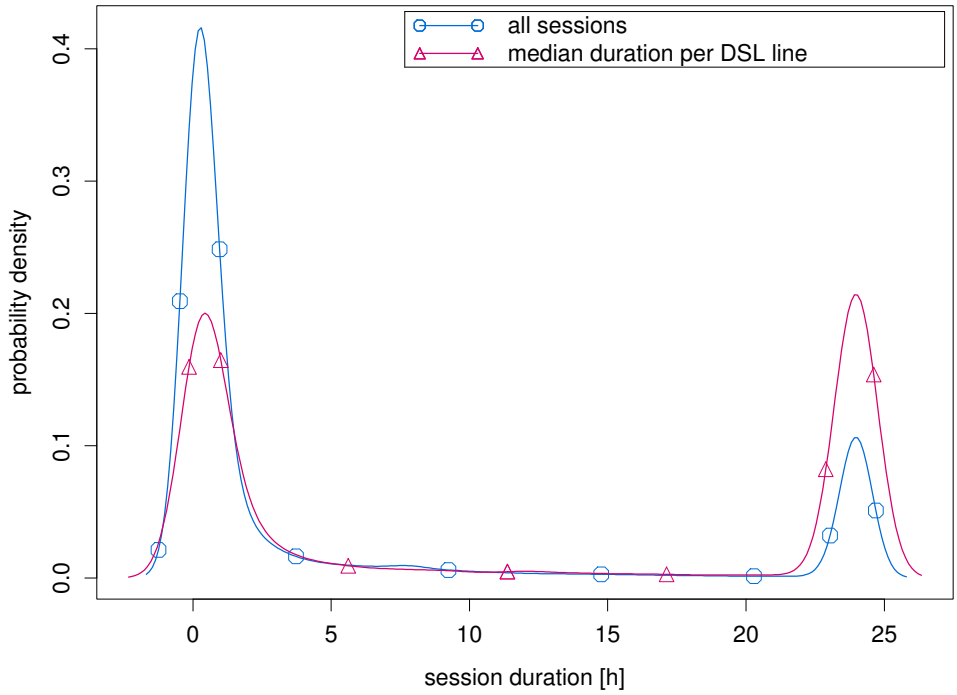

Figure 4.1: **PDF of session durations for sessions with duration >5 minutes for dataset TEN.**

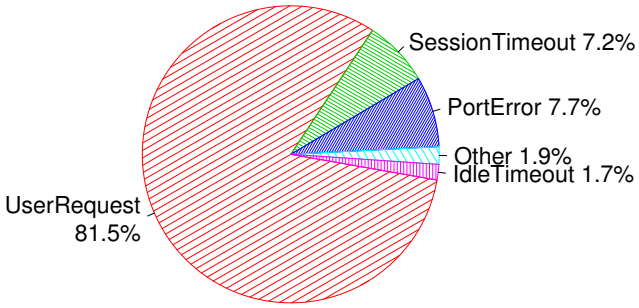

Figure 4.2: **DSL (Radius) session termination causes distribution for sessions lasting longer than 5 minutes.**

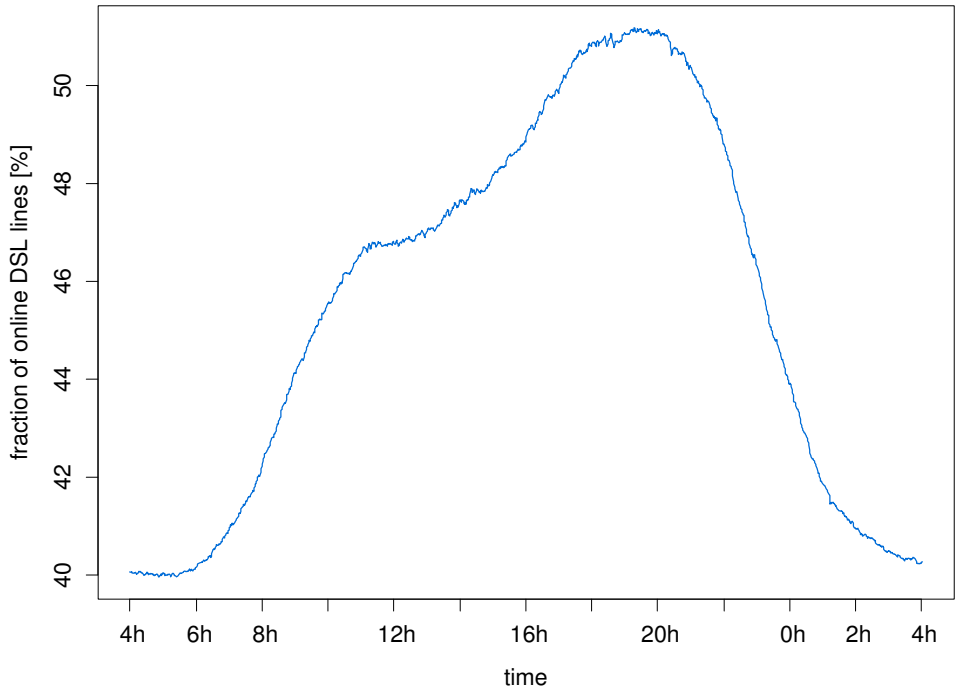

Figure 4.3: **Relative number of concurrent DSL lines across time for one 24h weekday period of dataset TEN. Note the base-line at 40 %.**

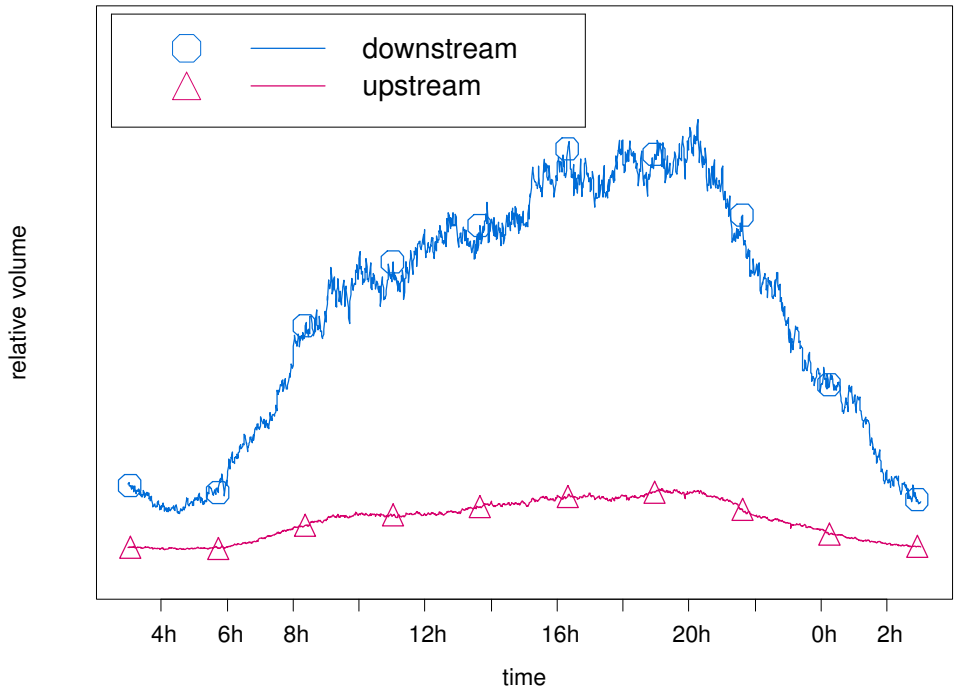

Figure 4.4: **Bandwidth usage of all DSL lines across time (1 min bins).**

effort to aid users in keeping down costs. We checked the default settings for several popular home routers available to us and verified that they indeed have a small idle timeout. The second most common termination cause is PortError, which likely results when users power off their DSL modem as part of powering down their entire computing setup.

Since many DSL sessions are short and Radius does not preserve IP address assignments across sessions, we therefore expect (and find) IP addresses used for multiple DSL lines across each dataset. During a 24 hr period we find 50% of the IP addresses assigned to at least two distinct DSL lines, and 1–5% to more than ten DSL lines. These results underscore *the peril involved in using an IP address as a long-term reliable host identifier*.

Previous work found that for consumers diurnal patterns start with activity in the morning, steadily increasing throughout the course of the day, with the height of activity starting in the early evening and lasting till midnight [43, 46]. We see this same overall pattern in terms of the number of active DSL sessions, as shown in Figure 4.3. However, we note that the variation is in fact modest, with 40% of

Table 4.2: **Overview of NAT results. Top three rows are relative to total number of active lines, remaining rows are relative to "Lines with active hosts" (B.2)**

Ref.	Description	SEP08	APR09	AUG09a	AUG09b	Base
A.1	Lines using NAT	89 %	91 %	92 %	92 %	num.
B.1	Lines on which only NAT gateway is active	9 %	10 %	14 %	18 %	total lines
B.2	Lines with active hosts (NATed and unNATed)	91 %	90 %	86 %	82 %	
C.1	Lines with unNATed Windows	9 %	8 %	7 %	7 %	
C.2	Lines with unNATed Linux/Mac	1 %	1 %	1 %	1 %	
D.1	Total systems (OS only)	141 %	142 %	143 %	140 %	
D.2	Total systems (OS + browser version)	155 %	162 %	179 %	172 %	
E.1	Lines with > 1 host (OS only)	30 %	31 %	31 %	30 %	
E.2	Lines with > 1 host (OS + browser version)	36 %	39 %	49 %	46 %	

the lines permanently connected. We also observe a slight day-of-week effect, with Sundays having larger numbers of concurrent sessions, and Friday/Saturday lower daily maxima than other weekdays.

We also observe a diurnal pattern in bandwidth usage, per Figure 4.4, with the relative differences now being much more pronounced. After all, keeping a session alive does not imply any bandwidth usage per se.

4.3 NAT Usage and Hosts per DSL Line

In this section we present the results from our NAT analysis. We first discuss the prevalence of NAT devices at DSL lines before continuing with the number of hosts per line. Finally, we investigate NAT detection with different data set types.

4.3.1 NAT Usage

Overall, we find that NAT is prevalent and that the vast majority of DSL lines use NAT to connect hosts to the Internet. We also find that a significant number of lines connects more than one host. Table 4.2 summarizes our key findings. More than 90 % of lines utilize NAT (Table 4.2, row A.1). This result is in contrast to

the findings of Armitage [12] who only found 25 % of the IPs were behind a NAT. On 9–18 % of lines (B.1) only the NAT devices itself is active[1], that is, we see only traffic that we attribute to the NAT gateway and no traffic from regular hosts. The remaining lines (82–91 %, B.2) have active hosts (those lines may or may not be NATed).

We next take a closer look at DSL lines with *active hosts* and determine how many of these lines are using NAT. We find that only 8–10 % (C.1 and C.2) of lines with active hosts are not NATed, i.e., there is only one host which is directly connected.

Finally, we investigate how many more hosts than lines are present: the ratio of detected hosts to the number of lines. In rows D.1 and D.2 we show the number of observed hosts relative to the number of lines with active hosts. For D.1 we use the heuristic which counts every unique TTL and OS combination as a separate host (OS only). For row D.2 we also increment the per line host count if we observe TTL-OS combinations with multiple version of the same browser family (OS + browser version). According to our definition, we will always see more hosts than lines with active hosts. However, the differences are strikingly large—up to 1.79 times as many hosts than lines in AUG09a using the OS + browser version counting method. Independent of the estimation method the number of hosts behind NAT gateways, our host counts are far larger than the estimations by Beverly [17] from 2004, who estimated 1.09 times more hosts than IPs. This difference might be due to 6 additional years of NAT gateway deployment, different vantage points (Internet peering/exchange point vs. broadband access), different observation periods (1 h vs. 24 h), and/or information base (SYN trace vs. TTL plus HTTP logs).

4.3.2 Number of Hosts per Line

Given that we see so many more hosts than lines with active hosts, we next investigate lower bounds for the number of lines with more than one host. If there are many such lines it implies there are many public IP addresses with more than one host, thus limiting the utility of IP addresses as host identifiers. We see (Table 4.2, rows E.1 and E.2) that 30–49 % of lines have more than one active host. We note that between APR09 and AUG09a the number of lines with more than one host (OS + browser version, row E.2) increases significantly. We attribute this to an increase in browser heterogeneity. Following the release of MSIE 8 in late March 2009, we observe a significant share of MSIE 6, 7, and 8 in AUG09, while only MSIE 6 and 7 have a significant share in SEP08 and APR09. Consider the example

[1] We term a device or host as active if it sent IP packets during the trace

that two hosts use a DSL-line and both have WinXP and MSIE 7. In this case we cannot distinguish between them. However, if one is upgraded to MSIE 8 while the other is not, then we can distinguish them.

In Figure 4.5 we present a more detailed look by plotting the fraction of lines with n hosts. We only present plots for SEP08 and AUG09a, the other traces exhibit similar behavior. We focus on the bars labeled "all" first. Note that we observe up to 7 % of lines with more than 3 hosts. We also investigate whether this high number of lines with multiple hosts is due to several computers (PCs or Macs) that are used via the same line or whether mobile hand-held devices (e.g., iPhones, see Chapter 7), or game consoles (e.g., Wii) are responsible for this. We identify these devices by examining the HTTP user-agent string. If we exclude mobile hand-held devices and game consoles, 25–28 % (OS only; 34–44 % with OS + browser version) of lines have more than one host (not shown in table). Therefore, we conclude that multiple end-hosts are in general due to users connecting several computers. However, we also find that the more hosts we observe behind a NAT, the higher the probability that a mobile or gaming device is present.

4.3.3 NAT Analysis with Different Data Set Types

As discussed in Section 4.1.5, we also use reduced data sets (http, no TTL, and no useragent) and compare the NAT usage estimates to those based on the full data set available to us (all). Figure 4.5 compares the number of hosts per line for the different data sets. Note, without HTTP user-agent data there is no difference between the scheme for OS only and OS + browser version. Most accuracy is lost when relying on IP TTL only (no useragent). Removing the IP TTL (no TTL) and relying on HTTP user-agents information only shows slightly better results. Compared to all information, using HTTP logs annotated with TTL information (but discarding all non-HTTP activity, http) gives a very good estimate of NAT prevalence.

4.4 Impact of Shorter Time-Scales on NAT

So far we have limited our discussion to a static view of NAT behavior, i. e., we analyzed whether a DSL line is NATed and how many hosts are connected via this line. If a line has more than one host, IP addresses cannot be reliably used as host identifiers when considering time-scales of one day (our trace duration). However, it is possible that even though a line has two hosts, the first host is only active in the morning while the second host is only active in the evening.

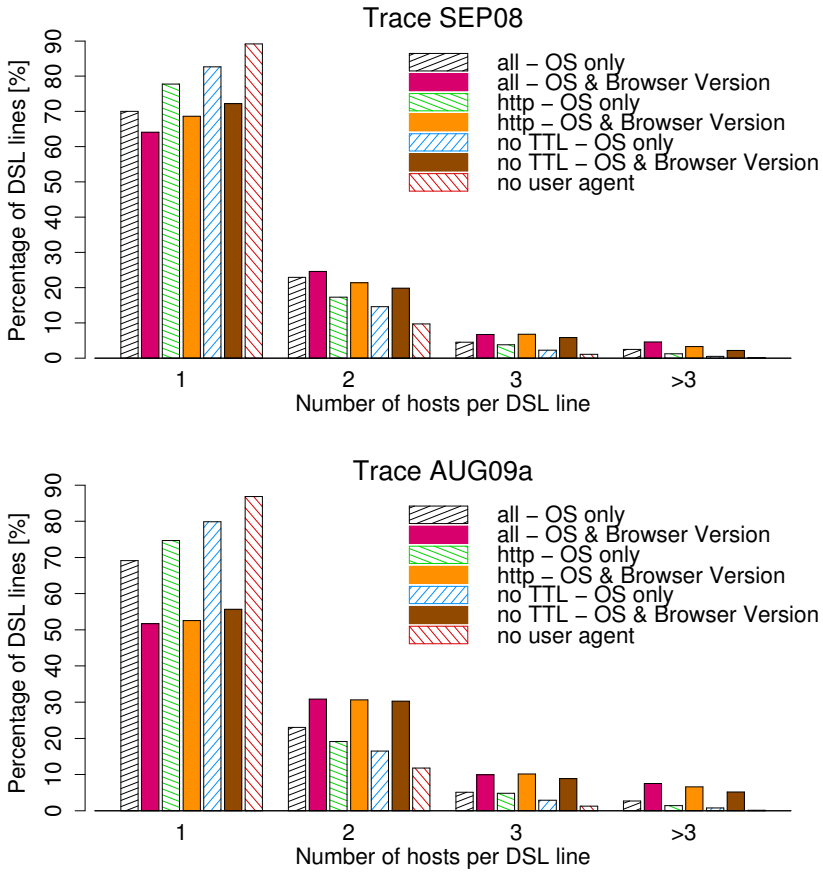

Figure 4.5: **Fraction of DSL lines vs. number of hosts per line for SEP08 and AUG09a**

Thus, although the line has two hosts, they are not used at the same time. This reduces the ambiguity of using IP addresses as host identifiers over smaller time intervals (e. g., by utilizing timeouts). Likewise, two hosts behind a NAT device may contact different network services. Therefore, we also analyze whether we still observe multiple hosts behind a single IP address when considering only traffic to/from particular Web domains.

4.4.1 Analysis Approach

To answer if multiple devices are used at the same time, we compute the *minimum* inter activity time (IAT) between a HTTP request from one host to the HTTP request from another host on the same DSL line. If we observe an minimum IAT

of T seconds then we know that two or more distinct hosts were active at this line within T seconds. As we need timestamps for this analysis we cannot use the output of the `ttlstats` tool (Section 4.1.4) as it aggregates all activity of a line for scalability reasons. Therefore, we revert to using HTTP request logs[2], which corresponds to the "http" data type and use the "OS only" counting method.

To decide whether multiple hosts at a DSL line contact the same domains at the same time, we include the second level domains (e.g., `example.com`, `example.at`) from HTTP's Host header into our analysis, i.e., we calculate the IATs between different hosts on the same line accessing the same web service. This enables us to see the effects of multiple hosts behind a NAT device when only analyzing a subset of the data, i. e., traffic to/from a specific web service. We focus on at set of web services that are popular among our user population.

4.4.2 Results

In Figure 4.6 we plot the fraction of lines with two or more hosts for increasing minimum IATs. This plot enables us to study how close in time two (or more) hosts are active via the same line. This allows us to estimate by how much ambiguity can be reduced by using a timeout, i. e., by using the IP-to-host mapping only for a limited time.

Even with intervals *as low as 1 sec* we observe *more than 10 % of lines with multiple hosts*. When considering IATs of 1 h, almost 20 % of lines see activity from multiple hosts. We thus conclude that if a line has multiple hosts they are likely active at the same time or within a short time period. We confirm these results by applying the static analysis (see Section 4.1.3 and Section 4.3.2) for slices of the traces, i. e., we subdivide each trace into time bins of 1, 5, 10, 30, and 60 minutes and repeat the analysis for each bin.

Next, we explore if this ambiguity in IP to end host mappings persists when looking at a specific web service (by second level domains). In Figure 4.7 we again plot the fraction of lines with multiple hosts for increasing minimum IATs when only considering traffic to/from a specific Web service[3] in AUG09a. We normalize by the number of users observed per web service. For comparison we repeat the result from Figure 4.6 (top most points, label "all domains"). As expected the different domains do not experience as many DSL lines with multiple hosts, especially for short intervals (1s to 1min). Nevertheless, except for Microsoft, looking at IATs of

[2] These logs include timestamps for every request. We rely on Bro [86] for HTTP parsing.

[3] We present results on Apple, Doubleclick, Facebook, Google, Microsoft, Rapidshare, and YouTube

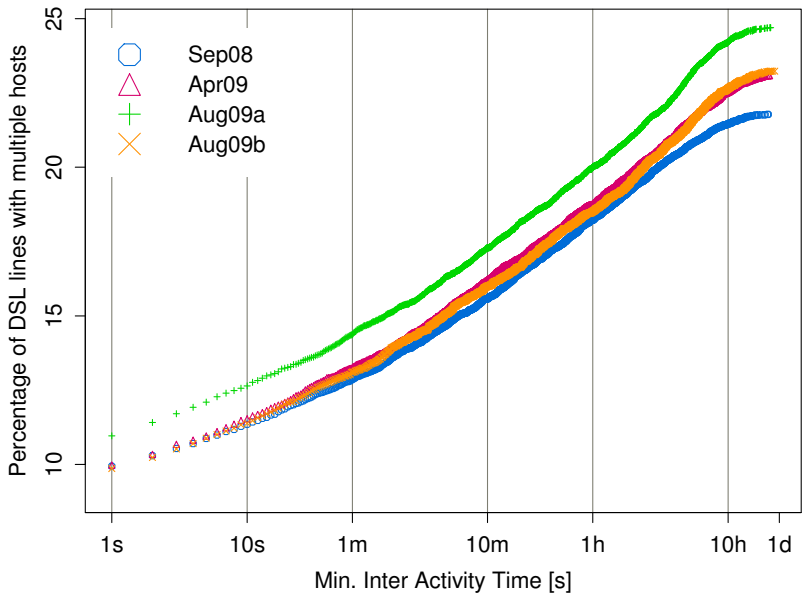

Figure 4.6: **Fraction of DSL lines with more than one active host within a particular time interval using OS only (y-axis starts at 10 %).**

10 hours we observe multiple hosts on 5-15 % of the DSL lines (IPs) using the web service. Furthermore, we note that ubiquitous services such as Google or third-party Ad-servers, such as Doubleclick, are exposed to a higher fraction of IPs with multiple hosts. Another notable difference is observed for Rapidshare. Rapidshare starts with a high fraction of lines with multiple hosts even for short IATs (8 %) which is closer to the overall data set.

In 2007, Casado et al. [21] analyzed NAT from the perspective of Web service providers. They found that over their 7 month observation period 5–10 % of the IPs contacting *their services* had multiple hosts. However, they report a total 177 M Web request over 7 month, while we record over 70 M requests per day (up to 4 M per service). Therefore, our results focus on the popular Web services.

4.5 NAT Analysis Considerations

We aim at estimating the number of active end-hosts per DSL line. Our methodology is tailored towards underestimating the actual number of end-hosts per line for the following reasons: As explained in Section 4.1.3 we do not count different browser types as multiple hosts, although this might be a good indicator for

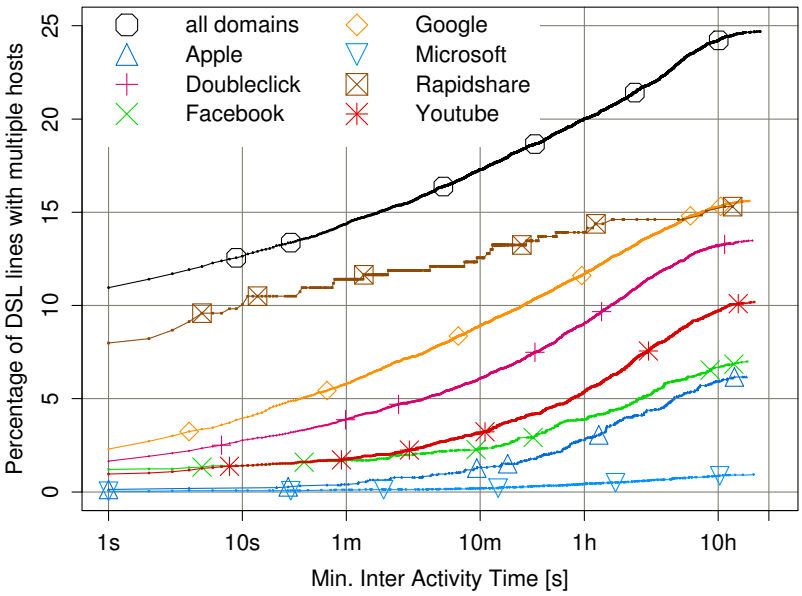

Figure 4.7: **Fraction of DSL lines with more than one active host within a particular time interval by Web service using OS only for AUG09a.**

multiple hosts. *How likely is it to have multiple Browsers installed?* Furthermore, we focus mainly on IP and HTTP. Parsing additional application protocol headers might reveal additional hosts that were not counted (e. g., P2P peer IDs; however only a small fraction of DSL lines use P2P protocols).

On the other hand there are factors that can bias our results towards overestimating the number of hosts per DSL line: If a user has two operating systems installed on one host (e. g., Windows for gaming and Linux for working) our method counts this machine as two different hosts. Likewise, if a user updates his browser during our observation period we also count the same machine twice. However, these artifacts decrease as we consider shorter IATs since it requires time to reboot another operating system and/or update a browser. Therefore, the results for small IATs are reasonable lower bounds for the number of hosts per line.

4.6 Summary

We started this chapter with DSL level characteristics, examining session durations, their termination causes, and the number of concurrent sessions. Session

durations are surprisingly short, with a median duration of only 20–30 minutes, while we would have expected several hours to days. Our termination cause analysis turned up that most sessions end due to termination from the user end, which we attribute to default router configurations based on former timed contracts. As a consequence, IP addresses are reassigned frequently, with up to 4% of addresses assigned more than 10 times a day. This indicates that the use of IP addresses as host identifiers can prove quite misleading over fairly short time scales.

Next, we presented a novel approach for detecting DSL lines that use network address translation to connect to the Internet. Our approach is able to infer the presence of a NAT device and to provide lower bounds for the number of hosts connected behind the NAT gateway. For lines with multiple hosts connected we also studied the temporal behavior to see whether multiple hosts are active at the same time. Our approach relies on IP TTL information and HTTP user-agent strings and we analyze the accuracy when using less information (e.g., only TTLs, or only user-agent strings) for the NAT analysis. We find that most accuracy is lost when user-agent strings are omitted.

We find that 10 % of DSL lines have more than one host active *at the same time* and that almost 20 % of lines have multiple hosts that are active within one hour of each other. Overall 30–49 % of lines have multiple hosts. Checking if multiple simultaneously active hosts per line contact the same remote destination we show that the fraction of lines with multiple hosts drops significantly. However, still remains noticeable. These results underscore the perils involved when using IPs as host identifiers.

Chapter 5

Application Layer Characteristics

To understand the popular applications among our user population, we examine our application classifications and anonymized application-layer header traces. We in addition assess how well purely port-based classification would perform for correctly identifying residential traffic patterns, and characterize traffic asymmetries.

We then focus on the dominant protocol, HTTP, and analyze its usage in depth before turning to the achievable throughput that various P2P and client-server application protocols achieve.

To simplify the presentation, we focus our discussion on SEP08. However, we verified our results across all traces (WEEK, APR09, AUG09) and explicitly point out differences. In particular, we use the 14 samples from WEEK to verify that there are no dominant day-of-week or other biases apparent in the 24 h traces. In addition, we cross-checked our results with sampled NetFlow data exported by 10 of the ISP's routers. This further increases our confidence in the representativeness of our application mix results.

5.1 Application Protocol Usage

Previous studies of Internet application mix found HTTP to predominate around the turn of the century. Fraleigh et al. [45] analyzed packet level traces recorded from the Sprint backbone in 2001, finding that in most traces HTTP contributed $> 40\%$ of all bytes, though several traces had P2P contributing 80%.

Subsequent studies found that P2P became the dominant application. Ipoque and Cachelogic both used data from their deployed deep packet inspection and traffic management systems at selected customers sites to assess the application usage [85, 102, 103]. Cachelogic claimed that by 2006 P2P accounted for more than 70% of the traffic, with Ipoque supporting this claim for 2007. For 2008

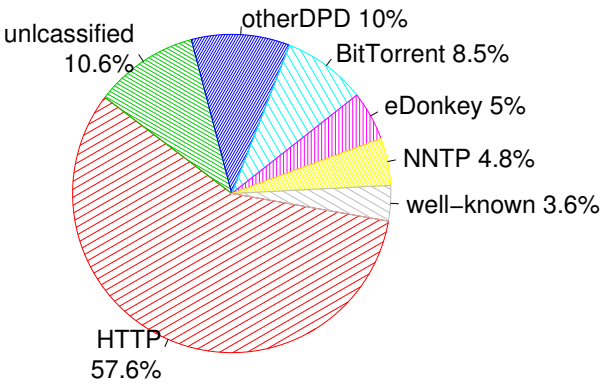

Figure 5.1: **Application protocol mix for trace SEP08.**

Ipoque found that P2P in Europe accounted for more than 50% of traffic (with Web contributing another 25%).

On the other hand, Hyun-chul et al. reported that payload-based analysis conducted in 2004 from within the PAIX backbone found almost no P2P traffic, but more than 45% HTTP [54]. On the other hand, the same study developed how at various university networks the traffic differs; for example, at KAIST in 2006 they found under 10% HTTP, and 40–50% P2P.

Cho et al. [24, 25] also found in 2008 that TCP port 80 contributed only 14% of all bytes in Japanese ISP backbones (9% in 2005), with the bulk of traffic being on unassigned ports. None of the default P2P ports contributed more 1% of the traffic volume. (The authors point out that WINNY, the most prevalent P2P application in Japan, uses unassigned ports.) They found that residential traffic exhibited a shift to more streaming and video content, which agrees with recent blog and news reports that claim that P2P traffic has somewhat declined, with streaming media increasing [9, 116]. With an assumption that the unassigned ports indeed reflected P2P, their datasets indicated that P2P dominated the total traffic volume.

From a somewhat different perspective, Kotz and Essien [64, 65] reported that 50% of wireless traffic in 2001 on a university campus, which included residential buildings, used HTTP's well-known ports, with 40% of this traffic *incoming* to local servers. Henderson et al. [53] compared these results with newer traces from 2003/2004 of the same network, finding major shifts in the application mix (HTTP 63%→27%, File systems 5%→19%, P2P 5%→22%), and that more traffic stayed on-campus than in 2001 (70%, up from 34%). Of the P2P traffic, 73% remained

internal. Therefore, we cannot easily compare these results to residential broadband use. Finally, Fraleigh et al. [45] also used a port-based approach on 2001 data, finding that on some links 60% of the bytes come from P2P and only 30% from HTTP, although most of their traces have more than 40% HTTP.

Given this context, we now turn to an analysis of application usage in our 2008/2009 residential traces.

5.1.1 Application Usage Analysis

To robustly identify application protocols, we employ the Bro system's Dynamic Protocol Detection (DPD) [36]. DPD essentially tries to parse each connection's byte stream with parsers for numerous protocols, deferring determination of the corresponding application until only that application's parser recognizes the traffic. DPD also uses regular expression signatures to winnow down the initial set of candidate parsers. The Bro distribution includes full DPD parsers/recognizers for BitTorrent, FTP, HTTP, IRC, POP3, SMTP, SSH, and SSL. We extended the set of detectors with partial recognizers for eDonkey and Gnutella (both based on L7-filter signatures [68]), NNTP, RTP, RTSP, SHOUTcast, SOCKS, and Skype.

In the SEP08 trace we can classify more than 85% of all bytes, with another 3.6% using well-known ports, as reflected in Figure 5.1. We find that *HTTP, not P2P,* is the most significant protocol, accounting for 57% of residential bytes. We also find that NNTP contributes a significant amount of volume, nearly 5%. Almost all of the NNTP bytes arise due to transfers of binary files, with RAR-archives (application/rar) being among the most common file types, suggesting that the traffic reflects the equivalent of file-sharing. Indeed, Kim et al. [61] did a subsequent study of NNTP traffic and found that most NNTP traffic is due to binary downloads.

We find that P2P applications—BitTorrent, Gnutella, and eDonkey—contribute $< 14\%$ of all bytes, with BitTorrent the most prevalent, and Gnutella almost non-existent. However, the L7-filter signatures for eDonkey may be incomplete. We observe a significant amount of traffic (1.2%) on well-known eDonkey ports that the classifier fails to detect as eDonkey. The distribution of connection sizes for this traffic closely matches that for traffic positively identified as eDonkey (and differs from other applications). If we presume that this indeed reflects eDonkey traffic, then the overall share of P2P traffic increases to 17–19%, with eDonkey's popularity roughly the same as BitTorrent's. But even if we assume that *all* unclassified traffic is P2P, the total P2P share still runs below 25%.

P2P applications could also in principle use HTTP for data download, thus "hiding" among the bulk of HTTP traffic and increasing the significance of P2P traffic volume. However, our in-depth analysis of HTTP traffic (Section 5.2) finds that this is not the case.

Streaming protocols[1] (RTSP, RTMP, SHOUTcast) account for 5% of the traffic in terms of bytes. We identify RTSP and SHOUTcast using partial DPD parsers, while we identify RTMP's based only on its well-known port. We also find noticeable Voice-over-IP traffic (Skype [19], RTP), about 1.3% of the total bytes.

In order to increase our confidence in the representativeness of our application mix results, we analyzed sampled NetFlow data exported by 10 of the ISP's routers. This data shows that 50% of the traffic comes from TCP port 80. We further compared our results with those from a commercial deep-packet-inspection system deployed at a different network location, finding a close match.

Our analysis of the other traces confirms the findings outlined above. In particular the other traces confirm that our results are not biased by the day-of-week we choose. However, while the HTTP traffic share in the APR09 trace is about the same, we find slightly more unclassified traffic. We note that the overall P2P traffic decreases somewhat, and shifts from eDonkey to BitTorrent (now 9.3%). Also the fraction of NNTP traffic decreases. On this day it only accounted for 2.2% of the traffic. When we investigate the rural Indian AirJaldi community network, we find that HTTP dominates the application protocols mix here as well, accounting for 56–72 % of the total traffic volume. However, instant messenger and VoIP traffic also contribute a significant share, which is in contrast to the European ISP where we observe more NNTP and P2P.

We might expect that application usage differs widely between users with different access speeds. Figure 5.2 shows the application mix seen for different downstream bandwidth rates. Although the mix does vary, the changes are modest, other than for more P2P traffic with higher bandwidths, and much higher NNTP prevalence for the 17000 Kbps class. However, only a small percentage of lines use NNTP, so its contribution to traffic mix can see more variation.

However, we do find that lines with higher access bandwidth have a higher utilization in terms of average volume per line. Lines in the 3500 and 6500 Kbps categories contribute about twice as many bytes per line than lines in the 1200 Kbps class, and 17,000 Kbps lines three times more. We also find that general traffic per line is consistent with a heavy-tailed distribution, and the top 2.5% of lines account for 50% of the traffic.

[1] We do not consider video delivery via HTTP as streaming. We refer to those as progressive HTTP downloads.

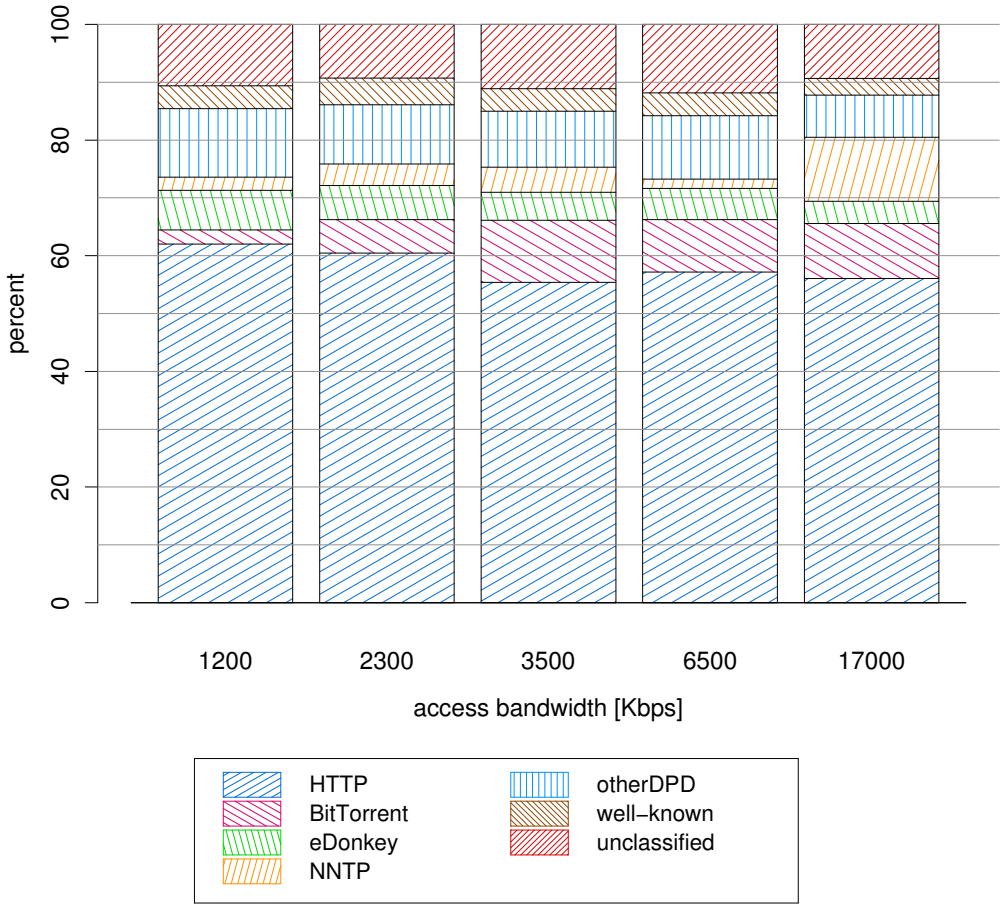

Figure 5.2: **Relative application mix per access bandwidth for SEP08. Bottom bar is HTTP, top bar unclassified.**

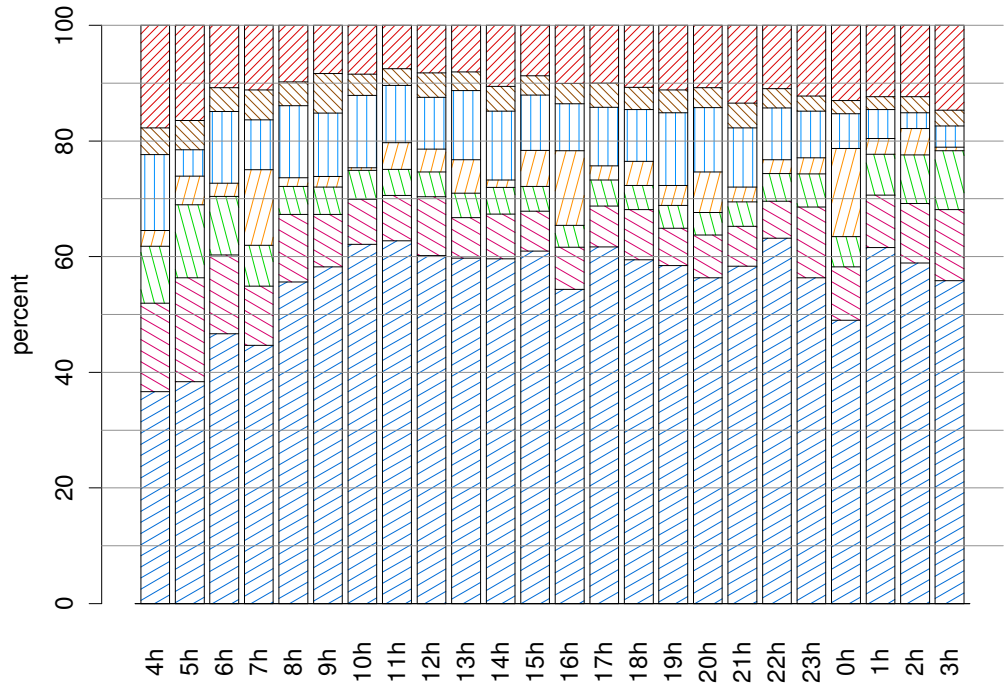

Figure 5.3: **Relative application mix hour-by-hour for SEP08. Same legend as in Figure 5.2.**

To see if time-of-day effects influence the application mix, we examine the application mix per hour, see Figure 5.3. We would expect to to observe more bulk downloads and less interactive traffic during off-hour period, which our data confirms. Night-time traffic includes a larger fraction of P2P traffic, though HTTP remains dominant during every time slot. Also, we again note high variability in NNTP due to the small number of lines using it.

In contemporaneous work Erman et al. [41], Labovitz et al. [69], and Sandvine Inc. [98] also studied the application mix in different network environments. Erman et al. analyzed the traffic of a major US broadband provider in the context of understanding the potential for forward caching. They find that HTTP contributes 61% on average and 68% during the busy-hour to the traffic volume in the downstream direction while P2P only contributes 12%. Labovitz et al. and Sandvine used results from their deployed deep packet inspection and traffic management systems at selected customers sites to assess the application mix and came to similar conclusions as we. As such, their results strengthen our observation that HTTP is again on the rise and P2P on the decline.

5.1.2 Application Mix of P2P vs. Non-P2P Lines

Next we study if the application usage of those lines that frequently use P2P differs from those that do not. We find that roughly 3% of DSL-lines use P2P protocols and that their traffic contribution accounts for 30% of overall volume. If a line uses P2P protocols, they usually also account for most of the line's traffic: 29% BitTorrent and 17% eDonkey. However, HTTP is still popular and is responsible for 23% of transferred bytes. We also note that the fraction of unclassified traffic is higher at 23%, corresponding to roughly 64% of all unclassified traffic. There is hardly any NNTP usage, only 0.6% of bytes.

Non-P2P lines predominantly use HTTP, for which it contributes 72% of their traffic volume, followed by NNTP with 6.5%, with only 5.2% of the traffic unclassified. Streaming services are also more dominant in this group (6.7%).

5.1.3 Does Port-Based Classification Work?

Very often in networking studies it is easier or more tenable to acquire TCP/IP transport information rather than relying on deep packet inspection systems. A significant question concerning the accuracy of such studies regards the degree to which one can soundly infer application protocols based solely on the TCP/UDP port numbers that connections use. Certainly, in adversarial settings, classification

Table 5.1: **DPD vs. destination port based application detection for SEP08. V_D is the volume identified by DPD for a given protocol P, V_P is the volume observed on the P's default port(s), and V_{DP} is the intersection of the two (running on P's default port and detected as P).**

Protocol	V_{PD}/V_D	V_{PD}/V_P
HTTP	97.5%	98.1%
BitTorrent	4.8%	66.1%
eDonkey	36.6%	55.9%
SSL	75.2%	86.1%
NNTP	66.7%	95.3%
RTSP	92.6%	99.1%

based on port numbers has quite limited power, due to the ease by which end systems can vary the ports they use. However, for non-adversarial situations, one might hope to leverage a predominant tendency for applications to indeed stick with the port assigned for their use.

Our DPD-based analysis—which is highly accurate for those applications where we have a full protocol parser, and still potentially quite accurate when we employ only a partial parser—presents an opportunity to assess the accuracy of port-based classification using fairly solid ground truth.

Numerous previous studies have indicated that the advent of P2P has rendered port-based approaches infeasible. Cho et al. [25] found that on Japanese Internet backbone links, 79% of traffic (by bytes) uses unknown ports, and that TCP port 80 contributes only 14% of bytes. In 2004 Karagiannis et al. [59] found P2P traffic increasingly moving away from well-known ports to dynamically negotiated ports. Kim et al. [54] found that port-based detection quality is inversely proportional to the fraction of P2P traffic.

We confirm that for current residential traffic a port-based approach works quite well. Table 5.1 shows for dominant application layer protocols how well a port-based approach would have performed. For each protocol P, column V_{PD}/V_D is the fraction of the traffic volume observed on P's default port(s) that DPD identifies as P. Column V_{PD}/V_P shows the proportion of the traffic on P's port that would be correctly identified by only inspecting the port number.

We interpret the table as follows. Most of the HTTP traffic (97.5% of bytes) does indeed appear on port 80 (middle column), and when looking at traffic on port 80

we find that 98.1% of those bytes come from HTTP (righthand column). The largest non-HTTP application on port 80 is SHOUTcast, a HTTP-like streaming protocol. We therefore conclude that for our traffic, classifying port 80 traffic as HTTP yields a good approximation for the total volume of HTTP traffic.

NNTP can only be partially identified by its default port (119). About two-thirds of NNTP traffic uses that port, and of the traffic appearing on that port, nearly all (95.3%) is indeed NNTP. From DPD we know that the remainder uses the well-known HTTP proxy port, 3128. For SSL-based protocols (HTTPS, IMAPS, POP3S, SSMTP, NNTPS) we find roughly 75% using well-known ports, while more than 90% of RTSP bytes appear on its default port (554).

The story is vastly different for P2P protocols, however. Since many institutions try to block P2P traffic with port-based filters, most P2P protocols have evolved to use non-standard, dynamically negotiated ports. Still, one third of the detected eDonkey traffic uses its well-known ports, and finding traffic on either those ports or on the BitTorrent ports generally means that the traffic is indeed caused by those protocols. (Interestingly, we find that 3% of BitTorrent traffic appears on *eDonkey* ports.)

5.1.4 Traffic Symmetry

A common assumption regarding residential traffic is that the downstream dominates the upstream, i.e., most bytes are transfered to the local side. Indeed, this assumption has shaped—and is ingrained in—the bandwidth allocations of ADSL and cable broadband offerings.

In our datasets, we observe that most bytes appear in connections originated locally, with only 10% due to connections originated remotely. The largest fraction of incoming traffic is unclassified (33% of bytes), significantly higher than for outgoing connections, and with P2P the most significant contributor by volume (28% BitTorrent, 17% eDonkey). Voice-over-IP and streaming protocols also contribute significant volume to incoming connections (10%). Incoming FTP data connections for active FTP sessions account for just over 1% of bytes in incoming connections. Finally, we find that very few lines offer "classic" Internet services like SMTP or HTTP.

When looking at the number of bytes transfered upstream and downstream, i.e., the symmetry of traffic, we find that 85% of all bytes come downstream, i.e., the asymmetry assumption does hold (though likely bandwidth asymmetry helped shape this). This proportion is much higher than seen in the Japanese backbone studies [24, 46], which found only 55% of volume was downstream. However, they

found P2P dominated their traffic mix, thus contributing to symmetry. For our traffic, we find that for P2P applications only 59% of bytes come downstream, yielding an upload/download "share-ratio" of $41/59 \approx 0.7$—still resulting in less symmetry than seen in the Japanese studies.

5.2 HTTP Usage

As HTTP dominates the traffic in our datasets, we now examine it more closely to characterize its usage. A basic question concerns what has led to its resurgence in popularity versus P2P traffic, with two possible reasons being *(i)* HTTP offers popular high-volume content, e.g., [22, 94], and/or *(ii)* HTTP serves as a transport protocol for other application layer protocols, including possibly P2P [9, 116]. We find that 25% of all HTTP bytes carry flash-video, and data exchanged via RAR archives contributes another 14%. Thus, clearly much of HTTP's predominance stems from its use in providing popular, high-volume content. We further find that in terms of volume, HTTP is *not* significantly used for tunneling or P2P downloads.

Many facets of HTTP usage have seen extensive study, as thoroughly surveyed by Krishnamurthy and Rexford [66]. Some studies have focused on understanding user behavior [11, 14, 32], while others have examined changes in content [128] and the performance of web caching [1, 3, 14, 42]. Other work has looked at media server workloads regarding file popularity and temporal properties, such as in terms of live media streams collected from a large CDN [111], and file reference characteristics and user behavior of a production video-on-demand system in large-scale use [133].

More recently, various efforts have aimed at understanding from passive measurements how the rapid advent of "Web 2.0" applications has changed HTTP traffic patterns [99], as well as Web-based applications such as YouTube [47, 136] and online social networks [48, 80, 100]. Others have employed active probing to study specific features of such applications [22].

Sites like alexa.com employ user-installed toolbars to track the popularity of various Web sites across demographic groups. They find that google.com, yahoo.com, youtube.com, and facebook.com currently rank among the most popular sites in terms of number of visits. In contrast, in this study we analyze popularity in terms of traffic volume.

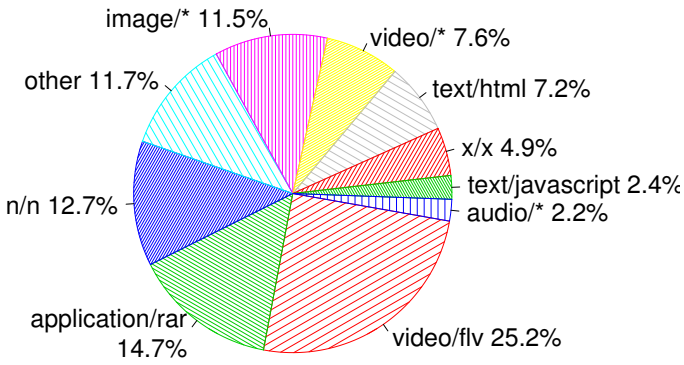

Figure 5.4: **Top HTTP content-types by volume for trace SEP08.**

5.2.1 Content Type Distribution

We use Bro's HTTP analyzer to parse the anonymized HTTP headers and compute the size of each HTTP request/response pair. To identify the content types of objects, we both examine the HTTP Content-Type header and analyze the initial part of the HTTP body using libmagic. We find more than 1,000 different content-types in HTTP headers. Surprisingly, the results of these two approaches often disagree: 43% of all HTTP bytes (28% of requests) exhibit a mismatch. Some disagreements are minor and easy to resolve. For example, in the absence of a standardized MIME type representation we can find several different strings used for the same type. We also often see generic use of application/octet-stream as Content-Type. In other cases, the sub-type differs: for example, the Content-Type header may specify "image/gif," while libmagic yields "image/jpeg".

When Content-Type and libmagic disagree, we try to identify the most likely "true" content type by using heuristics. We start by normalizing the content types and giving priority to libmagic for those content types with well-known formats, e.g., most image and video types. For other formats, we manually examine the mismatches and pick the most likely resolution. We report mismatches we could not resolve as "x/x" in our results, and generic or unidentified content types, such as application/octet-stream, as "n/n". All in all, our analysis illustrates the need for considerable caution when basing an assessment of content types solely on the Content-Type header.

Figure 5.4 shows a pie chart of the distribution of bytes per content type from the SEP08 trace. The most common content-type by volume is flash-video (video/flv)—

the format used by sites such as youtube.com and many news sites—which contributes 25% of the bytes. This is followed by the archive format RAR (application/rar), which accounts for 15% of HTTP traffic.

The unknown or unidentifiable content-types together account for 18% of the HTTP traffic. We find that a significant portion of this traffic reflects automated software updates, as 14% of the unidentifiable bytes come from a single software update site. Image types (GIF, PNG, and JPEG) contribute 11.4% of bytes, while video types other than Flash account for only 7.6%.

During the night we observe a higher fraction of RAR objects and unknown objects, while the relative popularity of HTML and image types decreases. This indicates that the former arise due to bulk transfers rather than interactive browsing.

The general content-type distribution is essentially unchanged when considering the APR09 trace. However, the fraction of non-flash-video (video/flv) video content (9%) increases, while audio content decreases. Moreover, the fraction of unknown content types from the automated software site falls to 7.5% in APR09. We also confirmed that the presented results are not subject to day-of-week effects by comparing them with results from WEEK trace.

Drawing upon recent data from a major US broadband provider, Erman et al. [41] also report similar content type distributions. They find that video content corresponds to 32% of HTTP traffic, and compressed file downloads, e.g., RAR, for 16% of traffic.

When separating lines with and without P2P protocol usage, we find that the content-type distribution for non-P2P lines closely matches the overall one. However, lines that use P2P have a smaller fraction of flash-video (20%) and RAR archives (11%), and a larger fraction of unidentified content-types (25%) We note that 28% of this unidentified traffic is served from CDNs and 8% from a Direct Download Provider.

5.2.2 Distribution Across Domains

Next we examine the distribution across domains, presenting the results for the SEP08 trace in Table 5.2. We base our analysis on extracting the second-level domain from the HTTP Host header. We find that the byte distribution per domain fairly closely matches a Zipf distribution, per Figure 5.5. The top 15 domains account for 43% of all HTTP bytes. Since flash-video is the most voluminous content-type, it is not surprising to find sites offering videos among the top domains, and indeed most of the traffic to/from these video portals has type video/flv.

Table 5.2: **Top HTTP domains (anonymized) for trace SEP08**

Rank	Domain	Fraction of Traffic
1	Direct Download Provider	15.3%
2	Video portal	6.1%
3	Video portal	3.3%
4	Video portal	3.2%
5	Software updates	3.0%
6	CDN	2.1%
7	Search engine	1.8%
8	Software company	1.7%
9	Web portal	1.3%
10	Video Portal	1.2%

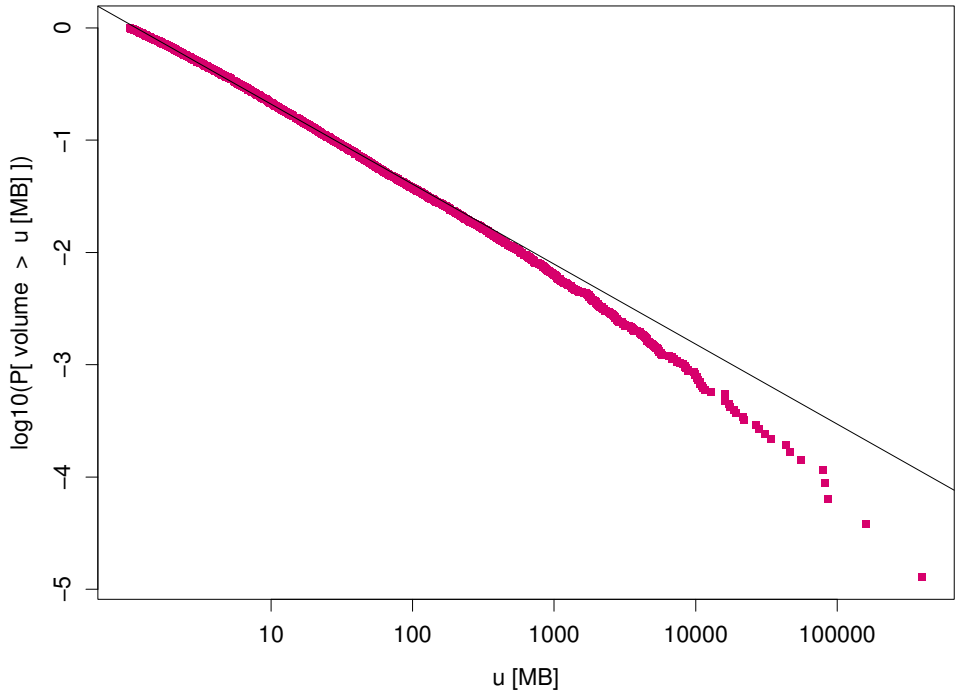

Figure 5.5: **CCDF of HTTP volume per domain, for domains with >1 MB of total traffic for trace SEP08.**

Table 5.3: **Top HTTP user-agents by volume for SEP08.**

Rank	User-agent	Fraction of Traffic
1	Firefox 3	24.6%
2	MSIE 7	20.4%
3	MSIE 6	13.6%
4	Firefox 2	11.9%
5	Unclassified	5.5%
6	Safari	4.3%
7	Network libraries	4.0%
8	Opera	2.8%
9	Streaming clients	2.5%
10	Download managers	1.6%

A Direct Download (DDL) provider also accounts for a significant fraction of HTTP traffic. These DDL providers (also called "One-click providers") host large files for their customers. When a user uploads a file, they receive a (encoded) URL that provides subsequent access for downloading the file. Users can then distribute the URLs to friends or share them in online forums. About 16% of the HTTP traffic involves Direct Download providers, with one provider in particular heavily dominating this traffic (93% of DDL traffic volume). Nighttime traffic exhibits a strong shift towards DDL sites; they account for 24% of HTTP bytes during the 4 AM hour. DDL providers also originate almost 90% of all application/rar bytes.

Similar results hold for the other traces, with only some changes in the lower ranks. Given the small difference in volume for these domains, we attribute such changes to normal day-to-day differences rather than long-term trends.

5.2.3 User-Agent Popularity

To assess the popularity of different types of web clients, we extract the User-Agent headers from the HTTP requests, group them into broader categories, and then rank these categories by transfered volume. We group user-agents that we cannot classify, and requests lacking a User-Agent header, as "Unclassified". Table 5.3 shows the results. We can attribute more than 82% of HTTP traffic to traditional Web browsers, with Firefox and Internet Explorer each having a share of approximately 35% each, while Safari and Opera only contribute 6% and 3% of HTTP traffic. We also crosschecked with the results described above to verify that a large fraction of the traffic due to these traditional web clients involves well-known domains. We do

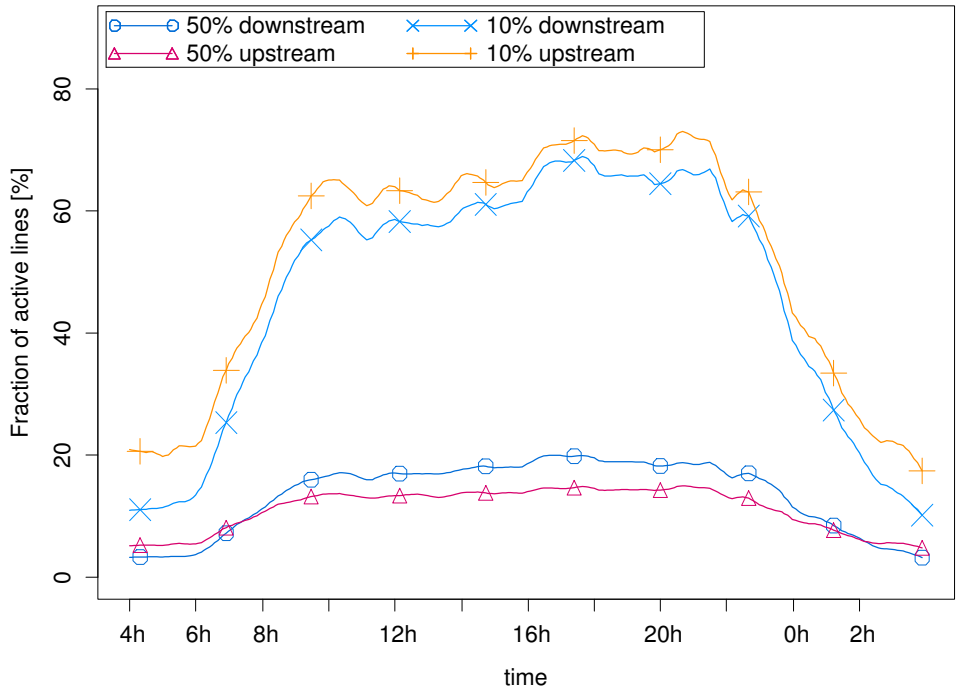

Figure 5.6: **Fraction of active lines using 50%/10% of their available upstream/downstream bandwidth at least once per 5 minute bin (smoothed) for SEP08.**

not see a significant volume contribution by advertised P2P clients. Further, even if such P2P traffic falls into the "Unclassified" bin, it represents little in terms of overall volume. Therefore, in our dataset we do not observe a large proportion of P2P systems running on top of HTTP, unless they employ mimicry of well-known browsers, and also manipulate content types and domains.

5.3 Achieved Throughput

Next, we examine how many lines actually utilize their available access bandwidth across a substantial period of time. We count the number of transfered bytes per DSL line across 1 sec bins and then calculate the throughput per bin. We call a line *active* if it sent at least one packet, or received at least 5 KB, in each bin. We then compare these results to the available access bandwidth for each DSL line,

determining how many lines exceeded 10% or 50% of their bandwidth for at least one second during a given 5 min period.

Figure 5.6 shows that most lines use only a small fraction of their bandwidth. Less than a quarter of the active lines exceed 50% of their bandwidth for even *one second over a 5 minute* time period. However, during the day we observe 50–70% of active lines achieving at least a 10% bandwidth utilization. These results are consistent with findings from Siekkinen et al. [105].

To gauge whether there is a principle network limitation on obtainable performance, we analyzed the achieved throughput per unidirectional flow, distinguishing flows by their application-layer protocol. To do so, we constructed the equivalent of NetFlow data from our packet traces, using an inactivity timeout of 5 sec. Figure 5.7 shows the distribution of the achieved throughput for downstream flows, given they transfered at least 50 KB. We observe that HTTP and NNTP achieve throughputs an order of magnitude larger than those for P2P and unclassified traffic (note the logarithmic scale). We also find that other DPD-classified traffic, as well as traffic on well-known ports, achieves throughput similar to that for HTTP and NNTP. These findings suggest that a portion of unclassified traffic is likely P2P. For flows with more data ($>$ 500 KB), the difference in throughput actually increases slightly. Furthermore, we see that the throughput for all of these larger flows increases as well. When looking at flows in the upstream direction (not shown) we find that all applications achieve similar throughput.

Some P2P applications open multiple parallel connections in order to download content from several peers at the same time. To analyze this behavior, we constructed *hostflows*. For each DSL-line we aggregate all flows of size > 50 KB that overlap time-wise into one hostflow. We calculate the throughput of such a hostflow by dividing the total number of transfered bytes by the total duration. We plot the distribution of hostflow throughput for the downstream direction in Figure 5.8. The plot reveals that the difference in achieved throughput between P2P applications and client-server-based applications remains. As such, the upstream capacity of other peers combined with application restrictions effectively throttles P2P transfers. Interestingly, we find that NNTP performs even better when considering host flows, yet still fails to fully utilize the available access bandwidth. This is most likely a result of users using a customized NNTP client that uses multiple connections for bulk download rather than a traditional newsgroup reader [61].

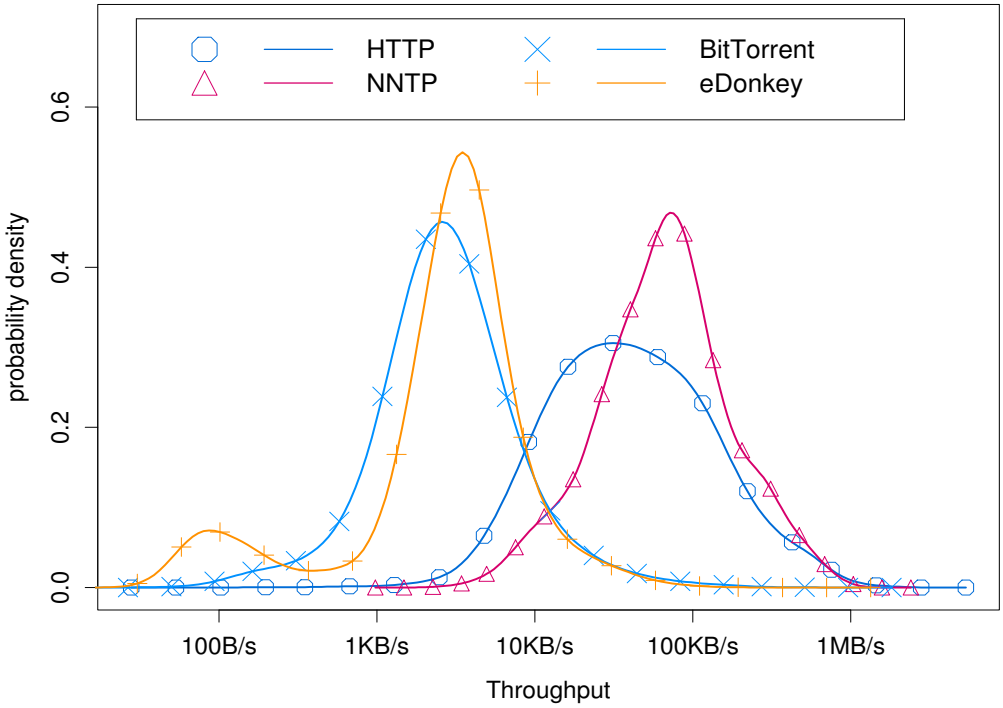

Figure 5.7: **Achieved throughput of downstream flows with size >50 KB by application protocol for SEP08.**

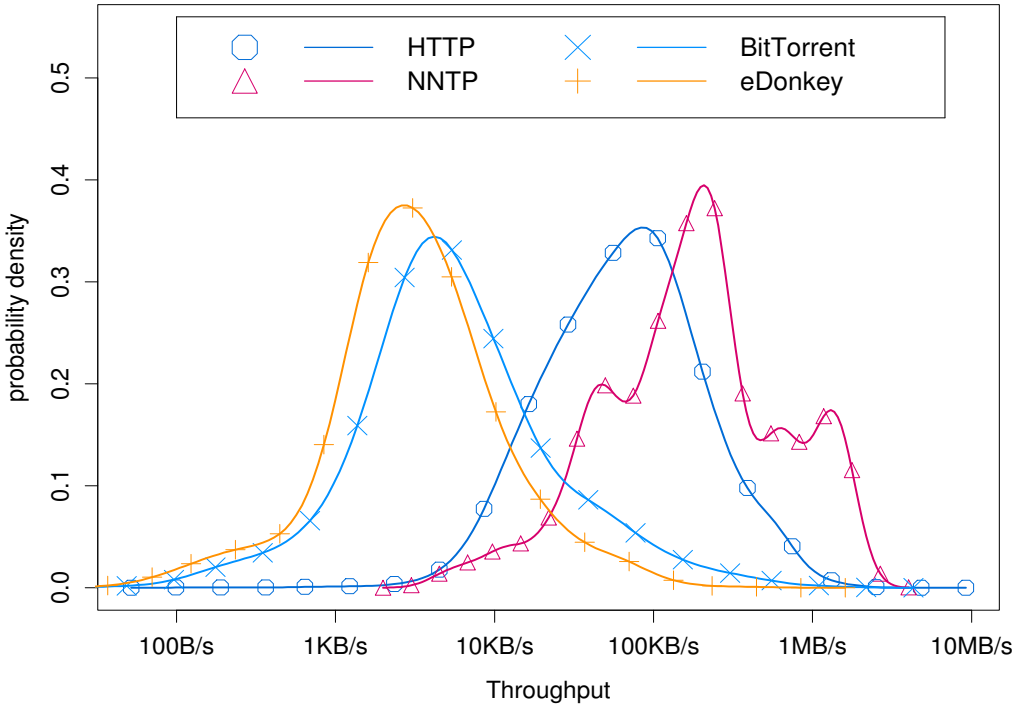

Figure 5.8: **Achieved throughput using downstream hostflows for trace SEP08 by application protocol. Only includes indivual flows with size >50 KB.**

5.4 Summary

We examined usage of different applications and their impact on overall traffic. We observed that P2P no longer dominates in terms of bytes. Rather, HTTP once more carries most of the traffic, by a significant margin (>50%). We further found that NNTP has a significant traffic share. While we used Bro's DPD [36] to identify applications, we also examined the efficacy we would obtain from a simple, purely port-based approach for application classification, finding it works quite well for our datasets, due to the prevalence of HTTP, NNTP, and streaming applications. It does not work as well for P2P, however.

To understand why HTTP is again the dominant application, we looked at a number of facets of its usage. We found that flash-video, the format used by video portals such as `youtube.com` and news sites, contributes 25% of all HTTP traffic, followed by RAR archives. The latter are mostly downloaded from Direct Download providers associated with file-sharing. We did not find a significant share of HTTP traffic attributable to P2P protocols or application protocols using HTTP as a transport protocol.

We also observed that connections from client-server applications, like HTTP and NNTP, achieve an order of magnitude higher throughput than P2P connections, yet still fail to fully utilize the available access bandwidth.

We note that a number of these results agree with those of other contemporaneous studies of Internet traffic [41, 69, 98], suggesting that the trends are representative for a significant fraction of the Internet.

Chapter 6

Transport Protocol Features

We next delve into exploring transport protocol layer features of residential broadband traffic. We start this analysis with investigating the prevalence of TCP options and TCP configurations, in order to get an overview of possible performance limitations due to TCP option usage. We then explore the factors that affect the performance that users experience.

To simplify the presentation, we focus our discussion on SEP08. However, we verified our results across all traces (WEEK, APR09, AUG09) and explicitly point out differences. In particular, we use the 14 samples from WEEK to verify that there are no dominant day-of-week or other biases apparent in the 24 h traces.

6.1 TCP Characteristics

We start with exploring which of the various TCP options and configurations we see in actual use. Doing so allows us to calibrate our expectations with regard to TCP throughput performance. We limit our analysis to connections that transfer some actual TCP payload, which excludes a large number of unproductive connections caused by backscatter, scanning, or other establishment failures. The excluded connections contribute about 0.1% of all bytes, but amount to 35% of all connections.

To compare our results to previous studies, we need to determine the usage of options on a *per-host* basis. However, unlike previous studies we have a significant number of DSL lines using NAT (see Section 4). Since we cannot identify the originating host for each TCP connections, we assess option usage in two ways. The first technique considers each DSL line identifier as a single host, and attributes any options observed in packets associated with the line to that host. Doing so obviously undercounts the number of hosts. For the second approach, we assume that each distinct TCP option set represents a distinct host. This likely overcounts

the number of hosts, so by employing both strategies we can bracket the ranges for host-based use of various TCP options.

Window Scaling

Window Scaling enables efficient data transfer when the bandwidth-delay product exceeds 64 KB. We find window scaling advertisements in 32–35% of the SYNs in our dataset, with 4% of the connections failing to successfully negotiate the use of window scaling. When focusing on only connections transferring more than 50 KB, we find only a small change, with 34–38% successfully negotiated window scaling. Finally, we observe that 45–62% of the *hosts* we monitor advertise window scaling (across traces and across our under- and over-estimates for host count). In contrast, Medina et al. reported that 27% of the observed client hosts advertised window scaling in early 2004 [77]. Of those advertisements, 97% were found to be zero (i.e., the client advertises the *ability* to scale windows, but not the desire to do so). In our dataset, we do not find a predominance of scale factors of zero; most scale factors are in fact non-zero, and cover a wide range. Even with our rough counting of hosts, we can see that use of larger windows has become more routine over the past 5 years.

TCP Timestamp

Timestamps help TCP to compute more accurate round-trip time estimates, and serve to disambiguate old packets from new ones in very high-speed transfers. We observe timestamps advertised in 11–12% of the connections in our dataset, with 8% of the connections ultimately negotiating their use. We further observe that 21–39% of the *hosts* (across traces and host-counting methods) advertise timestamps, versus 22% as observed by Medina et al. [77]. Further, Veal [120] probed a variety of web servers and concluded that 76% of the servers will use timestamps when requested by the client.

Selective Acknowledgment (SACK)

SACK facilitates more effective recovery from lost data segments. We find that 97% of connections in our dataset advertise support for SACK, with 82% of the connections successfully negotiating its use. In addition, we observe that roughly 9% of the connections that negotiate SACK have at least one instance whereby a receiver uses SACK to report a discontinuous arrival (either due to loss or re-ordering). Finally, we observe 82–94% of the hosts we monitor advertising SACK

(across traces and host-counting strategies). Medina et al. reported that in 2004 88% of the clients attempted to use SACK [77], and that active probing found roughly 69% of successfully contacted servers supported SACK.

Maximum Segment Size (MSS)

The MSS governs the largest data segment a TCP sender will transmit. Across all TCP traffic, we find advertised values in the 1300–1460 byte range in 98% of the connections. These values arise from the very common 1500 byte Ethernet MTU, minus space required for TCP/IP headers, as well as space for additional tunneling headers.

Explicit Congestion Notification (ECN)

ECN enables routers to signal conditions of congestion without necessarily employing packet drops. We find virtually no support for ECN, observing only a handful of monitored hosts (no matter how they are counted) advertising support for it in their SYN packets.

Summary

We find that usage of performance improving TCP options varies considerably. SACK enjoys widespread deployment and use; window scaling is quite common in terms of both support and effective (non-zero) employment; ECN sees almost no use.

6.2 Performance/Path Characteristics

We now turn our attention to factors that affect the performance that users experience—spanning network effects, transport protocol settings, application behavior, and home networking equipment.

In a previous study, Dischinger et al. [35] used active measurements to probe 1,900 broadband host from 11 major providers in Europe and North America. They found that the last-mile predominates as the performance bottleneck and induces high jitter in the achievable throughput. They also found that broadband links have large queuing buffers of several hundred to several thousand ms, and that 15% of last-mile RTTs exceed 20 ms. However, they do not compare access versus

remote contributions to RTT. While their study covers a more diverse set of hosts, our approach leverages capturing all activity of residential hosts.

Jiang and Dovrolis [57] estimated TCP RTTs from passive measurements of unidirectional packet data using SYN – SYN/ACK – ACK handshakes and a slow-start based approach. They found that 90–95% of connections have RTTs < 500 ms at various academic links. Aikat et al. [4] examined the variability of RTTs *within* a connection using data from the University of North Carolina. They report that a striking 15% of TCP connections have median RTTs >1 s. However, their analysis does not take delayed ACKs into account. Fraleigh et al. [45] analyzed packet level traces from the Sprint backbone from 2001, finding that the median RTT never exceeded 450 ms across their 9 traces. Only 3 traces had median RTTs >300 ms, while 6 traces had median RTTs of <50 ms.

Siekkinen et al. [105, 106] analyzed performance limitations experienced by ADSL users using passive measurements of approximately 1,300 DSL clients. They found that most users do not utilize the available bandwidth, and that most traffic is application-limited—particularly for P2P applications, which often actively limit the transfer rate. Network limitations like congestion or TCP windows only affected a small number of transfered bytes.

Zhang et al. [135] analyzed Internet flow traces from various access, peering, and regional links within a Tier-1 provider in 2002 to understand from where performance bottlenecks arose. They found that the most frequent performance limitations were network congestion and advertised receiver window sizes.

Given this context, we now turn to an analysis of performance limitations in our 2008/2009 residential traces.

6.2.1 TCP Performance Limitations

TCP's advertised window can have a significant impact on performance, as the window must equal or exceed the bandwidth-delay product for a connection to fully utilize the network path's capacity. If too small, the data sender must pause and wait for ACKs before sending additional data, whereas with a large enough window data can steadily stream. We use the access bandwidth to compute bandwidth-delay products for all connections and find that in the downstream direction, 44% of all connections that transferred at least 50 KB have a bandwidth-delay product that exceeds the maximum advertised window, but this proportion drops to 15% for the upstream direction (which due to bandwidth asymmetry does not require as large of a window).

We find that the maximum advertised window observed per connection tends to be fairly small, with a median across all connections in our dataset of 64 KB. Interestingly, the use of window scaling does not significantly affect advertised window size; the median for such connections increases only slightly, to 65–67 KB. However, the 75^{th} percentile for connections with window scaling is roughly 190 KB, as opposed to the limit of 64 KB imposed by a lack of window scaling.

We note, however, that connections with small advertised windows might in fact have their performance more significantly limited by TCP's response to congestion. We assess loss/reordering events by checking whether a sender ever fails to send monotonically increasing sequence numbers. Loss plays a key role in achievable TCP performance [75, 83], and TCP can confuse reordering for loss [18], causing it to perform congestion control actions that hinder performance. We find that roughly 10% of TCP connections experience such events. Furthermore, 33% of connections that transfer $>$ 50 KB experience loss or reordering. These rates are consistent with the observation that 9% of connections that negotiated SACK actually exchanged a SACK block, as did 30% of connections that transfered at least 50 KB. In addition, we find that about 1% of connections required SYN retransmissions in order to successfully establish.

Finally, we find that at some points the receiver's advertised window "closes" (drops to zero). Generally, this behavior indicates that the receiving application has failed to drain the operating system's TCP buffer quickly enough, and therefore TCP must gradually advertise less available buffer. As the advertised buffer space decreases, the sender's ability to keep enough data in flight to fully fill the network path diminishes. We find that for 4% of the downstream connections the advertised window drops to zero, while this phenomenon occurs for 3% of the upstream connections.

6.2.2 Round-Trip-Times (RTT)

We gathered our measurements at the ISP's broadband access router, which is the first IP router that traffic from the local hosts encounters. We can therefore divide the end-to-end RTT that the residential connections experience into a local component, measured from our monitor to the end system and back, and a remote component, from our monitor over the wide-area Internet path to the host at the other end of the connection.

We estimate TCP RTTs using the connection setup handshake (SYN, SYN/ACK, ACK) [57], ignoring connections with SYN or SYN/ACK retransmissions, and connections in which the final ACK carries data (which can indicate that an "empty"

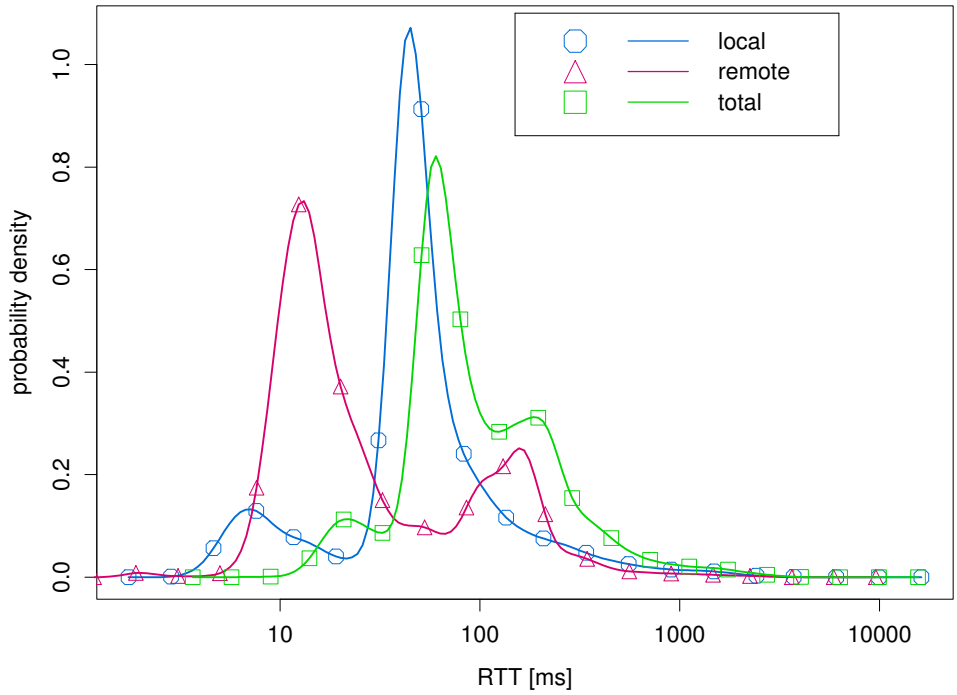

Figure 6.1: **TCP round trip times for trace SEP08.**

ACK has been lost). Figure 6.1 shows the smoothed probability distribution of the RTTs. We found it quite surprising to observe that in many cases the local RTT exceeds the remote RTT, i.e., *the time to simply get to the Internet dominates over the time spent traveling the Internet.*

The difference manifests itself throughout most of the distribution. For example, the median, 75^{th}, 90^{th}, and 99^{th} percentiles of the local RTTs are all substantially larger than their remote counterparts, and we find that 1% of local RTTs exceed 946 ms, while for remote RTTs the corresponding delay quantile is only 528 ms. The 99^{th} percentile of total RTT is 1328 ms, with a 90^{th} percentile of 278 ms and a median of 74 ms. While RTTs are often fairly low, we also observe several cases for which the local RTT reaches values in the 2–6 sec range and beyond.

Local RTTs follow a bi-modal distribution, with one peak at 7 ms and another, larger one at 45 ms. This is consistent with the fact that most DSL lines use *interleaving* [56, 62], which increases delay, while a smaller number of the DSL lines use the "fast path" feature, which does not contribute any significant delay.

Remote RTTs exhibit three modes, at 13 ms, 100 ms, and 160 ms, with the lat-

ter two somewhat blurred in the plot. Likely these modes reflect the geographic distribution of remote hosts (e.g., Europe, US East coast, US West coast).

6.2.3 Impact of Access Technology

The not infrequent appearance of large local RTTs led us to investigate their possible cause. Typically, large RTTs reflect large queuing delays. Indeed, Dischinger et al. [35] found that residential broadband links can exhibit queuing delays of several seconds when a DSL line is fully utilized.

Manual inspection of sequence number plots of some connections with large RTTs (>1000 ms) indeed shows such queues building up. We therefore checked whether those lines utilized their access bandwidth during these events. We found, however, that this is *not* always the case: while we often see significant traffic on these DSL lines, they do not necessarily utilize their upstream or downstream bandwidth fully. A more detailed manual analysis reveals other effects, too, such as RTTs within a connection suddenly jumping by an order of magnitude.

One possible cause could be wireless links in users' homes, given the plausibility of a large fraction of broadband users employing 802.11 wireless to connect their computers to the Internet. In densely populated, urban areas, users often "see" numerous wireless networks, and therefore can experience non-negligible contention for the medium.

To assess this hypothesis, we used several DSL links (1x 8000 Kbps and 3x 2000 Kbps downstream) to estimate upstream and downstream throughput and queuing delays using active measurements done with the nettest tool.

Using wired connections, we are able to fully utilize the DSL link's bandwidth. When using wireless connections, the achieved throughput often drops to 400–1000 Kbps. In both cases, we experience queuing delays of several seconds. However, the reduced throughput when using wireless access causes the queue to start building up at lower rates. In addition, while we were unable to saturate the 8000 Kbps link[1] with a wired connection, and therefore had low or imperceptible queuing delay, using wireless the queuing delay still rose to several seconds.

These results show that wireless networks can have a significant impact on the achievable throughput. In particular, 11 Mbps wireless cards and wireless connections in areas with many other wireless senders, and/or with poor link quality, face significant performance degradation. We verified that wireless connections,

[1] Due to a bottleneck in the Internet between the DSL line and the measurement server

in uncontested environments and with current 54 Mbps wireless devices, offer the same throughput and queuing delay as wired connections.

6.3 Summary

We analyzed transport protocol characteristics in terms of TCP options. We found that window scaling and SACK have become more popular since Medina et al.'s previous study [77], with SACK employed by more than 90% of clients. Window scaling is also often used, but does not in fact result in larger advertised receiver windows.

We assessed performance and path characteristics of TCP connections, noting that most DSL lines fail to utilize their available bandwidth. Examining TCP round-trip-times, we found that for many TCP connections the access bandwidth-delay product exceeds the advertised window, thus making it impossible for the connection to saturate the access link. Our RTT analysis also revealed that, surprisingly, the latency from the DSL-connected host to its first Internet hop dominates the WAN path delay. This discrepancy can however be explained by ADSL's interleaving mechanism. We found that WAN delays are often as little as 13 ms, but local RTTs not infrequently exceed 1000 ms, a phenomenon that is likely caused by the use of wireless equipment in the customers home and ensuing contention on the wireless hop.

Chapter 7

Mobile Hand-held Device Characterization

Today, advanced mobile hand-held devices (MHDs, e. g., iPhones and BlackBerrys) are very popular. MHDs have evolved rapidly over the years—from pure offline devices, to cell phones with GSM data connectivity, to 3G devices, and universal devices with both cellular as well as WiFi capabilities. Their increased graphics and processing power makes these devices all-in-one PDAs and media centers. Today's MHDs can be used to surf the Web, check email, access weather forecast and stock quotes, and navigate using GPS based maps—to just name some of the prominent features. This increase in flexibility has caused an increase in network traffic. Indeed, cellular IP traffic volume is growing rapidly and significantly faster than classic broadband volume [130].

We cast a first look at Internet traffic caused by mobile hand-held devices. We use our anonymized residential DSL broadband traces, spanning a period of 11 month, to study MHD behavior and their impact on network usage. We are thus able to observe the behavior of MHDs when they are connected via WiFi at home and compare their traffic patterns to the overall residential traffic characteristics. Some devices (most notably the iPod touch and iPhone) require WiFi connectivity rather than cellular connectivity for some services. Other services are more likely to be used via cellular connectivity due to user mobility, e. g., looking up directions on Google Maps, while walking around town or driving. Although we only focus on residential MHD usage and not MHD usage in cellular networks, our analysis gives first insights into what kind of services users are interested in when they are at home and have access to all services. This information is crucial for 3G cellular providers to anticipate usage patterns and future traffic growths.

The remainder of this chapter is structured as follows. In Section 7.1 we present our methodology, Section 7.2 presents our results. In Section 7.3 we discuss related work before we conclude this chapter in Section 7.4.

7.1 Methodology

We base this study on the anonymized residential packet-level traces, annotated with anonymized DSL line card port ID. We use traces SEP08, APR09, AUG09a, AUG09b (see Chapter 2 for details). The line card port ID annotation enables us to uniquely distinguish DSL lines since IP addresses are subject to churn and as such cannot be used to identify DSL lines (see Section 4.2).

7.1.1 Identifying MHDs

To understand how MHDs are utilized we need to identify not only their presence in our traces but also their contributions. This is non-trivial as MHD users commonly do not just operate the MHD over their DSL-line but also/mainly computers or set-top boxes. Note, that all devices active via one DSL-line usually share a single IP address. Therefore, we rely on network signatures which we gather by observing and recording MHD behavior in a controlled environment.

Among the currently popular MHD devices are Symbian-based phones, BlackBerrys, iPhones and iPods, Windows Mobile based phones, and Google Android phones [124]. We collected manual traces using tcpdump for all device types except BlackBerrys[1]. With each device we performed the following set of actions using a wireless access-point for data collection: connecting to the access-point, accessing several Web sites, watching videos on YouTube, using other mobile applications like Weather and Stocks, checking and sending emails, using Facebook, and updating/installing mobile applications on the MHD.

Analyzing these manual traces reveals that HTTP dominates the protocol mix and that most mobile applications, including Weather, Stock quotes, AppStore, and YouTube, use HTTP. From our manual traces we extract a list of HTTP user-agent strings for each device and OS combination.[2] We further augment this list by well-known strings from other mobile devices, e.g., BlackBerrys. This captures the strings of the standard applications. However, it is not possible to compile a list of all user-agent strings that MHD application writers may use. However, since most developers rely on standard libraries, we add patterns for these libraries. For example, most applications for Apple devices use the Apple CFNetwork library for communication and CFNetwork usually adds its name and version number to the

[1] Manual trace collection was performed with Google's G1 (Android 1.5), Apple's iPod touch (iPhone OS 2 & iPhone OS 3), HP's iPaq (Windows Mobile), HTC Touch 3G (Windows Mobile), Nokia 810 (Maemo Linux), and Nokia E61 (Symbian). Thanks to all device owners.

[2] We note that these MHD user-agent strings differ from user-agent strings used by PCs/Macs.

end of user-agent strings. While Mac OS X also uses CFNetwork, the version numbers used by the iPhone and Mac OS X are disjoint and we can distinguish them. Based on this collection of user-agent strings we create patterns for *(i)* identifying DSL lines that "host" MHDs and *(ii)* identifying and classifying MHD usage of HTTP.

7.1.2 Application Protocol Mix

Finding signatures for identifying non-HTTP traffic caused by MHDs is more difficult since most other application protocols, e. g., POP, do not add device related information to their user-agent strings. Furthermore, they may use encryption.

One obvious approach for overcoming this limitation is to assume that MHDs and regular computers are used consecutively, i. e., not used at the same time at the same DSL line. Based upon this assumption one can classify all traffic after a HTTP request from a MHD on a DSL line as MHD traffic (relying on a timeout). However, we show in Section 7.2.1 that the underlying assumption is incorrect. A majority of the lines shows contemporaneous activity from MHDs and regular computers.

Therefore, we take advantage of another characteristic of network devices—their IP TTLs. The default IP TTLs of popular MHDs differ from those of the most commonly used home OSes. The default TTL of iPhones/iPods and Macs is 64, Symbian uses 69, while Windows uses 128. This enables us to separate MHD usage from regular PC usage for some combinations of OSes. While we cannot distinguish iPhones/iPods from Macs or Windows Mobile from Windows we can use IP TTLs to separate the other combinations. Our observations show that the majority of home OSes is Windows while the majority of MHDs are iPods or iPhones. In order to separate those, we first select all DSL lines for which *every* HTTP request with a TTL[3] of 64 or 69 is originated by a MHD (as identified via the user-agent). The assumption is that all traffic on these lines with TTL 64/69 is then caused by a MHD. Thus, we can then use Bro's DPD [36] on this traffic to get a first impression of the application protocol mix of MHDs. Since this approach excludes lines with certain combinations of MHDs and regular computers we are left with 54–59 % of the lines with MHDs. In addition, if the activity of the regular computer does not include HTTP we might misclassify its traffic. We note that we use this heuristic only for analyzing the application protocol mix; we use user-agent strings for all other analyses.

[3] We take NAT devices and our hop distance to the end system into account.

Table 7.1: **Overview of MHD pervasiveness.**

Name	Size	# MHDs	MHD HTTP Traffic Volume	MHD HTTP Traffic % of HTTP
SEP08	≈4 TB	≈200	≈2 GB	0.1 %
APR09	≈4 TB	≈400	≈9 GB	0.4 %
AUG09a	≈6 TB	≈500	≈15 GB	0.6 %
AUG09b	≈5 TB	≈500	≈15 GB	0.7 %

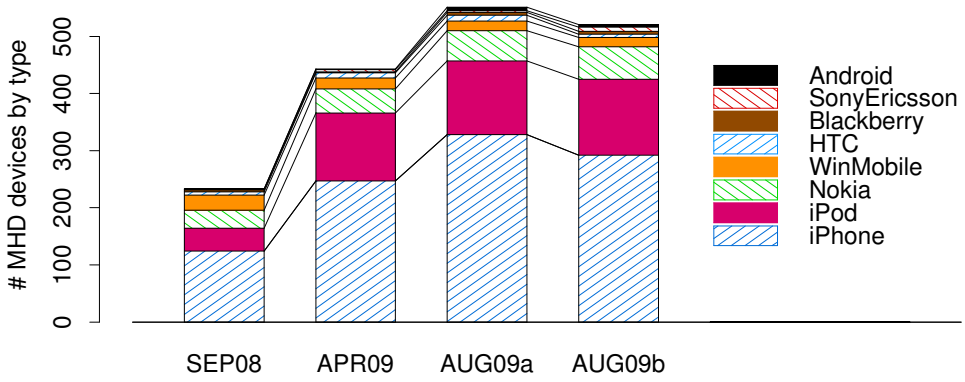

Figure 7.1: **Popularity of MHD device types**

7.2 Results

After reporting on the pervasiveness of MHDs we focus on their protocol mix. Then we characterize MHDs' HTTP traffic, analyze mobile application usage, and present results on iTunes and AppStore usage.

7.2.1 MHD Pervasiveness

On a significant number of the DSL lines we observe traffic from MHDs (see Table 7.1). Indeed, in the most recent trace, AUG09, 3 % of active lines have MHD activity. Moreover, the contribution of MHDs to the observed HTTP traffic is also substantial (up to 0.7 % of HTTP bytes). This indicates that some MHD users may find it more convenient to use their mobile devices at home even if they have a regular computer as well. Note, HTTP's share of overall traffic volume is 50–60 %, see Section 5.1.

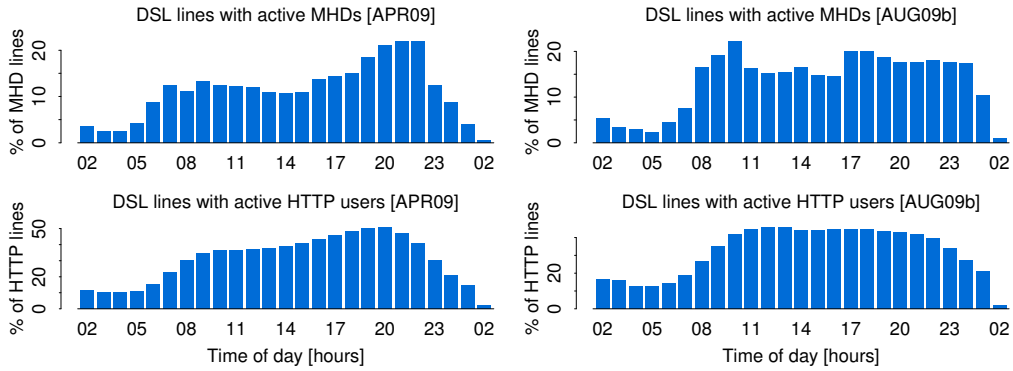

Figure 7.2: **Number of lines with MHD activity vs. number of lines with HTTP activity for APR09 and AUG09b.**

There is a strong temporal trend underlined by the rapid growth in the number of lines with MHDs' activity and in the MHDs' HTTP traffic volume. The number of lines with MHDs almost doubled between SEP08 and AUG09. The HTTP traffic volume from MHD grew sixfold while the overall traffic volume increased only slightly and the overall HTTP volume increased by 22 % at our vantage point.

Figure 7.1 shows the distribution of active devices types for all traces. We observe that Apple devices (iPhone and iPod touch) clearly dominate, both in terms of number of lines and traffic volume (not shown). They account for 86–97 % of MHDs' HTTP traffic and 71–87 % of the devices. This is in contrast to the market shares of the devices [124]. Possible explanations are that Apple users *(i)* find their device very convenient even for home use and/or *(ii)* are looking for a multimedia device that "also works as a phone". Indeed, the iPod touch is an iPhone without phone capability. We note that starting from APR09 the number of lines with iPods outnumber the number of lines with all non-Apple MHDs combined.

We already pointed out that we have a substantial number of DSL lines "hosting" MHDs. Now we want to illustrate how the use of MHDs is distributed over the course of a day. To determine how the use of MHDs is distributed across time we plot the relative number of lines with active MHDs per hour (top) and the percentage of lines with HTTP traffic per hour for APR09 and AUG09b in Figure 7.2. We see that MHDs are used throughout the day. While we see a similar behavior when looking at overall HTTP traffic, we see that MHD usage has a stronger pick-up in the morning (AUG09b even shows a peak). Overall HTTP traffic on the other hand slowly ramps up during the day. Again the convenience of using the mobile device may be a possible explanation. Users can use them to check their emails or the weather when "starting their day". The low byte contribution of mobile devices

in the morning hours supports this claim (figure not shown).

Next, we examine if MHDs and regular computers are used consecutively or whether they are used contemporaneously. To assess this, we compute for each DSL line and for any two subsequent HTTP requests their inter-request-times (IRTs) and label them as *(i)* both from MHDs, *(ii)* both from non-MHDs, or *(iii)* from MHD and non-MHD. We then compute the shortest IRT for mixed activity (MHD and non-MHD) per DSL lines. We find that for 33–39 % of MHD lines the shortest IRT for mixed MHD/non-MHD activity is less than one second. For IRTs of less than one minute (five minutes) up to 62 % (72 %) of the lines have mixed activity.

7.2.2 Application Protocol Mix

While our approach for analyzing the application protocol mix of MHDs is limited (see Section 7.1.2), it still gives us a first impression of MHDs' traffic composition. We find that HTTP clearly dominates across all of our traces. HTTP contributes 80–97 % of all MHD bytes. Email related protocols account for more than 9 % of the bytes in SEP08, 2.3–2.5 % in APR09 and AUG09a. However, it drops to 0.2 % in AUG09b, most likely due to a different usage patterns on weekends. In general, no other protocol has a traffic share of more than 1.5 % with the exception of 13 % unclassified traffic in APR09, and 15 % RTMP streaming in AUG09a, caused by only a handful of MHDs.

7.2.3 MHD Web Traffic

Given that HTTP traffic accounts for the vast majority of MHD traffic, we now examine it more closely to characterize its usage and how it differs from overall HTTP usage. We use anonymized HTTP headers and identify HTTP requests from MHDs using user-agents strings as discussed in Section 7.1.1.

To identify the content-type of each transfered HTTP object we join information from the Content-Type HTTP header field and an analysis of the initial part of the HTTP body using libmagic, per Section 5.2. We then group the content-type into a handful of categories. We classify downloads of mobile applications as apps, video and audio content as multimedia, and images as web-browsing, because the latter are usually an integral part of Web pages.

Figure 7.3 shows the HTTP content type categories for MHDs and compares them with all HTTP traffic. We find that multimedia content is the most voluminous MHD content-type across all traces followed by application downloads. Interestingly, XML objects are also common. They account for 2–5 % of the transfered

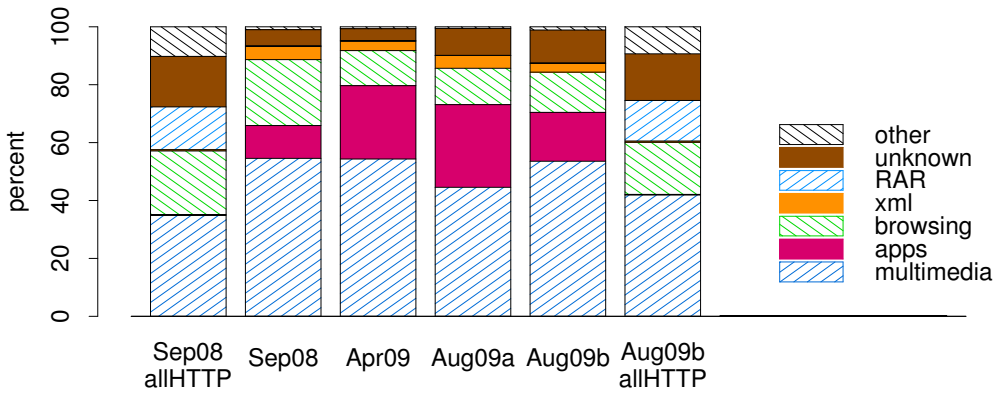

Figure 7.3: **HTTP content type categories by volume. Comparing MHD traffic to all HTTP traffic.**

HTTP bytes. XML is used by many applications for status and data updates, e.g., weather forecasts, stock quotes, and sport results. Surprisingly, Web surfing itself (text based content-types and images) is only the third largest category contributing less than 14 % in the 2009 traces (23 % in SEP08).

Comparing these results to all HTTP traffic (see Section 5.2) we find that downloads of mobile applications and XML contribute a significantly smaller fraction to the content type mix. In contrast the volume contributed by RAR archives to all HTTP traffic is significantly larger. Browsing is a bit more prevalent in all HTTP traffic (18–22 %). Multimedia content is the biggest contributor for both. However, for all HTTP traffic flash-video is the most popular video codec, while MHDs use MPEG coding, which is likely influenced by the large number of mobile Apple devices that do not support flash-video.

The volume share per DNS domain reflects the distribution of MHD content-types. Apple's `apple.com` is responsible for most of the traffic due to application downloads. Note, only the AUG09a trace shows a significant number of iPhone application downloads from third-party sites rather than the Apple's AppStore. YouTube and Stream.fm are the next most popular domains. For overall HTTP traffic, One-Click-Hosters and video portals are among the top domains by volume.

To answer the question if MHD HTTP traffic characteristics differ from overall HTTP traffic, we compare the distribution of HTTP object sizes. See Figure 7.4 for a plot of the Cumulative Complementary Distribution Function (CCDF) and Probability Density Function (PDF) for APR09. The results for the other traces are similar. We find that both distributions are consistent with a heavy-tailed

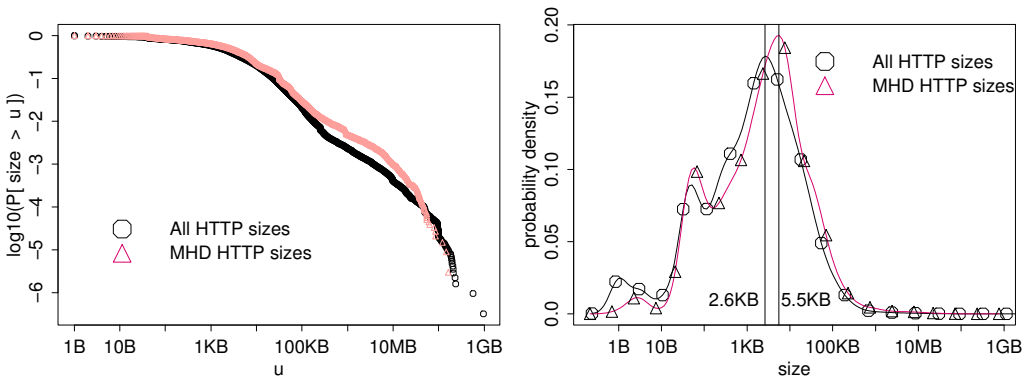

Figure 7.4: **Size of HTTP objects for all traffic and MHD traffic for trace APR09.**

distribution. While the dominating mode of object sizes downloaded by MHDs is larger (see support lines), the tail is heavier for all HTTP traffic.

7.2.4 Mobile Applications

Figure 7.5 shows the popularity of the top MHDs' applications. The most popular application is Apple's browser Safari. Up to 62 % of all devices are using it. This is followed by iTunes (up to 37 %) and Weather (up to 32 %). For non-Apple MHDs we observe that the browser is also the most popular application. Overall we find that Apple's default applications clearly dominate. Surprisingly, the popularity of Maps is relatively low. One possible explanation is that one rarely needs directions while at home. CoreMedia, the media player of iPhones and iPods, is also quite prevalent. This application is responsible for playing videos accessed via the YouTube application or the browser. The YouTube application itself is only used for searching videos, tagging, and navigating within YouTube. Locationd is the wireless positioning system used on Apple devices.

To understand if users take advantage of specialized applications available for popular Web services, we select two Online Social Networks that are popular in our user base: Facebook and StudiVZ. For both OSNs there are specialized applications available for the iPhone/iPod MHDs. We find that roughly half of the users (50 % $\pm$ 10 %) use the specialized applications while the other half continues to use the built-in browser. This relationship is stable throughout our 11 month observation period.

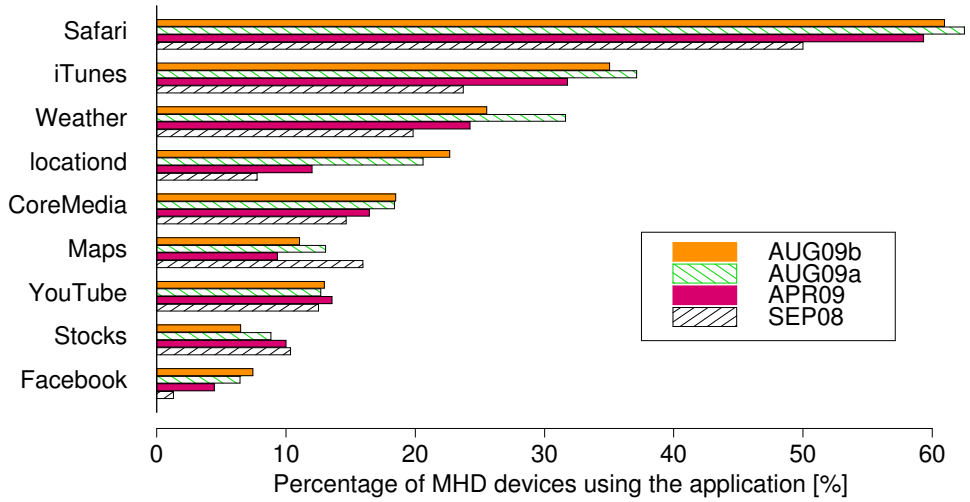

Figure 7.5: **Application popularity by number of MHD devices using a specific application.**

7.2.5 Application and Media Downloads

Given that we are observing traffic from residential DSL lines, we have the ability to evaluate if users use their mobile devices or their regular computer to download mobile applications and/or multimedia content. Due to the prevalence of Apple devices in our dataset we now focus on Apple iTunes store and Apple AppStore.

We find that applications are predominantly downloaded directly to the MHD (see Table 7.2), e.g., more than 70 % of downloads for the 2009 traces. Surprisingly, we see that for AUG09a and AUG09b the total volume of application downloads in terms of bytes is almost the same for regular computers and MHD, i. e., the mean application size is larger for applications downloaded by PC/Macs. A detailed analysis reveals that this is caused by outliers; the median application size is the same for both.

We see a vastly different behavior for media downloads or purchases from Apple's iTunes store. Downloads are almost exclusively done via the regular computers. We see several thousand media files being accessed in the 2009 traces. However, only a handful of downloads are via MHDs which results in a small byte contribution.

Table 7.2: **Downloads from AppStore**

Trace	# Apps available	by PC/Mac Volume	by PC/Mac # Req	by MHD Volume	by MHD # Req
SEP08	3,000	<1 GB	<100	<1 GB	<100
APR09	7,500	<1 GB	>100	>2 GB	>250
AUG09a	70,000	>2 GB	>150	>3 GB	>450
AUG09b	70,000	>3 GB	>150	>3 GB	>400

7.3 Related Work

Only a small number of studies have focused on Internet traffic in 3G mobile or cellular networks. Svoboda et al. [113] analyze various aspects of cellular 2G and 3G (GPRS and UMTS) traffic using anonymized header traces from 2004 and 2005. They study traffic volume per user and protocol mix. In terms of protocol mix, they find that HTTP is the dominant protocol with 40–60 % of traffic. Heikkinen et al. [52] analyze P2P usage from passive UMTS header traces in Finland from 2005–2007. Web traffic accounts for 57–79 % of bytes from mobile hand-held devices, email for 10–24 %, and P2P is not noticeable. Williamson et al. [127] analyze packet/data call event traces from a CDMA2000 network from 2004. They focus their analysis on link-layer behavior, session properties, and user mobility.

Several studies have analyzed TCP performance and low-level traffic characteristics in GPRS and CDMA data networks [16, 70, 129]. Other studies analyze the content requested or available for mobile devices. Using data from 2000, Adya et al. [2] analyze the Web server logs of a major commercial site and study the requests of mobile clients. They find that stock quotes, news, and yellow pages were the most commonly accessed content in their traces. Timmins et al. [115] use active measurements to crawl the Web for sites offering specialized content for mobile devices. Verkasalo [121] studies how Symbian phone features are used by instrumenting the handset. He finds that the camera feature and games are the most common multimedia applications.

Trestian et al. [117] analyzes mobility and web-application usage in a 3G network from a metropolitan area. We on the other hand, focus on stationary usage when MHDs are connected at home via WiFi. Trestian et al. characterize web-application usage by counting the number of HTTP request and find that social networking, music, and e-mail are the most common web. They do not assess how many *users* utilize a particular application, which is the approach we use to characterize application usage.

7.4 Summary

Our analysis of residential broadband DSL lines of a large European ISP shows that there is a significant and increasing number of active MHDs. Our characterization of the traffic shows that MHDs are active on up to 3 % of the monitored DSL lines. We find that iPhones and iPods are by far the most commonly observed MHDs. This has an impact on the most popular mobile applications: Safari (Apple's browser), iTunes, and Weather. The largest fraction by volume of MHD HTTP content is multimedia. Comparing HTTP object sizes of overall and MHD traffic we find that MHD HTTP objects are on average larger. The contribution of MHDs to the overall traffic volume is still small, but rapidly growing, especially compared to the overall traffic growth.

Chapter 8

Residential Network Security

After analyzing traffic characteristics of residential networks, we next turn to malicious network activity and risky behavior in such networks. While conventional wisdom holds that residential users—and particularly home users in rural or developing regions—are responsible for much of today's Internet insecurity, little systematic study has examined whether such views in fact reflect reality. In this chapter we describe observations from monitoring the network activity (anonymized for user privacy) of the 20,000 DSL customers of the European ISP as well and of more than 5,000 users of a community network in rural India. To assess malicious activity we develop a set of metrics, including signatures for known malware patterns and behavioral techniques for finding scanners and spammers. We analyze the relationship between problems flagged by these metrics versus the security awareness of our user populations (anti-virus and OS software updates) and potential risky behavior (accessing blacklisted URLs). Overall we find that both environments have roughly comparable levels of problematic behavior, in both cases indicating only a small fraction of actively malicious hosts. However, we also find quite prevalent risky behavior across our data sets, and that security awareness steps such as using anti-virus updates do not correlate with a lower degree of problematic activity.

8.1 Motivation and Background

Conventional wisdom says that residential users or users of community networks are responsible for much of today's Internet insecurity as they typically lack the means to maintain and secure their systems to the necessary degree. In particular, this reasoning proceeds, this must be the case for rural or developing regions, where the lack of infrastructure and technical expertise further limits the sophistication of their protection. However, so far few systematic studies have examined whether this presumption reflects reality.

In this chapter, we pursue such a study. We examine the network activity of 20,000 residential DSL customers within an European urban area and several 1,000 users of a community network in rural India (AirJaldi [5]). If conventional presumption holds, we should not only see a significantly higher level of malicious activity in these networks than in well-maintained enterprise networks, but also a major difference *between* the two environments. To check this hypothesis, we develop a set of security metrics, including behavioral metrics for identifying scanners and spammers, and utilize signatures for known malware patterns. We apply the metrics to anonymized passive network traces captured within each environment. For comparison, we also perform the same analysis at a large research laboratory that has a track record of effectively protecting its roughly 12,000 systems while maintaining a very liberal default policy.

Furthermore, we analyze the relationship between problems flagged by our metrics and the level of security awareness in our populations. One may expect that users who perform regular OS updates and deploy anti-virus software are less likely to have their systems compromised. Likewise, intuition may suggest that users who partake in particularly risky behavior have more security problems. To check these assumptions, we examine the end-user systems in our data sets for signs of regular operating system updates and anti-virus deployments as well as for contacts to URLs blacklisted by Google's Safe Browsing API [50].

The results of our analysis reveal a number of observations that contradict conventional wisdom, including:

- We find only a small fraction of actively malicious hosts both at the European ISP as well as the rural community network in India.
- Indeed, a comparison of these two environments shows that there is no significant difference in the levels of malicious activity.
- While OS software updates and anti-virus technology are widely deployed we do not find a correlation with a lower degree of malicious activity.
- Likewise, while we observe frequent risky behavior we find that such behavior does not increase the probability of malicious activity as much as one may presume.

We note that our study only corresponds to a first look at the security properties of residential traffic. Since today's malware landscape is large and diverse, identifying the full extend of badness within a large data set is not only daunting but most likely impossible. Still, we believe that our choice of metrics covers a significant share of typical malicious activity, and our study lets us thus challenge assumptions

that people tend to express about the insecurity of residential and community networks.

We structure the remainder of this paper as follows. We describe our data annotations in Section 8.2 before we introduce our methodology in Section 8.3. We then study security awareness and risky behavior in Section 8.4 before investigating malicious activity in Section 8.5. We present related work in Section 8.6. Finally, we discuss our results and present avenues for future work in Section 8.7.

8.2 Data Annotations

We rely on the Time Machine traces collected at the European ISP and LBNL, which span 14 and 4 days respectively (see Table 2.3). In addition, we use the AirJaldi traces (see Table 2.4) for our assessment of malicious activity and risky behavior. See Chapter 2 for a description of the environments and data sets we use. To determine whether either the presence of network address translation (NAT), the amount of customer activity, or operating systems influence the level of malicious activity we see, we annotate our traces with meta information on NAT usage, activity level, and OS.

8.2.1 NAT Usage

NAT is often used in residential settings to connect a single or multiple hosts to the Internet via a single DSL line. In addition, NAT gateways also act as firewalls blocking unwanted incoming connections and scans, thus protecting otherwise vulnerable hosts from infections. We use our NAT detection approach from Chapter 4 to annotate the DSL lines of the European ISP with NAT usage information. For AirJaldi we cannot use our NAT detection approach to annotate the traces as our approach relies on OS diversity and on a small number of hosts per NAT gateway. At LBNL we do not need NAT annotations since in general NAT gateways are not part of the network setup.

8.2.2 Activity Levels

To measure a DSL line's "activity level", we count its number of daily HTTP requests. We then sort all lines by these numbers and label the top 10 % as having *high* activity; the top 50 % of lines has having *medium* activity; and the bottom

10 % as having *low* activity.[1] In addition, we also calculate *cumulative activity* levels by counting the number of HTTP requests up to a given day D, and then applying the labeling scheme to the accumulated values.[2]

We use the same approach to assign activity levels to IP addresses over the 4-day LBNL trace. For AirJaldi we do not compute cumulative values as the traces' durations are less than 40 hr.

We find that the property of a DSL line having *multiple* hosts hidden behind a NAT gateway correlates with higher activity levels: the correlation with high activity is 0.20; with medium activity it is 0.35; with low activity there is negative correlation. However, we do not observe strong correlation between lines *not* deploying NAT and activity levels. Although we find that the correlation is slightly negative for high and medium activity and slightly positive for low activity.

8.2.3 Operating Systems

A common assumption is that exposure to malware varies between operating systems where some systems are perceived to be more secure due to a better architecture or a smaller market share that makes them less attractive targets. To assess the degree to which this is the case, we annotate our traces with the operating system(s) that are used on a DSL line (or IP). To identify operating systems, we use the user-agent strings of HTTP requests. We select user-agent strings from the most popular browsers (Firefox, Internet Explorer, Safari, and Opera) and extract the operating system version from them. We are thus able to identify operating system combinations for more than 90 % of DSL lines of the European ISP. Analyzing the operating system mix, we find that 59 % of lines use only a single version of Windows, while 23 % use several different Windows versions. Macs are present on 7.6 % of lines. To assess malicious activity on Macs, we next select the lines on which only Mac user-agent strings were present but no Windows user-agents. To further ensure that there are no Windows systems on Mac lines, we also exclude lines on which we observe IP TTLs that are consistent with Windows[3]. After this pruning, we find that 2.7 % of all lines have only Macs. We further find that Linux is present on 3.68 % of lines. However, Linux is in general used in combination with other operating systems. Thus, the fraction of lines with only Linux is too small

[1] Note, that this implies that lines with high activity are also included in the medium activity class.

[2] We have also experimented with determining activity levels by measuring the number of outbound *connections*. The results however match those yielded by the HTTP-based metric and we thus do not discuss them further.

[3] Windows used a default TTL of 128, Mac OS X uses 64.

to separately assess malicious activity for them. For the AirJaldi environment we find that we observe too many different operating systems per IP and therefore cannot assess malicious activity based on OSes. At LBNL, we find that less than 60 % of active IPs have user-agent strings with operating system information. We find that 70 % of those IPs use only a single version of Windows, and 2.8 % use multiple Windows versions. We see Macs in use with 19 % of those IPs and Linux with 8.6 %.

8.3 Methodology

To identify DSL lines and hosts that show malicious activity or are infected with malware respectively we use a twofold approach. We use three metrics that each indicate unnatural and malign behavior but are not limited to any particular kind of malware. Namely, the metrics capture *address scanning, port scanning, and spamming*. Furthermore, we use network level signatures for three malware families, Zlob, Conficker, and Zeus. In addition, we study the usefulness of the intrusion detection system Snort [97] with the Emerging Threats [39] rule sets, as a blanket approach. Note, however, that our experience shows (Section 8.3.4) that the latter approach is not usable. In addition to problematic and malicious behavior, we also check if users are security aware (e. g., that they use anti-virus scanners) and whether they exhibit *risky* behavior. For compactness of presentation, we describe our methodology in terms of the European ISP and point out any differences that apply for any of the other environments.

To take advantage of the long duration of our traces, we analyze and report malicious and risk behavior separately for each day of our multi-day traces (ISP and LBNL). We further cumulate the results by accumulating all activity over different trace durations and in total. For example, consider a DSL line that started to be a scanner on day 4. In the daily data sets this line is marked as a scanner on day 4. In the cumulative data sets this line is marked as scanner on day 4 and for *all following days* regardless of whether the line is again acting as scanner on any of the following days or not. The cumulated data sets thus allow us to check how malicious activity changes across longer observation periods, and the cumulative data for the last day of a trace reflects the aggregate behavior over the full observation period. For AirJaldi we do not differentiate between per day and per trace analysis since no trace is longer than 40 hr.

To determine the influence of risky behavior on malicious activity and to understand the relationship between different malicious activity metrics we calculate their correlations and their conditional probabilities in the following way: For

both, the daily and cumulated data sets, we calculate a (binary) vector for each day with one entry per DSL line which indicates if the metric is applicable or not, e.g., the line is identified as scanner, has high HTTP activity, uses anti-virus, etc. This allows us to examine the full correlation matrix of the metrics, where the correlation is obtained by dividing the covariance of two vectors by the product of their standard deviations. If two metrics are independent the correlation coefficient is 0. The correlation is +1 in the case of a perfect positive linear relationship and -1 in the case of a perfect decreasing linear relationship. Other values between -1 and 1 indicate the degree of linear dependence between the variables. The closer the coefficient is to either -1 or 1, the stronger the correlation between the variables. Likewise, the probability of a DSL lines having a property (on a given singe or cumulative day) is:

$$\Pr[property] = \frac{\text{num. lines with } property}{\text{num. active lines}}$$

and the conditional probability is defined as usual as

$$\Pr[property1|property2] = \frac{\Pr[property1 \cap property2]}{\Pr[property2]}$$

8.3.1 Scanning

Extensive *address scanning* is a commonly used badness indicator since it is often the precursor of an attack that exploits a vulnerability on the sites that answer the scan. Most Network Intrusion Detection Systems (NIDS) therefore detect and report such scanners. Their detectors typically target *external* hosts probing a monitored network. However, for our study, we are instead interested in finding *outbound* scanners. This is considerable harder as the potential probes are embedded within the host's benign activity (whereas when checking inbound scanning a monitor will *only* see the malicious traffic). We find that in our setting, a classic threshold-based scan detector is easily mislead by, e.g., users browsing the Web. Detectors that count *failed* connection attempts work better but they still misfire, e.g., for P2P and related traffic classes.

We therefore use an approach which is loosely borrowed from TRW [58]: use the *ratio* of successful vs. unsuccessful TCP connections initiated by the originator. We define an unsuccessful TCP connection as one where the client sends a SYN packet but either receives a TCP reset or no answer at all. For our data set we find a bi-modal distribution of success ratios for all *pairs* of DSL lines and remote IPs, see Figure 8.1. It shows that the ratio between a line's number of successful outgoing connections to a particular destination and its total number is generally

either $\ll 0.2$ or $\gg 0.8$. For a given line we thus define a destination as either *successful* or *unsuccessful*, depending on the category of the pair.[4]

One unsuccessful approach for finding scanning DSL lines is to determine their ratio of successful destinations. However, there is no clear cutoff and therefore this metric, see Figure 8.2, does not provide us with clear distinction between malicious and benign activity. Presumably, P2P clients can also have a large numbers of unsuccessful destinations thus rendering this approach impractical. To overcome this problem we use an additional feature—port numbers—for our badness indicator. The motivation is that scanners typically tend to limit their activity to a small subset of ports with known vulnerabilities.

We identify these ports using a twofold approach. We *(i)* investigate the ports with the highest number of unsuccessful connections in our ISP data set and select the ports associated with perilous or vulnerable services (e. g., Windows RPC, data bases, remote desktop protocols). We *(ii)* augment this list by port numbers observed in *incoming* scans that are flagged by the production IDS at LBNL, again manually confirming that the services are perilous or vulnerable. Using this approach we identify 14 ports. After prefiltering our data set to include only connections to those 14 ports we see a nice separation. Now, almost all DSL lines either have only a small number (less than 20) or a very large number (more than 1,000) of unsuccessful destinations. A scatter plot for day 9 is shown in Figure 8.3. For this day we actually see either more than 10,000 unsuccessful destinations or less than 100 except for one DSL line with 380 unsuccessful destinations.

For the scanning metric of our study, we use two different thresholds. A DSL line triggers the scanning metric if we see more than *(i)* 1,000 or *(ii)* 100 unsuccessful destinations within our data set, considering only the subset of ports. We moreover confirm that this nice separation can also be observed for older data sets collected at the same location and at LBNL and AirJaldi. Using this metric we flag a line as scanner on a particular day if it exceeds our thresholds. For the example day that we plot in Figure 8.3 we find that 0.057 % (threshold 1,000) and 0.062 % (threshold 100) of lines are scanners.

Another dimension of scanning is *port scanning*, where malicious hosts may probe many ports on a single remote IP. Since in general no benign application needs to contact many ports on a single IP we can use a threshold based approach to classify a line as a port scanner. A DSL line that contacts at least two remote hosts on more than 50 ports *unsuccessfully* within a single day is considered a port scanner for that day.

[4] In the following, we ignore any pairs that do not fall into either category; these are less than 2 %.

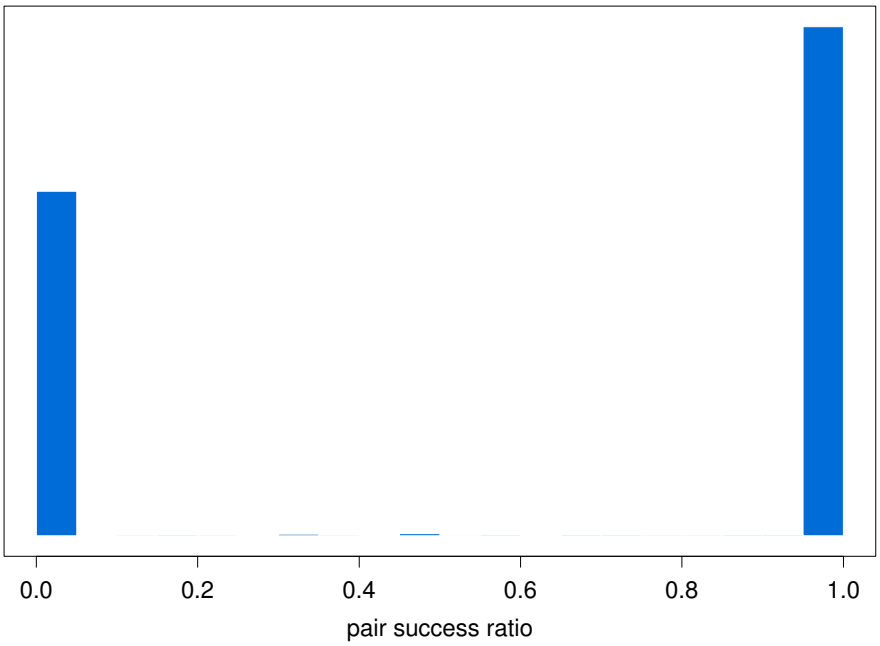

Figure 8.1: **Histogram of connection success ratio per ⟨DSL line, remote IP⟩ pair for day 9 of the ISP trace.**

We choose these values based on our experience: we pick 50 distinct ports, since FTP or P2P clients might contact some ports unsuccessfully, but not as many as 50. We also require at least two destinations to be (unsuccessfully) contacted in order to reduce false positive. Otherwise, a user who manually checks a destination, e.g., for debugging purposes, would already lead to a hit. Roughly 0.1 % of DSL lines are port scanners according to this definition across the 14 day ISP trace. If we flag a line as port scanner if it scanned just one remote host per day, this number increases to 0.5 %. However, most of them scanned only a single IP during the whole 14 day period. Even with our metric which includes two hosts we find that most port scanner still only scan on a single day.

8.3.2 Spamming

Another way for assessing badness is whether a DSL line sends spam. To find an appropriate metric for classifying a line as spammer we now explore the SMTP traffic in detail. Overall, roughly 20 % of DSL lines use SMTP. Since spammers might be more likely to have more unsuccessful commands we analyze the return codes of SMTP commands relying on anonymized SMTP headers. However, we

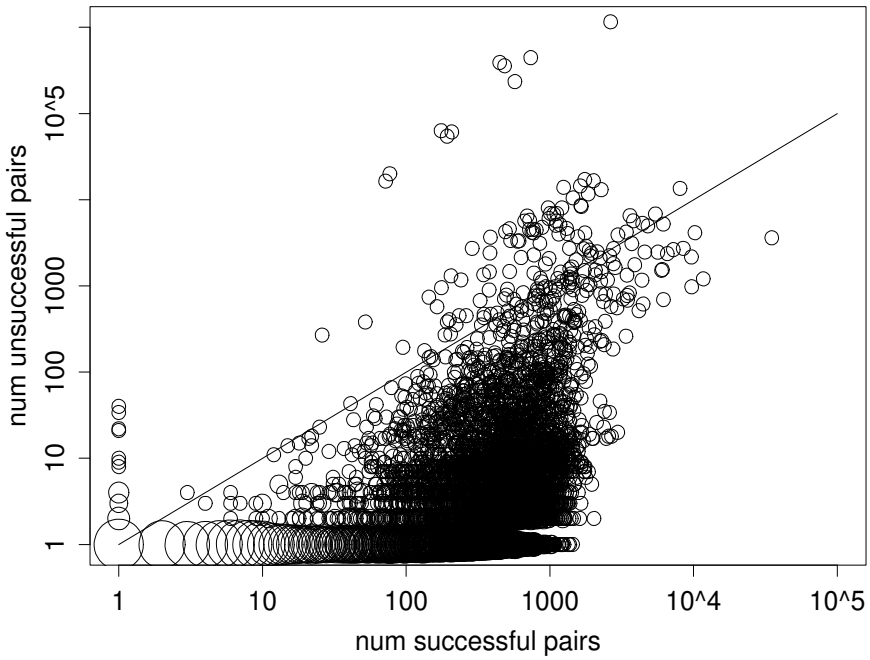

Figure 8.2: **Scatter plot of number of successful and unsuccessful ⟨DSL line, remote IP⟩ pairs per DSL line for all ports for day 9 of the ISP trace. The size of the circles reflect the number of lines having this specific combination.**

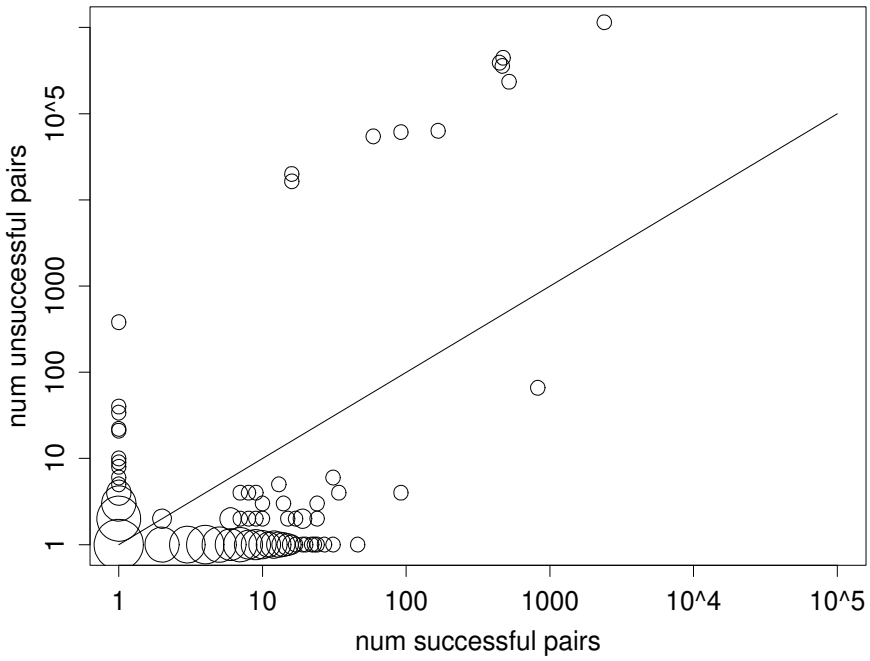

Figure 8.3: **Scatter plot of number of successful and unsuccessful ⟨DSL line, remote IP⟩ pairs per DSL line for suspicious ports only for day 9 of the ISP trace. The size of the circles reflect the number of lines having this specific combination.**

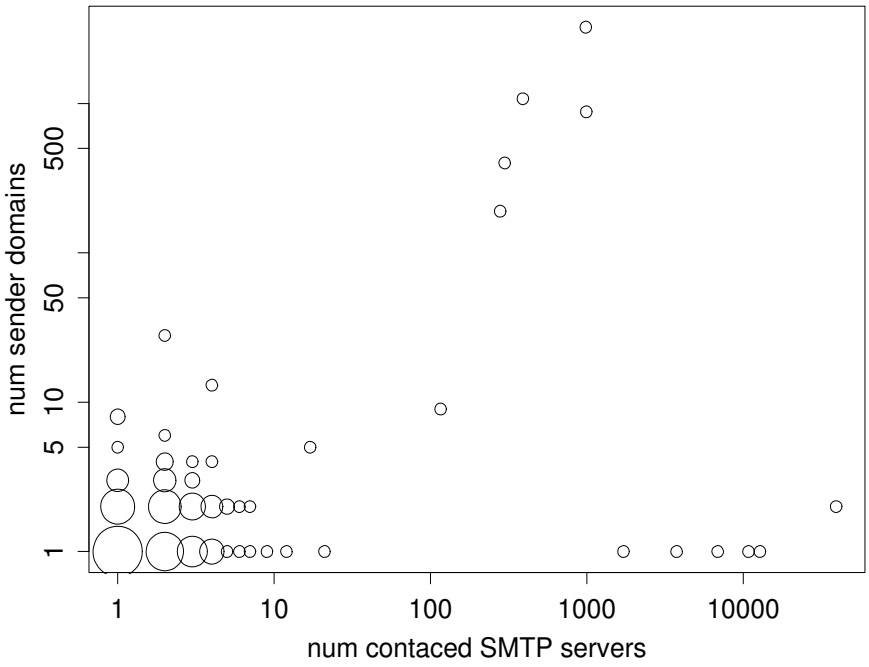

Figure 8.4: **Spamming: Scatter plot of number of distinct sender domains vs. number of contacted SMTP hosts per DSL line for day 9 of the ISP trace. The size of the circles reflect the number of lines having this specific combination.**

find that this is not the case. The distribution of success ratios and counts has a wide spread—we do not observe a bi-modal distribution that could be used to derive a spamming metric. Among the reasons are e. g., that we find that some lines seem to have misconfigured clients that send the same command over and over without adhering to the server's reply codes.

Another possible metric is the number of distinct SMTP servers a line tries to contact or the number of distinct (anonymized) *sender* domains a DSL line uses. The number of distinct SMTP servers can be an indicator since we do not expect DSL customers to run their own SMTP servers and thus they should rely on a (small) number of e-mail providers to deliver their mails. This is necessary since the IP range of the monitored DSL lines is known to be dynamic and many SMTP servers reject mails from any dynamic addresses without authentication. Using the same rationale, we do not expect a large number of distinct sender domains per DSL line. Figure 8.4 shows the number of distinct sender domains vs. the number of SMTP hosts contacted per DSL line. We find that DSL lines consistently either contact less than 25 SMTP servers or more than 100. Indeed,

most lines contact less than 10. The number of distinct sender domains is spread more evenly. Nevertheless, the majority of lines has less than 10 distinct sender domains. We thus classify a line as *spammer*, if it contacts more than 100 distinct SMTP servers. We again evaluate the number of distinct SMTP servers for each day separately and flag a line as spammer if it exceeds the threshold for that day. For the example day in Figure 8.4 we find that 0.07 % of lines are spammers. Older data sets collected at the same location as well as the AirJaldi and LBNL data sets show similar characteristics.

One might argue, that it is also possible to check if an IP address is blacklisted by an anti-spam providers, e. g., Spamhaus. However, this is difficult within our setting. Address assignments are dynamic and we experience considerable address reassignments, see Section 4.2. It is thus very likely that a single spam host causes many IPs to be blacklisted. Moreover, many anti-spam providers blacklist *all dynamically assigned* IP ranges.

8.3.3 Known Malware Families

To better estimate malicious activity we next examine badness indicators related to known malware families. In particular, we focus on Zlob (aka DNS.Changer), Conficker, and Zeus.

One property of the Zlob malware family [126] is that it changes the DNS resolver settings of infected hosts. It even tries to change the resolver settings of the DSL routers. Zlob changes the DNS resolver to a known set of malicious IPs. The motivation is that these malicious DNS resolvers can then return wrong answers and thus redirect some queries to malicious servers. This presumably facilitates phishing. We thus classify a DSL line as being infected with Zlob if it uses one of those DNS resolvers. We note that Zlob targets both Windows as well as Mac systems (social engineering by fake codec download). Thus Zlob is a prime candidate for verifying the potential infections of Mac systems.

Another malware family that has attracted a lot of attention is Conficker [90]. Even though Conficker started already in the fall of 2008 there are still many Conficker infected hosts in the wild. For April 2010 the Conficker Working Group [29] reports 5–6 M unique Conficker A+B infected IPs and 100-200 k unique Conficker C infected IPs. Note, that the number of IPs is not necessarily a good estimation of the population size. The Conficker Working Group estimates that population size is between 25 % and 75 % of the number of unique IPs per day. Given this prevalence of Conficker we search for signs of Conficker activity. To find rendezvous points Conficker is known to generate a list of 250 different domain names per day

(Conficker A and B, Conficker C uses 50,000 domains) and then tries to resolve them in DNS. Since the algorithm for generating the domain names is known [118] we compute the relevant domain names for our observation period and check which DSL lines try to resolve these domains. To account for potential clock skew of the client machines we also include the domain names for the days before and after our observation period.

However, Conficker C is known to generate domains that are legitimately registered (due to short domain names and the large number of generated domains). To rule out false positives we thus only flag a DSL line as infected with Conficker if it did at least 50 lookups for known Conficker domains. Typically an infected line looks up at least 250 domains per day. Indeed, we find that DSL lines either look up less than 10 Conficker domains or $\gg 50$, with most lines resolving exactly 250 domains. We note that we find considerably more Conficker A and B infections than Conficker C infections. This agrees with the number of IPs reported by the Conficker Working Group [29]. We also note that Conficker uses address scanning to find other vulnerable machines and spread itself. Thus, we expect to find a correlation between address scanning and Conficker domain lookups.

The final malware family we consider is Zeus. We use the Zeus Domainblocklist [134] to identify DSL lines infected with Zeus, by checking whether DSL lines try to resolve any domains on the list via DNS. However, since the list not only contains domain names that are random or appear random but also domain names that indicate scareware (e. g., `updateinfo22.com`), or type squatting (e. g., `google-analytiics.cn`) we require more than three lookups per line and day in order to classify a line as infected. If it had fewer lookups it is likely due to embedded ads or some other artifact but not due to a bot. Since we do not have a continuous feed of the Zeus Domainblocklist, we use a one day snapshot from Mar-23 for the ISP traces as well as for AirJaldi1 and AirJaldi2. We use another snapshot from May-06 for LBNL and AirJaldi3.

8.3.4 Emerging Threats Rule Set

Another way of detecting malicious activity is using a Network Intrusion Detection System (NIDS). However, it is non-trivial to find an appropriate rule set that does not generate too many false positives while also not missing much relevant behavior either. Since we are trying to get an overview of malicious activity, we attempt to utilize the popular Emerging Threats (ET) [39] rule sets for the NIDS Snort [97]. We exclude all rules that are labeled as aiming at finding inappropriate (e. g., gaming traffic or P2P) as we are not interested in such.

Analyzing one day of ISP traffic, we however get more than 1 million hits, flagging more than 90 % of the monitored DSL lines—clearly this has to include many false positives. By restricting the analysis to check only outbound activity, the fraction of flagged lines is reduced to 61 %; still too many to be a reliable indicator for malicious activity. The main problems with the Emerging Threats rule sets are their sheer size (more than 3,500 rules in total) and a general lack of documentation (many rules are not documented at all). In particular, rules do not include any indication on how specific they are (i. e., what the likelihood of false positives is), nor how severe the activity is that they target (e. g., is this rule triggered by a C&C channel or by adware). The rules that trigger most often for outgoing traffic include an adware user-agent, weird HTTP URI encodings, and duplicate user-agents. When inspecting the results, we even find a rule to flag the Alexa [6] toolbar. The large number of rules and the lack of documentation also make it impossible to manually select a high-quality subset of rules. We tried to whitelist rules that are likely non-malicious according to our objectives (e. g., Alexa). However, such whitelisting did not significantly reduce the number of DSL lines with alerts reported since the remaining rules were still not sufficiently crisp. For example, some of the often reported rules identify bots contacting their master by checking URLs for specific patterns. However, many of those patterns are rather generic and rely on the presence of single- or double-letter URL query parameters (e. g., `b=`, `tm=`). These also commonly appear in benign traffic.

We conclude that while the Emerging Threats rule sets might be an important tool for securing small networks with strict acceptable-use policies, such as small businesses, they are less appropiate for large networks and environments with more liberal security policies. These results are consistent with the experiences of Carlinet et al. [20] who run Snort on ADSL traffic from about 5,000 customers of a French ISP.

8.3.5 Security Awareness and Risky Behavior

To analyze whether DSL customers are aware of potential hazards and take suggested countermeasures we analyze *(i)* whether DSL lines use anti-virus scanners, *(ii)* if they regularly update their OS software (e. g., Windowsupdate), and *(iii)* if they download Google's Safe Browsing blacklist. For this we analyze the HTTP user-agent strings and HTTP server hosts (from HTTP's Host header). Most anti-virus and OS software updates are done using HTTP and they use specific user-agents and/or HTTP servers. Searching for those allows us to classify a DSL line as software updater and/or anti-virus user. Likewise, the HTTP servers serving Google's blacklist are well-known and we can thus identify those DSL lines

which use the blacklist.

Moreover, we also check if DSL lines actually request any blacklisted URLs. Such behavior clearly has to be considered risky. We do this by utilizing the Google Safe Browsing API blacklists and characterizing HTTP request as either "bad" or "safe". We update our blacklist copy every 25 minutes and store its history, i. e., we track changes in the blacklist over time. This allows us to verify whether a requested URL was blacklisted at the time of the request or only after the request took place.

8.4 Security Awareness and Risky Behavior

Next, we investigate the security awareness of the users of our networks and how prevalent risky behavior is. We start by studying the characteristics of the DSL customers from the European ISP, since it is the richest dataset with regards to meta information. We then compare the results from the European ISP with those from the AirJaldi community network and the LBNL setting where applicable.

8.4.1 Security Awareness

Figure 8.5 shows the fraction of active DSL lines as well as lines with anti-virus respectively OS software updates. On any given day 67–72 % of DSL lines perform OS software updates and 64–69 % of lines update their anti-virus signatures (or anti-virus engines). From the cumulated data we see that over the 14 day observation period up to 87–88 % of lines check for updates to their OS and anti-virus software. This highlights that the user-base is in principle aware of hazards and performs recommended precautions.

When focusing on DSL lines with activity from only Macs (and not Windows systems) we find that only up to 54 % do anti-virus updates and that up to 81 % perform software updates within our observation period.

8.4.2 Google Blacklist

But what about risky behavior such as requesting potential dangerous URLs? For this purpose we check all requested URLs against the Google Safe Browsing blacklist using Google's Safe Browsing API. Overall, we find that only about 0.03 % of all HTTP requests are blacklisted. However, if we investigate the per line behavior, we find that on any given day up to 4.4 % of the DSL lines request at

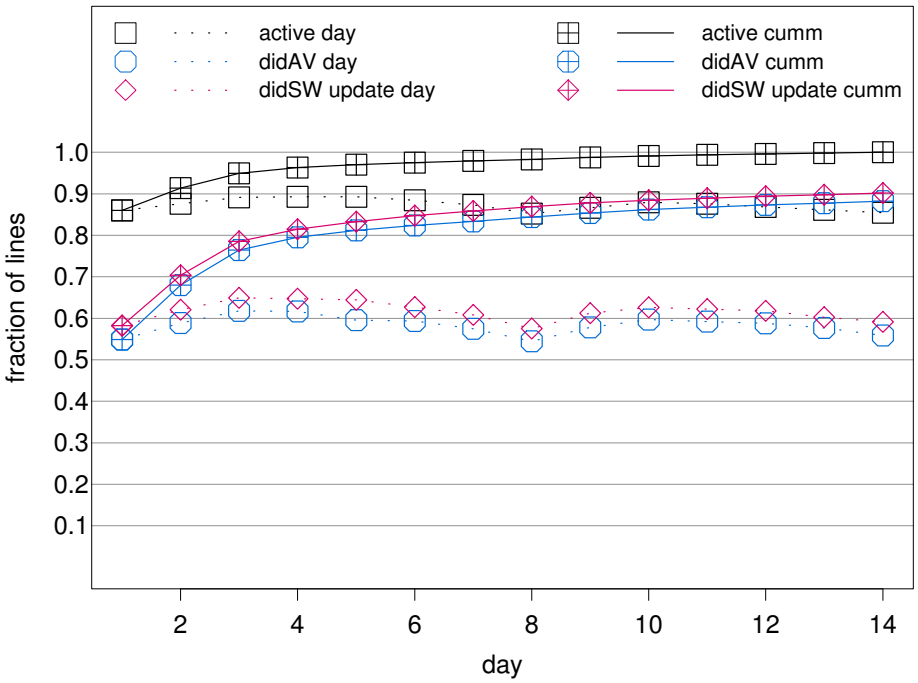

Figure 8.5: **Fraction of active DSL lines, lines with anti-virus updates, and lines with OS software updates for daily and cumulated data sets from the ISP.**

least one blacklisted URL. Across the whole 14 day period we see that a striking 19 % of lines request at least one blacklisted URL.

To check if the browser could have warned the user we next check if the user-agent placing the request actually downloaded the blacklist from Google. Our results are mixed. For some lines the hits are from user-agents that did not download the blacklist. However, many lines have user-agents that downloaded the list and still request a blacklisted URL. While some of these hits occur before the URL appeared on the blacklist[5], many requests occur even though the URL is on the blacklist. I. e., the user placing the request (presumably by clicking on a link) *should have been warned* by the browser that the URL may be malicious. This behavior is in contrast to the observed diligence regarding anti-virus and OS software updates.

To better understand this phenomen we study if the DSL lines that request URLs before they were blacklisted correlate with the DSL lines that request URLs that

[5] We allow a 1 hr grace period, i. e., we consider a URL blacklisted 1 hr after it appeared in our copy of the list.

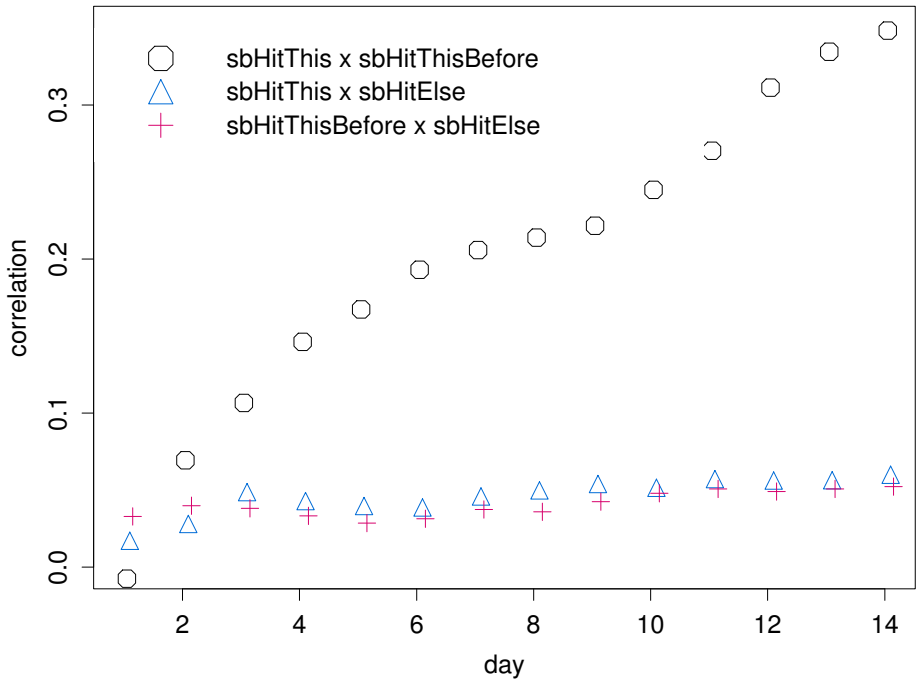

Figure 8.6: **Correlation between DSL lines that request blacklisted URLs before they were blacklisted, while they are blacklisted and lines for which the user-agent placing the request never downloaded the blacklist. Cumulated data set for the European ISP.**

are on the blacklist at the time of the request. Figure 8.6 shows the correlation of DSL lines requesting blacklisted URLs for which *(i)* the browser did not download the blacklist (sbHitElse), *(ii)* the browser downloaded the blacklist and the hits occurred before the URL was blacklisted (sbHitThisBefore), and *(iii)* the browser downloaded the blacklist and the hits occurred while the requested URLs are already blacklisted (sbHitThis). No correlation or weak correlation indicates that the users who request URLs before they were blacklisted do not request URLs already on the blacklist (i. e., the request only took place because the user was unaware of the potential malignity of the URL).

While correlation is small when we focus on single days there is significant correlation, 0.35, for the whole 14 day period between sbHitThisBefore and sbHitThis. Thus it seems that there are some DSL lines that regularly and consistently request blacklisted URLs. We note that Google suggests that browser developers implement a 25–30 min update interval for the blacklist. Furthermore, Google

states that a warning can only be displayed to the user if the blacklist is less than 45 minutes old, thus forcing browsers to regularly update the list.

Interestingly, the more active lines are also more suspectible to such behavior. We find that the fraction of lines requesting blacklisted URLs significantly increases, up to 55 % overall for DSL lines with high activity and 34 % for medium activity. There might be two reasons for this increase, *(i)* more activity increases the chances of finding a blacklisted URL or *(ii)* more active users might ignore the warnings.

As with blacklists, we find that there is also a correlation between DSL lines that update their anti-virus scanners (and OS software) and activity level. This might indicate that the more active users are more security aware but also tend to ignore warnings about potentially malicious URLs, maybe because they assume that their anti-virus software will protect them.

8.4.3 Comparison with AirJaldi and LBNL

Comparing security awareness at the European ISP with the AirJaldi network in India, we find them quite similar. In terms of fraction of HTTP requests we find that at AirJaldi approximately 0.02 % of them are blacklisted by Google (ISP: 0.03 %) while 0.5–1.12 % of HTTP requests are for updating anti-virus software at AirJaldi (ISP: 1 %). For OS software updates, the numbers differ: up to 2.8 % of HTTP requests are for OS software updates while only 0.3 % are so at the ISP. Assessing security awareness on a per host/line basis is difficult, however, given the layered NAT structure. We find that 3.2–4 % of the observed IPs at AirJaldi do anti-virus and OS software updates and that 3.8–5.4 % request blacklisted URLs. We note, however, that each IP address can connect anywhere between a handful and several hundred users.

We next turn to security awareness at LBNL. We find relatively few hosts that update anti-virus (24 %) and OS software (31 %). This can be explained by the fact that LBNL uses centralized, internal update servers. Furthermore, the operating system distribution also differs, with significantly more Linux and Mac hosts. These lower numbers are also reflected in a lower fraction of HTTP requests related to such updates: 0.5 % for anti-virus and 0.2 % for software updates. When we turn to risky behavior, we find that only 0.01 % of HTTP requests are blacklisted. In term of hosts we also find smaller number than at the ISP: up to 0.92 % of hosts per day and less 1.25 % overall. We also note that the relative increase between days is less at LBNL than at the ISP. However, we still observe the influence of activity levels, 6.1 % of hosts with high (2.4 % with medium) activity request blacklisted URL. Furthermore, the correlation between hosts that request URLs before they

are blacklisted and hosts that request URLs that are already blacklisted is 0.136, which is similar to the correlation we observe at the ISP (0.146) when considering only the first 4 days.

8.5 Malicious Activity

After finding that even though users appear to be security aware and take appropriate precautions they still engage in risky behavior, we now use our metrics for locating unnatural and malicious behavior. We emphasize that the overall estimate of malicious activity that we derive can only be a lower bound of the total. Clearly, there must be malware that our metrics are missing, and we cannot estimate how large that share is.

Moreover, we also study the influence of security awareness, risky behavior, activity level, usage of NAT, and operating systems on infection probability. We start by studying the characteristics of the DSL customers of the European ISP, since it is the richest dataset with regards to meta information. We then compare the results from the European ISP with those from the AirJaldi community network and the LBNL setting where applicable.

Figure 8.7 shows the likelihood that a line triggers any of the malicious activity metrics for the European ISP and is thus considered to be bad, labeled as `isbad`. We find that both on a per day basis as well as overall there is only a small fraction of actively malicious host, < 0.7 and < 1.3, respectively. Moreover, the percentage does not vary by much across days.

However, even though the cumulative probabilities are still small they are increasing over our 14 day period. This indicates that over time we are able to identify more lines with malicious activity and may imply that longer observation periods may reveal even more infected lines. There are two regions of growth, a short-lived, more rapid one up to 3 days, and then a slow but steady one beyond that. This may be due to *(i)* malware somewhat infrequently engaging in activities, *(ii)* malware somewhat infrequently engaging in activities that we are able to detect, or *(iii)* a short infection lifetime, i. e., a host is only infected for a short period of time. We speculate that the increase is due to a combination of all possible explanations. Moreover, we next show that the observability of infections is limited; e. g., an infected line may be used as a spammer for a subset of the time when it is active.

We find that the malware families are contributing the most to the overall badness estimation while spammers, scanners, and port scanners are less prominent on a

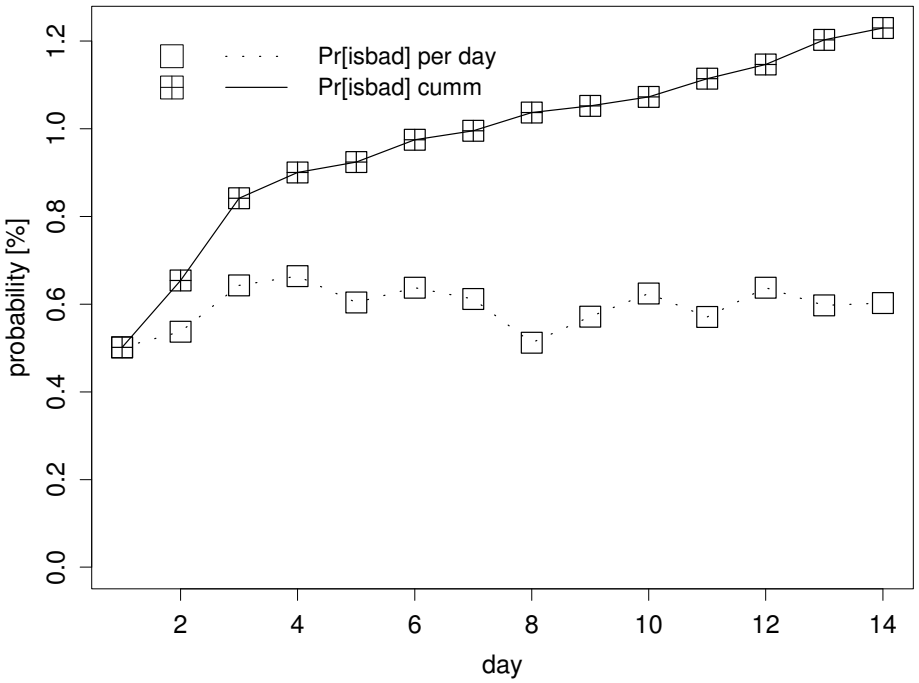

Figure 8.7: **Probability that a line triggers any of the malicious activity metrics (isbad) for the European ISP for daily and cumulated data sets.**

per day basis. See Table 8.1 for an overview. However, spammers are contributing significantly to the total observed badness. The reason is that more than 44% of the spammers are only active on a single day, i. e., they trigger the metric only on a single day. In contrast, only 6/11% of the scanning activity is limited to a single day. On average (mean and median) scanners are seen for 4 days. We note that both of our scanning thresholds (100 and 1,000) yield similar results. We thus focus on a threshold of 100 for the remainder of this paper. For most metrics we observe a difference between the mean number of days that the metric is triggered and the median number of days. This indicates that there is no consistent behavior by the malicious lines. Indeed, an in-depth analysis reveals that some spammers/scanners start their malicious behavior as soon as the line is active. Others stop in between or are only active for a short period. The fact that the malware families are usually (mean) triggered on 4 days confirms that the bots engage in activity on a regular basis. However, malicious activity such as port scanning or spamming seems to be limited to subperiods. There are additional causes for such sporadic activity: first the malware might be removed, the bot may crash due to pure software quality

Table 8.1: **Probability that a DSL line triggers a malicious activity metric on a per metric basis. The daily numbers summarize the range of the probability values per day and the cumulative numbers summarize the malicious activity across the full trace duration. To estimate the persistence of the malicious activity we also include the mean/median number of days that each metric triggered as well as the percentage of lines where the metric triggered only on a single day.**

Metric	Probability		Activity prevalence in days		Probability of single day activity
	daily prob.	cumm. prob.	mean	median	
Spam	0.03–0.10 %	0.25 %	3.6	2	44 %
Scan 100	0.01–0.06 %	0.09 %	4.3	4	11 %
Scan 1,000	0.00–0.06 %	0.08 %	4.1	4	6 %
Port Scan	0.01–0.03 %	0.06 %	3.5	2	39 %
Zlob	0.13–0.19 %	0.24 %	8.4	10	10 %
Conficker	0.17–0.26 %	0.23 %	6.5	6	27 %
Zeus	0.07–0.15 %	0.28 %	4.9	2	38 %
Total	0.50–0.66 %	1.23 %	5.9	4	28 %

and only be reactivated after a host reboots, or a host might be newly infected.

Surprisingly, we find that only a small overlap among the lines that trigger any of the metrics. Most lines with malicious activity trigger only a single metric (92 %) and around 7 % trigger two. Similar observations hold for other traces from this environment.

Nevertheless, we study how the different metrics correlate with each other, see Figure 8.8. We find that spamming and Zeus as well as scanning and Conficker show some correlation (over the full trace duration):

Metric A	Metric B	Corr.	Pr[A|B]	Pr[B|A]
Spam	Zeus	0.109	11 %	12 %
Scan	Conficker	0.227	10 %	50 %

Some of the other metrics[6] only show weak correlations between 0.011 and 0.037 (with conditional probabilities between 1.2 % and 7.7 %). All other combinations show no correlation (<0.002).

[6] ⟨spam,scan⟩ ⟨spam,Conficker⟩ ⟨spam,port scan⟩ ⟨Zlob,Conficker⟩ ⟨Zlob,port scan⟩ ⟨Zlob,Zeus⟩ ⟨Conficker,Zeus⟩

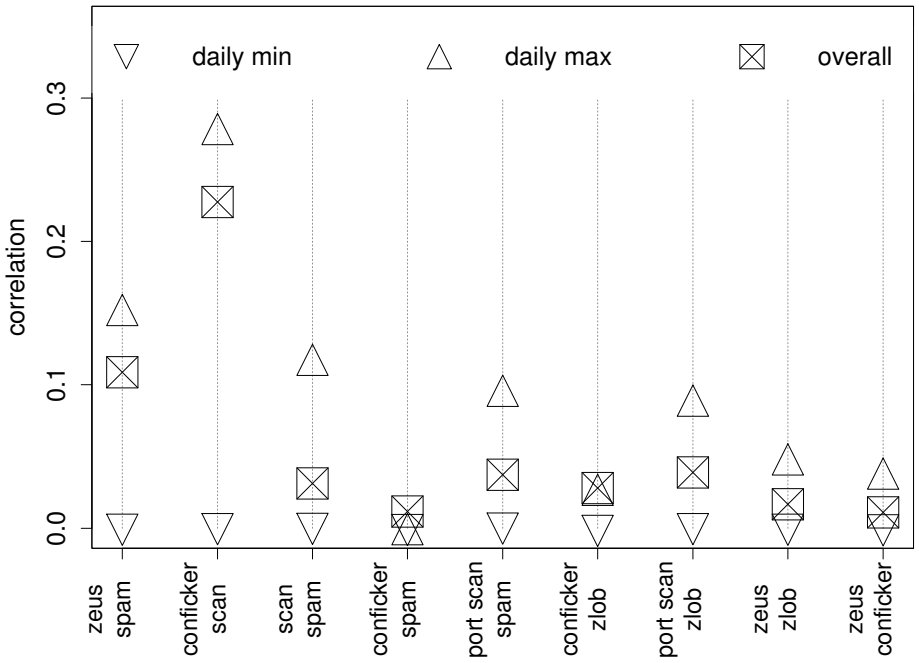

Figure 8.8: **Correlation between metrics for malicious activity for the ISP trace. We plot the min and max values from the daily data sets as well as the correlation over the full 14 day period.**

8.5.1 Influences on Malicious Activity

Next we check whether the likelihood that a line "isbad" is influenced by other parameters.

Influence of Anti-Virus and OS software Updates

We start our investigation with the influence of anti-virus and software updates. Surprisingly, we do not see strong influences, i. e., anti-virus software does not seem to reduce the likelihood of a line being infected with malware. When considering all badness metrics, we find that the probability of infection actual *increases slightly* when using anti-virus scanners (1.26 %, up from 1.23 %). Likewise there is also hardly any influence of OS software updates on the infection probability. Although, the probability of infection slightly decreases when OS software updates are done. We verified that these findings are not biased by NATed DSL lines with multiple

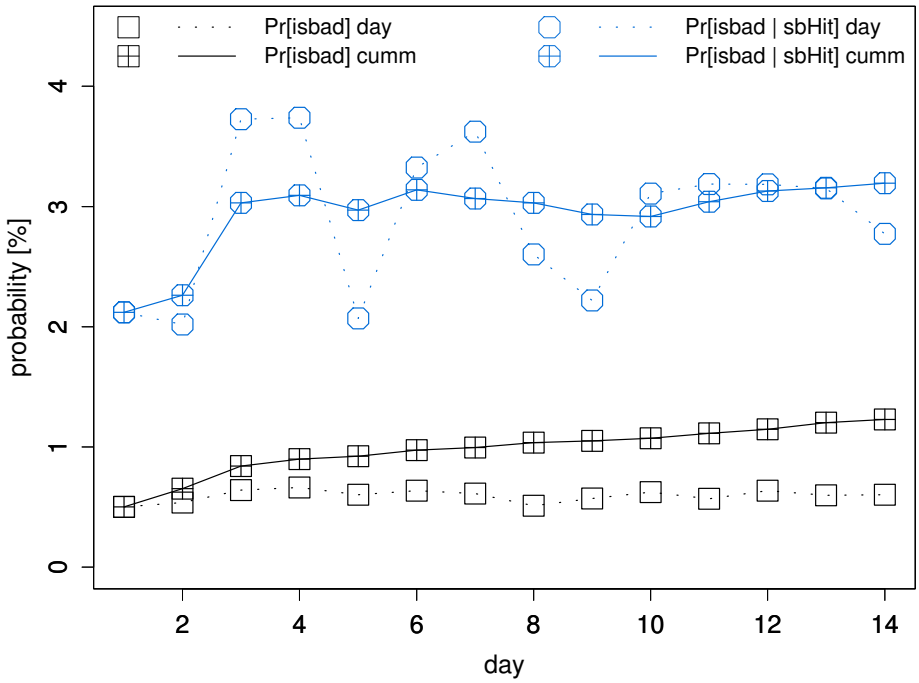

Figure 8.9: **Probability that a line triggers any malicious activity metric ("isbad") given it requested a blacklisted URL ("sbHit") for the ISP trace.**

hosts. I. e., even for lines that do not have multiple hosts the likelihood of infections is not significantly biased by the presence of anti-virus or OS software updates.

Influence of Blacklist Hits

Given the large fraction of DSL lines which request blacklisted URLs we next study if this behavior increases the risk of infection (see Figure 8.9). We see that the probability of a line triggering any of our metrics, "isbad", rises to 3.19 % (up from 1.23 %) if it requests a URL on Google's blacklist. While this is more than twice the overall probability it is still fairly small. We therefore conclude that while the, likely unsafe, practice of requesting blacklisted URLs is widespread, it does not immediately result in a significantly higher likelihood of infection.

Table 8.2: **Overall conditional probabilities of a DSL line triggering on any of our malicious activity metrics given NAT usage and activity for the ISP.**

Condition	Probability given cond.	Probability *not* given cond
multihost	1.81 %	0.73 %
unNATed	1.30 %	1.22 %
high activity	4.08 %	0.92 %
medium activity	1.94 %	0.58 %
low activity	0.46 %	1.32 %

Influence of Activity and NAT

Next, we evaluate whether NAT or different line activity levels influences infection probabilities. Regarding NAT usage we find that lines with multiple hosts are slightly more likely to be infected (1.81 %), whereas unNATed lines are just as likely to be infected than all lines. Activity levels of DSL lines on the other hand does have an influence. Just over 4 % of lines with high activity are infected while only 1.9 % of lines with medium activity are infected. For lines with low activity the infection probability drops to 0.46 %. Table 8.2 summarizes the influence of activity and NAT on infection probabilities.

Malicious Activity on Macs

Next, we turn to DSL lines with activity from only Macs (2.7 % of DSL lines) and evaluate their likelihood of infection. When analyzing the full 14 day observation period the likelihood of infection for Macs is less than for all lines, only 0.54 %. When we investigate the different badness metrics individually, the picture changes. For Macs only one metric triggers: Zlob, a malware that targets Mac and Windows systems. We observe that 0.54 % of Macs are infected and that we observe the infected Macs on every day. In comparison, 0.24 % of all lines are infected with Zlob. We note that we also find a single line that triggered on the Zeus metric on a single day by requesting the same domain 7 times, more than our threshold of 3. Since Zeus does not target Macs, this is a false positive and we exclude it from our analysis. A likely explanation is that the lookups are incidental and are not caused by a Zeus bot.

While these numbers do not suggest that Macs are worse off than Windows, it does indicate that they are also not better off, in particular, given that there are

	AirJaldi 1	AirJaldi 2	AirJaldi 3
IP 1	**ZeuS** Hi AV SW	**ZeuS** Med AV SW	**ZeuS** Hi AV SW
IP 2	**Conficker(3)** Med SW	**Conficker(1)** Med SW	**Spam** Med BLK AV SW
IP 3	Med BLK AV SW	Med AV SW	**Scan** Hi BLK AV SW
IP 4	X	X	**Spam** AV SW
IP 5	X	X	**Spam** Med BLK SW
IP 6	X	X	**Spam** Hi BLK AV
IP 7	Med BLK AV SW	**Scan** Hi BLK AV SW	Hi BLK AV SW
IP 8	Hi BLK SW	**Spam** Hi BLK AV SW	Hi BLK AV SW
IP 9	**Conficker(1)** Hi AV SW	Med AV SW	Med AV SW
IP 10	**Spam? Scan**	X	X
IP 11	**Scan** Med BLK AV SW	AV	AV

Hi / Med = High / Medium Activity AV = anti-virus SW = software update BLK = Blacklist hit
Shaded background = malicious activity

Figure 8.10: **Summary of malicious activity and annotations for all AirJaldi traces. Only these IP addresses had any malicious activity across all traces.**

malware families that specifically also target Macs. Given the rising market share of Macs this is not to be underestimated.

8.5.2 Comparison with AirJaldi and LBNL

Again we perform the same analysis both for the community network in rural India as well as the enterprise network LBNL.

AirJaldi

Within the Indian AirJaldi network we observe very limited malicious activity. However, given that AirJaldi uses a multi-tiered NAT hierarchy we can only explore malicious activity by IP address rather than per DSL line or per end-system.

Within each trace we observe between 180 to 260 active internal IP addresses which are shared among several 1,000 hosts. Across all three traces only a total of 11 IP addresses triggered any of our metrics and thus show signs of malicious activity. Figure 8.10 shows an overview of which metric triggered for which IP address for all three traces. We list all 11 IPs that triggered any of our metrics. An X indicates that this IP was not observed in a particular trace. We also annotate each IP with information about its activity level (Hi, Med), whether it requested blacklisted URLs (BLK), and whether it updated anti-virus (AV) or OS software (SW).

For scanning we use the a threshold of 100 unsuccessful destinations. Using a threshold of 1,000 eliminates IP 10 (in AirJaldi1) as possible scanner. A detailed look at the spammers shows that IP 10 contacted 56 remote SMTP servers, less than our cutoff of 100. However, since no other IP contacts more than 10 SMTP servers, we flag this IP as a likely spammer (Spam?) rather than a spammer (Spam).

Only two IP address trigger our malicious activity metrics in more than one trace. The remaining 9 IP addresses only trigger in one of the traces. We never observe more than 6 IPs with malicious activity any single trace and in only one case do we find multiple forms of malicious activity on a single IP. Unfortunately, we do not know if these 11 addresses are always reassigned to the same NAT gateway in each of our traces since AirJaldi uses dynamic IP assignments via DHCP. Indeed, we find that some of the malicious IPs are only active in a single trace while other IPs have malicious activity across all traces.

Even though we cannot reliably determine the number of infected hosts behind each of the IPs we can estimate. For Conficker this is possible since each infected host will in general generate 250 DNS lookups per day. Thus we estimate the number of infected hosts by dividing the total number of Conficker lookups by 250 and list the result in parentheses (Conficker(n)) in Figure 8.10. Exploring the number of destinations for scanners and spammers for each flagged IP we find that they are well within the range of what we observe at the European ISP. Therefore, we conclude that each of the reported scanning and spamming IPs is likely due to a small number of infected hosts. Indeed, most likely due to a single one.

Given the inability to identify end hosts we also cannot correlate activity levels, anti-virus, OS software updates, or Google blacklist hits with malicious activity. We do however, see that the 11 IPs flagged as malicious have in general high or medium activity level and that they often have blacklist hits and do perform OS software as well as anti-virus updates.

LBNL

We observe no malicious activity at LBNL. Even though our spamming and scanning metrics trigger on some LBNL hosts we confirmed that these hits are benign. The "spammers" are all legitimate mail servers and the "scanners" are hosts that perform scanning for penetration testing purposes. Moreover, we do not observe any Zeus or Conficker lookups nor do we observe any host contacting Zlob resolvers.

8.6 Related Work

In our study, we examine several metrics for malicious activity including scanning and spamming which are established indicators of compromised systems. Allman et al. [7] present a history of how scanning has developed over time. Most network intrusion detection systems (NIDS), such as the open-source systems Snort [97] and Bro [86], use simple threshold schemes to find scanners. Bro also provides a more sensitive detector, *Threshold Random Walk* [58], that identifies scanners by tracking a system's sequence of successful and failed connection attempts. However, as we discuss in Section 8.3.1, existing detectors do not work well for finding *outbound* scans, as needed for our study. Furthermore, the presence of P2P further complicates scan detection, because P2P clients behave similar to scanners, i. e., they unsuccessfully try to contact a number of IP addresses.

Spamming is often countered by blocking known offenders via DNS-based blacklists, such as SORBS [109] or Spamhaus [110]. However, due to the high IP address churn we experience (i. e., DSL lines frequently change their IP address, see Section 4.2) such blacklists do not provide us with a reliable metric. Furthermore, many blacklists include the *full* dynamic IP space. Spammers' use of botnets and "dark" IP space [92] further complicates the applicability of blacklists as targets are moving rapidly. Ramachandran et al. [93] identify spammers by observing the destinations they try to connect to. This eliminates the need for content inspection. The Snare system [51] extends this approach by building a reputation based engine relying on additional non-payload features. These approaches, however, deploy clustering algorithms and thus rely on suitable training sets which is not available for our data set. We, however, can inspect anonymized SMTP headers and such infer some semantics directly.

We also check our data sets for indicators of specific malware, all of which are analyzed and tracked in detail by other efforts: *Conficker* [29, 90]; Zlob [126]; and Zeus [134]. For example, Bailey et al. [13] survey botnet technology and

Dagon et al. [33] examine malware that changes a client's DNS resolver, including the Zlob trojan.

Another approach for finding malicious activity is to leverage a NIDS' output directly. Therefore, we also attempt to use general signature libraries for identifying malicious activity, e. g., the Emerging Threats library [39] for Snort. However, as discussed in Section 8.3.4, these are too inaccurate for deriving reliable metrics (which confirms the repeated experience of network operators while deploying signature-based NIDS). Carlinet et al. [20] also run Snort on ADSL traffic from about 5,000 customers of a French ISP to study what contributes to a user's risk of being infected with malware. For their study, Carlinet et al. removed 20 Snort signatures triggering the most alarms. However, they do not further analyze how the remaining ones contribute to the overall results, or whether there is relevant overlap between them. This confirms our experience, that large signature sets tend to flag too much benign activity and thus do not result in reliable metrics. We also leverage Google's Safe Browsing blacklist [50]; the approach used for collecting the blacklist is originally described by Provos et al. in [91]. More closer to our approach is work by Stone-Gross et al. [112], which combines several malware-specific metrics for detecting rogue networks. They however focus on finding malicious ISPs rather than end-user systems.

There has been some work on the prevalence of individual malware. The Conficker Working Group states that 3 million infected hosts is a "conservative minimum estimate", and it cites the study of an anti-virus vendor that finds that 6% of the monitored systems are infected. Weaver [123] estimates the hourly Conficker C population size in March/April of 2009 to average around 400-700 k infections. At the time of writing, the Zeus tracker reports 658 active Zeus C&C servers.

8.7 Discussion and Future Work

Given that conventional wisdom says that residential users or users of community networks are responsible for much of today's Internet insecurity we, for this study, select a number of indicators that are commonly used for detecting malicious activity. Some of them are behavioral and thus able to find any malware that exhibits such behavior (scanning, spamming), while others target specifics of a particular malware family and thus only allow us to understand infections from those (Conficker, Zlob, and Zeus).

While we are confident that our metrics reliably flag what they are designed to find, we emphasize that the overall estimate of malicious activity that we derive in our study can only be a lower bound of the total amount present. Clearly,

there must be malware that our metrics are missing, and we cannot estimate how large that share is. Indeed, we observe an indicator that it may be non-negligible: the victim sets detected by our metrics are mostly disjoint, which suggests that systems tend to get infected by a single malware, not by many simultaneously thus we underestimate the amount of total malicious activity.

However, our approach for finding malicious activity mimics how security-conscious sites typically monitor their network for security violations: by building a toolbox of independent detectors—such as attack signatures, blacklists, behavioral detectors for activity commonly associated with compromised hosts—they aim at increasing their coverage of malicious activity, without however being able to tell what they are missing.

When building such a toolbox, an important trade-off is the number of false alarms generated by each detector. Just as every site needs to find the right mix, we had to select a set of metrics which *(i)* find a significant subset of malware; *(ii)* have low false-positive rates; and *(iii)* can be used across different environments. While initially we started with a larger set of metrics, we found some of them to not fit well with our needs. For example, the semantics of many blacklists are not very crisp and often lead to many more hits than plausible; likewise, per our discussion in Section 8.3.4, generic NIDS signatures sets are prone to false positives. In contrast, we find the metrics used for this study to meet our criteria well.

There are a number of interesting conclusions about residential users and community networks that we can draw from our study, even when keeping the fundamental limitations in mind:

- A typical residential system is unlikely to be participating in scanning or spamming, and thus also unlikely to be with infected with any malware relying primarily on one of these vectors.
- Residential users are risk-aware: many of them update their systems regularly and deploy anti-virus software. However, doing so does not seem to to have much of an impact on their likelihood of being infected.
- Even though being risk-aware in principle, users regularly exhibit risky behavior: they contact malicious sites even though they must have been warned of doing so by their browsers. Interestingly, however, we find that such behavior only slightly increases their probability of being infected—significantly less than one may presume.

There is a second set of conclusions that our methodology allows us to infer: *relative* comparisons between different network environments. Assuming that our metrics cover a representative subset of malicious activity, we find that we see similar levels

of malicious activity at the European ISP and the rural Indian network. Likewise, we also find similar levels of both security-awareness and risky behavior in the two environments. In comparison, at LBNL we do not observe any malicious and significantly less risky behavior.

For future work we plan to extend our set of metrics for malware by adding more signatures for common botnets to extend the coverage of malicious activity we are able to observe. Furthermore, we plan to pursue a per *host* analysis of the AirJaldi environment. Indeed, the AirJaldi network operators are planning to improve the network design to regain visibility of the hosts.

Chapter 9

Conclusion

We study residential broadband Internet traffic using anonymized packet-level traces augmented with DSL session information. Our data covers more than 20,000 customers from a major European ISP. To ensure privacy, all data is immediately anonymized. The goal of this thesis is to understand the nature of residential traffic characteristics and malicious activity in such networks. In addition, we also want to compare our results to other network environments, including university networks, an enterprise network, and a rural community network in India. Unlike for university networks, researchers rarely have large-scale access to residential traffic, and thus its makeup, dynamics, evolution, and variations remains underexamined by other studies. Yet, understanding the nature of residential traffic characteristics is imperative for network operators to design and develop future network configurations and architectures.

Efficient Retrospective Traffic and Security Analysis

We present a network "Time Machine" (TM) that enables quick access to past network traffic and security forensics. We add Time Machine support to the open source Bro Network Intrusion Detection System (NIDS) and examine a number of applications (controlling the Time Machine, correlating NIDS alarms with associated packet data, and retrospective analysis) that such integration enables. Our results from an operational network show that such a combined TM $\leftrightarrow$ NIDS setup greatly enhances the possibilities of security analysts to assess NIDS notifications and security incidents.

Characterizing Residential Broadband Traffic

We analyze DSL sessions and find that sessions are surprisingly short. The median session duration is only 20–30 minutes. As a consequence IP addresses are reassigned frequently: 50 % of IP addresses are assigned at least twice per day. We

also present a novel approach to detect network address translation (NAT) and to estimate the number of hosts connected behind such a NAT gateway. We find that most DSL lines (90 %) indeed use a NAT gateway to connect to the Internet. Furthermore, we show that more than 10 % of lines have multiple hosts that are active at the same time at least once. These findings indicate that great care must be taken when using IP addresses as host identifiers, e. g., for estimating population sizes.

Next, we investigate application layer characteristics. Surprisingly, we find that HTTP and not P2P dominates the traffic mix by volume. More than 50 % of bytes are due to HTTP. We conclude that HTTP's resurgence is due to popular high-volume content hosted via HTTP (e. g., video portals and one-click hosters). We note that a number of these results agree with those of other contemporaneous studies of Internet traffic [41, 69, 98], suggesting that the trends are representative for a significant fraction of the Internet. We also assess performance and path characteristics of TCP connections and find that most DSL lines do not utilize their available bandwidth, mostly due to settings on end-hosts, e. g., TCP receiver windows that are too small to utilize the bandwidth given the observed bandwidth-delay-products. Furthermore, we observe that connections from client-server applications, like HTTP and NNTP, achieve an order of magnitude higher throughput than P2P connections but still fail to fully utilize their available bandwidth.

This highlights, that the characteristics of residential traffic have to be reevaluated constantly, since network traffic and user behavior and demand is constantly changing. Furthermore, the introduction of new devices to access the Internet (e. g., smartphones and tablets) also impacts traffic characteristics and usage patterns. Only a solid understanding of such characteristics enable network operators to plan, provision, and design future network architectures and to address current limitations. For example, consider the application protocol mix. Since the turn of the century it has changed several times, from being dominated by HTTP, to P2P, and back to HTTP with NNTP also appearing as a significant traffic contributor. Yet, P2P applications have different traffic demands (e. g., symmetry of upstream / downstream capacity) than client-server protocols, like HTTP, and thus require different traffic engineering approaches. Or consider the example of TCP options. While TCP window scaling has been introduced in the past to increase performance, we find that the default setting on many hosts fails to fully utilize the available bandwidth, thus limiting performance. Our results can be a guidance for network operators and operating system and device developers to improve network performance and end-user experience.

Malicious Activity and Risky Behavior

We also assess malicious activity and risky behavior of residential users and compare the results to other network environments. We select a number of indicators—signatures of common malware families as well as behavioral models such as scanning and spamming—to detect malicious activity. While we are confident that our metrics reliably flag what they are designed to find, we emphasize that the overall estimate of malicious activity that we derive in our study can only be a lower bound of the total amount present.

However, our approach for finding malicious activity mimics how security-conscious sites typically monitor their network for security violations: by building a toolbox of independent detectors. When building such a toolbox, an important trade-off is the number of false alarms generated by each detector. While initially we started with a larger set of metrics, we found some of them to not fit well with our needs. For example, the semantics of many blacklists are not very crisp and often lead to many more hits than plausible; likewise, generic NIDS signature sets are prone to false positives. In contrast, we find the metrics we use for this study to meet our criteria well.

There are a number of interesting conclusions about residential users and community networks that we can draw from our study, even when keeping the fundamental limitations in mind: Assuming that our metrics cover a representative subset of malicious activity, we find similar levels of malicious activity at the European ISP and the rural Indian network. We also find similar levels of both security-awareness and risky behavior in the two environments. Users in both environments are unlikely to be participating in spamming or scanning and thus unlikely to be infected with malware using such vectors. Users are also risk-aware: many deploy anti-virus software and update their systems. However, doing so does not seem to reduce the likelihood of infections. On the other hand, we find that even though users are generally risk-aware, they regularly exhibit risky behavior: they contact websites flagged as malicious by Google's Safe Browsing API even though they must have been warned by their browsers. Interestingly, however, we find that such behavior only slightly increases their probability of being infected.

These findings seem contrary to conventional wisdom, as anti-virus software does not reduce the likelihood of infections and risky behavior increases it by less than one may presume. Furthermore, we find that users in rural areas of developing regions are no more likely to be infected and are as risk-aware as users in Europe.

Future work

As Internet traffic is rapidly changing over time, with new applications and services appearing frequently, we plan to track the long-term evolution of traffic characteristics and malicious activity. Furthermore, so far we limited our analysis of residential networks to a single urban area. In future work we plan to extend our studies to more network locations to characterize differences between different geographical areas as well as different access technologies (e. g., 3G cellular data access). We also plan to investigate interactive and real-time sensitive traffic such as VoIP and gaming. Although these do not yet contribute a significant number of bytes, these protocols are important for the perceived Quality-of-Service of the customer.

In the area of security analysis we also want to conduct measurements over longer periods of time and different points in the network. We also plan to add more signatures for common botnets in order to extend the coverage of malicious activity we are able to detect. Furthermore, we want to investigate why security-awareness and risky behavior do not correlate with different levels of malicious activity.

Acknowledgements

I would like to thank my PhD advisor, Anja Feldmann, for guiding me during the course of my PhD studies. The International Computer Science Institute (ICSI) in Berkeley, CA provided me with the opportunity for an intership during which I worked on parts of this thesis. I would also like to thank Vern Paxson, who was my advisor at ICSI.

Robin Sommer helped me understand the internal workings and design principles of the Bro NIDS. The security team at LBNL enabled us to perform measurement and security studies on their network data through the help of Robin Sommer, who ran my analysis scripts and tools at LBNL. Stefan Kornexl wrote the initial proof-of-concept version of the Time Machine [63]. Matthias Vallentin and the network operators of AirJaldi provided me with access to the AirJaldi traces and insights into AirJaldi's network architecture.

Finally, I would like to thank my family, friends, and colleagues for their support during my PhD studies.

List of Figures

List of Tables

Bibliography

[1] Abrams, M., Standridge, C. R., Abdulla, G., Williams, S., and Fox, E. A. Caching proxies: Limitations and potentials. In *Proceedings of the International World Wide Web Conference (WWW)* (1995).

[2] Adya, A., Bahl, P., and Qiu, L. Characterizing alert and browse services of mobile clients. In *Proceedings of the USENIX Annual Technical Conference* (2002).

[3] Ager, B., Schneider, F., Kim, J., and Feldmann, A. Revisiting cacheability in times of user generated content. In *Proceedings of the IEEE Global Internet Symposium* (2010).

[4] Aikat, J., Kaur, J., Smith, F. D., and Jeffay, K. Variability in TCP round-trip times. In *Proceedings of the Internet Measurement Conference (IMC)* (2003).

[5] AirJaldi Network. `http://www.airjaldi.org`.

[6] Alexa. `http://www.alexa.com/`.

[7] Allman, M., Paxson, V., and Terrell, J. A brief history of scanning. In *Proceedings of the Internet Measurement Conference (IMC)* (2007).

[8] Anderson, E., and Arlitt, M. Full packet capture and offline analysis on 1 and 10 Gb/s networks. Tech. Rep. HPL-2006-156, HP Labs, 2006.

[9] Anderson, N. P2P traffic drops as streaming video grows in popularity. `http://arstechnica.com/old/ content/2008/09/p2p-traffic-drops-as-streaming-video-grows-in-popularity.ars`, 2008.

[10] Antonelli, C., Co, K., Fields, M., and Honeyman, P. Cryptographic wiretapping at 100 megabits. In *Proceedings of the International Symposium on Aerospace Defense Sensing, Simulation, and Control* (2002).

[11] Arlitt, M., and Williamson, C. Internet web servers: Workload characterization and implications. *IEEE/ACM Transactions on Networking (ToN) 5*, 5 (1997).

[12] Armitage, G. J. Inferring the extent of network address port translation at public/private internet boundaries. Tech. Rep. 020712A, Center for Advanced Internet Architectures (CAIA), 2002.

[13] Bailey, M., Cooke, E., Jahanian, F., Xu, Y., and Karir, M. A survey of botnet technology and defenses. In *Proceedings of the Cybersecurity Applications & Technology Conference for Homeland Security* (2009).

[14] Barford, P., Bestavros, A., Bradley, A., and Crovella, M. Changes in web client access patterns: Characteristics and caching implications. *World Wide Web Journal 2* (1999).

[15] Bellovin, S. M. A technique for counting NATted hosts. In *Proceedings of the Internet Measurement Workshop (IMW)* (2002).

[16] Benko, P., Malicsko, G., and Veres, A. A large-scale, passive analysis of end-to-end TCP performance over GPRS. In *Proceedings of the Conference of the IEEE Computer and Communications Societies (INFOCOM)* (2004), vol. 3.

[17] Beverly, R. A robust classifier for passive TCP/IP fingerprinting. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2004).

[18] Blanton, E., and Allman, M. On making TCP more robust to packet reordering. *ACM SIGCOMM Computer Communication Review 32*, 1 (2002).

[19] Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., and Tofanelli, P. Revealing skype traffic: When randomness plays with you. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)* (2007).

[20] Carlinet, Y., Me, L., Debar, H., and Gourhant, Y. Analysis of computer infection risk factors based on customer network usage. In *Proceedings of the SECUWARE Conference* (2008).

[21] Casado, M., and Freedman, M. J. Peering through the shroud: The effect of edge opacity on IP-based client identification. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2007).

[22] Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., and Moon, S. I tube, you tube, everybody tubes. In *Proceedings of the Internet Measurement Conference (IMC)* (2007).

[23] Chandrasekaran, S., and Franklin, M. Remembrance of streams past: Overload-sensitive management of archived streams. In *Proceedings of the International Conference on Very Large Data Bases* (2004).

[24] Cho, K., Fukuda, K., Esaki, H., and Kato, A. The impact and implications of the growth in residential user-to-user traffic. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)* (2006).

[25] Cho, K., Fukuda, K., Esaki, H., and Kato, A. Observing slow crustal movement in residential user traffic. In *Proceedings of the ACM Conference on Emerging Networking Experiments And Technologies (CoNEXT)* (2008).

[26] ClearSight Networks. `http://www.clearsightnet.com`, 2008.

[27] CNet News. Another suspected NASA hacker indicted. `http://www.news.com/2102-7350_3-6140001.html`.

[28] CoMo. http://como.sourceforge.net, 2008.

[29] Conficker Working Group. http://www.confickerworkinggroup.org, 2010.

[30] Cooke, E., Myrick, A., Rusek, D., and Jahanian, F. Resource-aware multi-format network security data storage. In *Proceedings of the Workshop on Large Scale Attack Defense (LSAD)* (2006).

[31] Cranor, C., Johnson, T., and Spatscheck, O. Gigascope: A stream database for network applications. In *Proceedings of the ACM SIGMOD Conference* (2003).

[32] Crovella, M., and Bestavros, A. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking (ToN) 5*, 6 (1997).

[33] Dagon, D., Provos, N., Lee, C. P., and Lee, W. Corrupted DNS resolution paths: The rise of a malicious resolution authority. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2009).

[34] Desnoyers, P., and Shenoy, P. J. Hyperion: High volume stream archival for retrospective querying. In *Proceedings of the USENIX Annual Technical Conference* (2007).

[35] Dischinger, M., Haeberlen, A., Gummadi, K. P., and Saroiu, S. Characterizing residential broadband networks. In *Proceedings of the Internet Measurement Conference (IMC)* (2007).

[36] Dreger, H., Feldmann, A., Mai, M., Paxson, V., and Sommer, R. Dynamic application-layer protocol analysis for network intrusion detection. In *Proceedings of the USENIX Security Symposium* (2006).

[37] Dreger, H., Feldmann, A., Paxson, V., and Sommer, R. Operational experiences with high-volume network intrusion detection. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)* (2004).

[38] Dunlap, G. W., King, S. T., Cinar, S., Basrai, M. A., and Chen, P. M. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (2002).

[39] Emerging Threats. http://www.emergingthreats.net/, 2010.

[40] Endace Measurement Systems. http://www.endace.com/, 2010.

[41] Erman, J., Gerber, A., Hajiaghayi, M. T., Pei, D., and Spatscheck, O. Network-aware forward caching. In *Proceedings of the International World Wide Web Conference (WWW)* (2009).

[42] Feldmann, A., Caceres, R., Douglis, F., Glass, G., and Rabinovich, M. Performance of web proxy caching in heterogeneous bandwidth environments. In *Proceedings of the Conference of the IEEE Computer and Communications Societies (INFOCOM)* (1999), vol. 1.

[43] Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J., and True, F. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking (ToN) 9*, 3 (2001).

[44] Feldmann, A., Maier, G., Mühlbauer, W., and Rogoza, Y. Enabling seamless internet mobility. In *Proceedings of the IEEE Workshopp on Local and Metropolitan Area Networks (LANMAN)* (2008). Invited Paper.

[45] Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T., and Diot, S. C. Packet-level traffic measurements from the Sprint IP backbone. *IEEE Network Magazine 17*, 6 (2003).

[46] Fukuda, K., Cho, K., and Esaki, H. The impact of residential broadband traffic on Japanese ISP backbones. *ACM SIGCOMM Computer Communication Review 35*, 1 (2005).

[47] Gill, P., Arlitt, M., Li, Z., and Mahanti, A. Youtube traffic characterization: A view from the edge. In *Proceedings of the Internet Measurement Conference (IMC)* (2007).

[48] Golder, S., Wilkinson, D., and Huberman, B. A. Rhythms of social interaction: Messaging within a massive online network. In *International Conference on Communities and Technologies* (2007).

[49] Gonzalez, J. M., Paxson, V., and Weaver, N. Shunting: A hardware/software architecture for flexible, high-performance network intrusion prevention. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)* (2007).

[50] Google. Google safe browsing api. `http://code.google.com/apis/safebrowsing/`, 2010.

[51] Hao, S., Feamster, N., Gray, A., Syed, N., and Krasser, S. Detecting spammers with SNARE: spatio-temporal network-level automated reputation engine. In *Proceedings of the USENIX Security Symposium* (2009).

[52] Heikkinen, M., Kivi, A., and Verkasalo, H. Measuring mobile peer-to-peer usage: Case Finland 2007. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2009).

[53] Henderson, T., Kotz, D., and Abyzov, I. The changing usage of a mature campus-wide wireless network. In *Proceedings of the ACM SIGMOBILE Annual International Conference on Mobile Computing and Networking (MOBICOM)* (2004).

[54] Hyun-chul, K., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., and Lee, K. Internet traffic classification demystified: Myths, caveats, and the best practices. In *Proceedings of the ACM Conference on Emerging Networking Experiments And Technologies (CoNEXT)* (2008).

[55] Intelica Networks. `http://www.intelicanetworks.com`, 2008.

[56] ITU. ADSL standards ITU G.992.1-5.

[57] Jiang, H., and Dovrolis, C. Passive estimation of TCP round-trip times. *ACM SIGCOMM Computer Communication Review 32*, 3 (2002).

[58] Jung, J., Paxson, V., Berger, A., and Balakrishnan, H. Fast portscan detection using sequential hypothesis testing. In *Proceedings of the IEEE Symposium on Security and Privacy* (2004).

[59] Karagiannis, T., Broido, A., Brownlee, N., Claffy, K. C., and Faloutsos, M. Is P2P dying or just hiding? In *Proceedings of the IEEE GLOBECOM* (2004).

[60] Karagiannis, T., Rodriguez, P., and Papagiannaki, K. Should internet service providers fear peer-assisted content distribution? In *Proceedings of the Internet Measurement Conference (IMC)* (2005).

[61] Kim, J., Schneider, F., Ager, B., and Feldmann, A. Today's usenet usage: Characterizing NNTP traffic. In *Proceedings of the IEEE Global Internet Symposium* (Mar. 2010).

[62] Kitz.co.uk. Interlaving explained. `http://www.kitz.co.uk/adsl/interleaving.htm`.

[63] Kornexl, S., Paxson, V., Dreger, H., Feldmann, A., and Sommer, R. Building a time machine for efficient recording and retrieval of high-volume network traffic (short paper). In *Proceedings of the Internet Measurement Conference (IMC)* (2005).

[64] Kotz, D., and Essien, K. Characterizing usage of a campus-wide wireless network. Tech. rep., Departement of Computer Science, Dartmouth College, 2002.

[65] Kotz, D., and Essien, K. Analysis of a campus-wide wireless network. *Wireless Networks Journal 11*, 1-2 (2005).

[66] Krishnamurthy, B., and Rexford, J. *Web Protocols and Practice*. Addison-Wesley, 2001.

[67] Kurose, J., and Ross, K. *Computer Networking: A Top-Down Approach*, fifth ed. Addison Wesley, 2009.

[68] L7filter. Application layer packet classifier for linux. `http://l7-filter.sourceforge.net/`.

[69] Labovitz, C., McPherson, D., and Iekel-Johnson, S. 2009 Internet observatory report. In *North American Network Operators' Group (NANOG) Meeting 47* (2009).

[70] Lee, Y. Measured TCP performance in CDMA 1x EV-DO network. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2006).

[71] Maier, G., Feldmann, A., Paxson, V., and Allman, M. On dominant characteristics of residential broadband internet traffic. In *Proceedings of the Internet Measurement Conference (IMC)* (2009).

[72] Maier, G., Mühlbauer, W., Rogoza, Y., and Feldmann, A. Enabling seamless internet mobility. In *Proceedings of the ACM Conference on Emerging Networking Experiments And Technologies (CoNEXT) Student Workshop* (2007).

[73] Maier, G., Schneider, F., and Feldmann, A. A first look at mobile hand-held device traffic. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2010).

[74] Maier, G., Sommer, R., Dreger, H., Feldmann, A., Paxson, V., and Schneider, F. Enriching network security analysis with time travel. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)* (2008).

[75] Mathis, M., Semke, J., Mahdavi, J., and Ott, T. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review 27*, 3 (1997).

[76] McGrath, K. P., and Nelson, J. Monitoring and forensic analysis for wireless networks. In *Proceedings of the International Conference on Internet Surveillance and Protection (ICISP)* (2006).

[77] Medina, A., Allman, M., and Floyd, S. Measuring the evolution of transport protocols in the internet. *ACM SIGCOMM Computer Communication Review 35*, 2 (2004).

[78] Miller, T. Passive OS fingerprinting: Details and techniques. `http://www.ouah.org/incosfingerp.htm`, 2005.

[79] Moore, D., Shannon, C., and claffy, k. Code-red: A case study on the spread and victims of an internet worm. In *Proceedings of the Internet Measurement Workshop (IMW)* (2002).

[80] Nazir, A., Raza, S., and Chuah, C.-N. Unveiling facebook: A measurement study of social network based applications. In *Proceedings of the Internet Measurement Conference (IMC)* (2008).

[81] netzikon. Zwangstrennung. `http://netzikon.net/lexikon/z/dsl-zwangstrennung.html`, 2009. In German.

[82] OECD. OECD broadband portal. `http://www.oecd.org/sti/ict/broadband`, 2008.

[83] Padhye, J., Firoiu, V., Towsley, D., and Kurose, J. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)* (1998).

[84] Park, K., Kim, G., and Crovella, M. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proceedings of the IEEE International Conference on Network Protocol* (1996).

[85] Parker, A. Cachelogic: P2P media summit.

[86] Paxson, V. Bro: A system for detecting network intruders in real-time. *Computer Networks Journal 31*, 23–24 (1999). Bro homepage: `www.bro-ids.org`.

[87] Paxson, V., and Floyd, S. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking (ToN) 3*, 3 (1995).

[88] Phaal, P. Detecting NAT devices using sFlow. `http://www.sflow.org/detectNAT/`, 2009.

[89] Ponec, M., Giura, P., Brönnimann, H., and Wein, J. Highly efficient techniques for network forensics. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)* (2007).

[90] Porras, P., Saidi, H., and Yegneswaran, V. An analysis of Conficker's logic and rendezvous points. Tech. rep., SRI International, 2009.

[91] Provos, N., Mavrommatis, P., Rajab, M. A., and Monrose, F. All your iFRAMEs point to us. In *Proceedings of the USENIX Security Symposium* (2008).

[92] Ramachandran, A., and Feamster, N. Understanding the network-level behavior of spammers. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)* (2006).

[93] Ramachandran, A., Feamster, N., and Vempala, S. Filtering spam with behavioral blacklisting. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)* (2007).

[94] Rapidshare. `http://www.rapidshare.com/`, 2009.

[95] Reiss, F., Stockinger, K., Wu, K., Shoshani, A., and Hellerstein, J. M. Enabling real-time querying of live and historical stream data. In *Proceedings of the International Conference on Statistical and Scientific Database Management (SSDBM)* (2007).

[96] Rigney, C., Willens, S., Livingston, Rubens, A., Merit, Simpson, W., and Daydreamer. Remote authentication dial in user service (RADIUS). RFC 2865, 2000.

[97] Roesch, M. Snort – Lightweight intrusion detection for networks. In *Proceedings of the Systems Administration Conference (LISA)* (1999).

[98] Sandvine. 2009 Global broadband phenomena. `http://www.sandvine.com/downloads/documents/2009\%20Global\%20Broadband\%20Phenomena\%20-\%20Executive\%20Summary.pdf`, 2009.

[99] Schneider, F., Agarwal, S., Alpcan, T., and Feldmann, A. The new web: Characterizing ajax traffic. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2008).

[100] Schneider, F., Feldmann, A., Krishnamurthy, B., and Willinger, W. Understanding online social network usage from a network perspective. In *Proceedings of the Internet Measurement Conference (IMC)* (2009).

[101] Schneider, F., Wallerich, J., and Feldmann, A. Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2007).

[102] Schulze, H., and Mochalski, K. Ipoque: Internet study 2007.

[103] Schulze, H., and Mochalski, K. Ipoque: Internet study 2008/2009.

[104] Shanmugasundaram, K., Memon, N., Savant, A., and Brönnimann, H. ForNet: A distributed forensics network. In *Proceedings of the International Workshop on Mathematical Methods, Models, and Architectures for Computer Networks Security (MMM-ACNS)* (2003).

[105] Siekkinen, M., Collange, D., Urvoy-Keller, G., and Biersack, E. Performance limitations of ADSL users: A case study. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2007).

[106] Siekkinen, M., Urvoy Keller, G., Biersack, E., and Collange, D. A root cause analysis toolkit for TCP. *Computer Networks Journal 52*, 9 (2008).

[107] Sommer, R. *Viable Network Intrusion Detection in High-Performance Environments*. PhD thesis, TU München, 2005.

[108] Sommer, R., and Paxson, V. Exploiting independent state for network intrusion detection. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)* (2005).

[109] SORBS. `http://www.au.sorbs.net`, 2010.

[110] Spamhaus Project. `http://www.spamhaus.org`, 2010.

[111] Sripanidkulchai, K., Maggs, B., and Zhang, H. An analysis of live streaming workloads on the internet. In *Proceedings of the Internet Measurement Conference (IMC)* (2004).

[112] Stone-Gross, B., Kruegel, C., Almeroth, K., Moser, A., and Kirda, E. FIRE: FInding Rogue nEtworks. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)* (2009).

[113] Svoboda, P., Ricciato, F., Pilz, R., and Hasenleithner, E. Composition of GPRS, UMTS traffic: snapshots from a live network. In *Proceedings of the Workshop on Internet Performance (IPS-MOME)* (2006), Salzburg Research Forschungsgesellschaft.

[114] Tanenbaum, A. S. *Computer Networks*, fourth ed. Prentice Hall Professional Technical Reference, Upper Saddle River, NJ, USA, 2003.

[115] Timmins, P., McCormick, S., Agu, E., and Wills, C. Characteristics of mobile web content. *Proceedings of the IEEE Workshop on Hot Topics in Web Systsems and Technologies (HotWeb)* (2006).

[116] Tomlinson, D. Plusnet report: More record breaking streaming and the latest iplayer news. `http://community.plus.net/blog/2008/07/17/more-record-breaking-streaming-and-the-latest-iplayer-news/`.

[117] Trestian, I., Ranjan, S., Kuzmanovic, A., and Nucci, A. Measuring serendipity: connecting people, locations and interests in a mobile 3G network. In *Proceedings of the Internet Measurement Conference (IMC)* (2009).

[118] Universität Bonn. `http://http://net.cs.uni-bonn.de/wg/cs/applications/containing-conficker/`, 2010.

[119] Vallentin, M., Sommer, R., Lee, J., Leres, C., Paxson, V., and Tierney, B. The nids cluster: Scalable, stateful network intrusion detection on commodity hardware. In *Proceedings of the Symposium on Recent Advances in Intrusion Detection (RAID)* (2007).

[120] Veal, B., Li, K., and Lowenthal, D. New methods for passive estimation of TCP round-trip times. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2005).

[121] Verkasalo, H. Empirical observations on the emergence of mobile multimedia services and applications in the U.S. and Europe. In *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia* (2006).

[122] Wallerich, J., Dreger, H., Feldmann, A., Krishnamurthy, B., and Willinger, W. A methodology for studying persistency aspects of internet flows. *ACM SIGCOMM Computer Communication Review 35*, 2 (2005).

[123] Weaver, R. A probabilistic population study of the Conficker-C botnet. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2010).

[124] Wikipedia. Smartphone: Operating systems. `http://en.wikipedia.org/w/index.php?title=Smartphone&oldid=315621107`, 2009.

[125] Wikipedia. Zwangstrennung. `http://de.wikipedia.org/w/index.php?title=Zwangstrennung&oldid=56733242`, 2009. In German.

[126] Wikipedia. Zlob trojan. `http://en.wikipedia.org/w/index.php?title=Zlob_trojan\&oldid=357679256`, 2010.

[127] Williamson, C., Halepovic, E., Sun, H., and Wu, Y. Characterization of CDMA2000 cellular data network traffic. In *Proceedings of the IEEE Conference on Local Computer Networks (LCN)* (2005).

[128] Wills, C., and Mikhailov, M. Studying the impact of more complete server information on Web caching. In *Proc. of the 5th International Web Caching and Content Delivery Workshop* (2000).

[129] Won, Y. J., Park, B.-C., Hong, S.-C., Jung, K. B., Ju, H.-T., and Hong, J. W. Measurement analysis of mobile data networks. In *Proceedings of the International Conference on Passive and Active Measurement (PAM)* (2007).

[130] Wood, N. Mobile data traffic growth 10 times faster than fixed over next five years. In Total Telecom: `http://www.totaltele.com/view.aspx?ID=448681`, 2009.

[131] Xie, Y., Yu, F., and Abadi, M. De-anonymizing the internet using unreliable ids. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)* (2009).

[132] Xie, Y., Yu, F., Achan, K., Gillum, E., Goldszmidt, M., and Wobber, T. How dynamic are IP addresses? In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)* (2007).

[133] Yu, H., Zheng, D., Zhao, B. Y., and Zheng, W. Understanding user behavior in large-scale video-on-demand systems. *ACM SIGOPS Operating Systems Review 40*, 4 (2006).

[134] ZeuS Tracker. `https://zeustracker.abuse.ch`, 2010.

[135] Zhang, Y., Breslau, L., Paxson, V., and Shenker, S. On the characteristics and origins of internet flow rates. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)* (2002).

[136] Zink, M., Suh, K., Gu, Y., and Kurose., J. Watch global, cache local: Youtube network traces at a campus network - measurements and implications. In *Proceedings of the Annual Multimedia Computing and Networking (MMCN)* (2008).