Technische Universität Berlin
Fakultät für Elektrotechnik und Informatik
Lehrstuhl für Intelligente Netze
und Management Verteilter Systeme

Analysis of New Trends in the Web from a Network Perspective

vorgelegt von
Fabian Schneider (Dipl. Inf.)

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:

Vorsitzende:	Prof. Dr. Ina Schieferdecker, TU Berlin
Berichtende:	Prof. Anja Feldmann, Ph. D., TU Berlin
Berichtender:	Prof. Dr. habil. Odej Kao, TU Berlin
Berichtender:	Walter Willinger, Ph. D., AT&T Labs–Research

Tag der wissenschaftlichen Aussprache: 30. März 2010

Berlin 2010
D83

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

Datum Fabian Schneider

Abstract

Over the last five years, several trends have changed the landscape of the World Wide Web, forming the new "Web 2.0". The advent of user generated content (blogs and wikis), the popularity of multimedia (e. g., YouTube and MySpace), and the penetration of Google's services (maps, mail, etc.) are commonly noticeable. In particular, the recent popularity of Online Social Networks (OSNs, e. g., Facebook and LinkedIn) has caused a fundamental change in how the Internet is used. For example, certain OSN users are only using the OSN internal messaging instead of email. This motivates us to examine the usage of these new Web trends and determine their impact on the network.

First, we present a traffic study of several Web 2.0 applications including Google Maps, modern Web-based email, and social networking Websites, and compare their traffic characteristics with the ambient HTTP traffic. We highlight the key differences between Web 2.0 traffic and all HTTP traffic. As such, our work elucidates the changing face of one of the most popular applications on the Internet: The World Wide Web. We find that "Web 2.0" applications unleash new HTTP traffic patterns which differ from the conventional HTTP request-response model. In particular, asynchronous pre-fetching of data in order to provide a smooth Web browsing experience and richer HTTP payloads (e. g., JavaScript libraries) of Web 2.0 applications induce larger, heavier, and more bursty traffic on the underlying networks.

Next, we focus on Online Social Networks. OSNs have already attracted more than half a billion users. However, our understanding of which OSN features attract and keep the attention of these users is poor. Studies thus far have relied on surveys or interviews of OSN users or focused on static properties, e. g., the friendship graph, gathered via sampled crawls. In this thesis, we study how users actually interact with OSNs by extracting anonymized clickstreams from passively monitored network traffic. Our characterization of user interactions within the OSN for four different OSNs (Facebook, LinkedIn, Hi5, and StudiVZ) focuses on feature popularity, session characteristics, and the dynamics within OSN sessions. We find, for example, that users commonly spend more than half an hour interacting with the OSN. Yet, the byte contributions per OSN session are relatively small.

Subsequently, we look into mobile hand-held device (MHD) usage that is observed when such devices are used at home. Our characterization of the traffic shows that mobile Apple devices (i. e., iPhones and iPods) are, by a huge margin, the most commonly used MHDs and account for most of the traffic. We find that MHD traffic is dominated by multimedia content and downloads of mobile applications.

Finally, inspired by the finding that Network News Transport Protocol (NNTP) traffic is responsible for up to 5 % of residential network traffic we investigate today's Usenet usage. We find that NNTP is intensively used by a small fraction of the residential broadband lines that we study and that almost all traffic is originated

by NNTP servers that require a monthly fee subscription. The accessed content resembles what one might expect from file-sharing systems—archives and multimedia files. Accordingly, it appears that NNTP is used by some as a high performance alternative to traditional P2P file-sharing options such as eDonkey or BitTorrent.

The analyses of this thesis are based on anonymized packet level recordings of real Internet traffic collected at different vantage points and from different user populations. From these recordings we extract traces of protocol events and activities (e. g., new TCP connections or application layer requests and responses). Next, all traffic related to a new trend is identified and then compared to the overall traffic. This allows us to predict the impact on the traffic in case these trends increase or decrease in popularity.

Packet capture in high speed networks is challenging. Limitations (e. g., bus or memory bandwidth, OS capturing stack) prevent comprehensive packet capture in these environments. The endeavor to perform packet capture with commodity hardware requires us to identify and then overcome some of the performance limitations. Knowing the limitations, we propose several possibilities to split or reduce the analysis load.

Zusammenfassung

In den letzten fünf Jahren haben verschiedene Trends die Landschaft des World Wide Web hin zum „Web 2.0“ verändert. Das Aufkommen von Inhalten, die von den Benutzern selbst erzeugt werden (engl. user generated content, z. B. in Blogs und Wikis), die Beliebtheit von Videos und Musik (z. B. in YouTube und MySpace) und die Verbreitung von Googles Diensten (z. B. Google Maps oder Google Mail) sind allgemein bemerkbar. Gerade die aktuell sehr große Beliebtheit sozialer Netze im Internet (engl. Online Social Networks, OSN), wie Facebook oder StudiVZ, verursacht einen grundlegenden Wechsel in der Art, wie das Internet genutzt wird. Zum Beispiel benutzen einige OSN-Nutzer nur noch das OSN-interne Nachrichten-System anstatt E-Mail. Dies motiviert uns, die Nutzung dieser Trends zu untersuchen und deren Auswirkungen auf den Netzverkehr zu bestimmen.

Wir beginnen mit einer Untersuchung verschiedener Web-2.0-Anwendungen und vergleichen die Charakteristiken des Web-2.0-Verkehrs mit allgemeinem HTTP-Verkehr. Als Anwendungen haben wir Google Maps, zwei Web-basierte E-Mail Dienste und ein OSN gewählt, die AJAX nutzen. Der Vergleich mit dem umgebenden HTTP-Verkehr hebt die Veränderung der Nutzung eines der populärsten Protokolle des Internet, nämlich des World Wide Web, hervor. Unsere Ergebnisse zeigen, dass Web-2.0-Anwendungen neue HTTP-Verkehrsstrukturen verursachen, die sich vom konventionellem HTTP-Anfrage-Antwort-Modell unterscheiden. Asynchrones Laden von Daten, die später nur vielleicht benötigt werden, erlaubt eine flüssige Benutzung der Dienste. Dies wird durch kleine JavaScript-Programme erreicht, die im Browser ausgeführt werden. Verbindungen von AJAX-Anwendungen übertragen mehr Daten, dauern länger und verursachen öfter höhere Verkehrsspitzen.

Danach untersuchen wir OSNs, motiviert durch die große Anzahl – eine halbe Milliarde – ihrer Benutzer. Trotz der hohen Nutzerzahlen ist unser Wissen darüber, welche OSN-Funktionen Benutzer faszinieren und deren Interesse erhalten, sehr beschränkt. Bislang waren Studien hierzu auf Umfragen und Befragungen beschränkt oder haben sich auf statistische Eigenschaften wie Freundschaftsgraphen bezogen. In dieser Arbeit ermitteln wir die tatsächliche Nutzung von OSNs durch Extraktion der HTTP-Anfrage- und -Antwortsequenzen aus passiven Aufzeichnungen des Netzverkehrs. Wir analysieren für vier OSNs (Facebook, LinkendIn, Hi5 und StudiVZ) die Beliebtheit ihrer OSN-Funktionen, die Eigenschaften der OSN-Sitzungen, sowie die Aktivität innerhalb von Sitzungen. Unsere Ergebnisse zeigen, dass Benutzer ungefähr eine halbe Stunde online sind, aber nur wenig Datenvolumen verursachen.

Daraufhin analysieren wir die Nutzung mobiler Geräte (z. B. des iPhones oder des Blackberrys), wenn diese zu Hause mittels DSL benutzt werden. In unseren Daten sehen wir fast ausschließlich iPhones und iPods, die auch für einen Großteil des Verkehrsvolumens verantwortlich sind. Die von diesen Geräten herunter geladenen Inhalte sind hauptsächlich Multimedia-Dateien oder mobile Anwendungen.

Obwohl NNTP[1] nur von einer kleinen Zahl Benutzer verwendet wird, werden darüber bis zu 5 % des gesamten Verkehrsvolumens in Aggregationsnetzen von Breitbandkunden übertragen. Motiviert durch diese Erkenntnis wenden wir uns im Folgenden dem Usenet zu. Die meisten Verbindungen gehen zu Servern, die eine monatliche Gebühr für die Nutzung erheben. Des Weiteren sind 99 % der übertragenen Inhalte typische File-Sharing-Inhalte wie Archive und Multimedia-Dateien. Es scheint, dass NNTP eine leistungsfähige Alternative zu bisherigen P2P-File-Sharing-Optionen wie BitTorrent oder eDonkey ist.

Unsere Analysen basieren auf anonymisierten Paketaufzeichnungen realen Internetverkehrs an verschiedenen Orten und verschiedener Benutzergruppen. Aus diesen Aufzeichnungen extrahieren wir Aktionen der Anwendungsprotokollebene, z. B. Aufbau einer neuen TCP-Verbindung oder Anfrage eines Objekts. Danach muss der gesamte Verkehr, der zu dem gerade untersuchten Trend gehört, bestimmt werden, um diesen mit dem restlichen Verkehr zu vergleichen. Dies erlaubt es vorherzusagen, welchen Einfluss die an- oder absteigende Beliebtheit eines Trends hat.

Die Aufzeichnung von Paketen in Hochgeschwindigkeitsnetzen ist eine Herausforderung. Insbesondere, wenn dies mit normalen Hardwarekomponenten geschehen soll, müssen deren Leistungsgrenzen (Bus- oder Speicherbandbreite, Aufzeichnungsfunktionen des Betriebssystems usw.) identifiziert werden, um danach die richtigen Komponenten auszuwählen. Unsere Erfahrung zeigt, dass trotzdem meistens die Leistung nicht ausreichend ist. Daher schlagen wir verschiedene Möglichkeiten vor, die Analyselast zu verteilen oder zu reduzieren.

[1] NNTP ist das Network News Transfer Protocol.

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 hand-held device traffic. In *Proceedings of the 11th International Conference on Passive and Active Measurement (PAM)* (Apr. 2010), vol. 6032 of *Lecture Notes in Computer Science*, Springer, pp. 161–170

Kim, J., Schneider, F., Ager, B., and Feldmann, A. Today's usenet usage: Characterizing NNTP traffic. In *Proceedings of the 13th IEEE Global Internet Symposium* (Mar. 2010)

Ager, B., Schneider, F., Kim, J., and Feldmann, A. Revisiting cacheability in times of user generated content. In *Proceedings of the 13th IEEE Global Internet Symposium* (Mar. 2010)

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)* (Nov. 2009), ACM Press, pp. 35–48

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* (Aug. 2008), ACM Press, pp. 183–194

Schneider, F., Agarwal, S., Alpcan, T., and Feldmann, A. The new Web: Characterizing AJAX traffic. In *Proceedings of the 9th International Conference on Passive and Active Measurement (PAM)* (Apr. 2008), vol. 4979 of *Lecture Notes in Computer Science*, Springer, pp. 31–40

Schneider, F., Wallerich, J., and Feldmann, A. Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware. In *Proceedings of the 8th International Conference on Passive and Active Measurement (PAM)* (Apr. 2007), vol. 4427 of *Lecture Notes in Computer Science*, Springer, pp. 207–217

Workshops and Poster Sessions

Ager, B., Schneider, F., and Feldmann, A. Cacheability of bulk content for ISPs. In *Poster Session of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Aug. 2009)

Schneider, F., and Wallerich, J. Performance evaluation of packet capturing systems for high-speed networks. In *Proceedings of the ACM Conference on Emerging Networking Experiments And Technologies (CoNEXT) Student Workshop* (Oct. 2005), pp. 284–285

Contents

Chapter 1

Introduction

In this thesis, we examine new trends in Internet usage (e. g., AJAX, Online Social Networks, NNTP, and mobile hand-held devices) from a network perspective. It is crucial to understand both how the Internet is used and the effect on the network. The analyses of this thesis are based on anonymized packet level recordings of actual Internet traffic collected at different vantage points and from different user populations. From these recordings we extract anonymized traces of protocol events and actions.

1.1 Motivation

The Internet emerged roughly 40 years ago from a research project with the goal of interconnecting a small set of supercomputers. Quickly, other universities joined the ARPANET and companies created networks all over the world. The first specification of IP and TCP paved the way for a single world-wide network—the Internet. Today, one quarter of the world's population and more than half of the citizens of developed countries have access to the Internet. With the advent of cellular and wireless networks, even wired connections are no-longer strictly necessary.

If there is any constant during the evolution of the Internet it is "change". In the beginning, the change was localized to improvements in the few protocols used for communication, e. g., window scaling, loss recovery, and RTT measurement have been added to TCP; later on persistent connections and pipelining found their way into HTTP. Presently, the way content is provided and presented to the masses is changing rapidly. Starting from text-only capabilities like message exchange and file-transfer, the Internet today is used for video conferences, television streaming, and presenting or exposing ourselves to the public. While dominating and popular protocols (e. g., TCP/IP and SMTP) date back to the beginnings of Internet, there are also new application layer protocols such as HTTP, and BitTorrent, which are well-established today. In addition, the range and the composition of interests of the Internet population has also changed dramatically. Reasons include reduced barriers to access and the ease of use of today's Internet capable devices, such as the iPhone.

In summary, the Internet is changing from year to year. New applications emerge and old ones fade away. With every change various entities ask the question: Will it impact my future plans or business operations, and if so, how? For any new trend it is therefore important to understand its implication on the traffic, its impact on the network, and its consequences for the users. For researchers this is important for debugging network and application performance as well as for improving the protocols underlying the applications. Based on measurements, researchers can develop usage models to evaluate novel network mechanisms and protocols. Network operators and Internet service providers are interested in traffic and usage patterns for network planing and dimensioning as well as future trend prediction. Internet application developers can use our insights to tune the performance of their applications to the relevant features.

In the past five years, several new trends have emerged in the Internet: There are some well-known ones like the iPhone, YouTube, Facebook, and Google's services but also some not so obvious ones such as User Generated Content (UGC), a new paradigm for Web-based services, AJAX, a framework that allows execution of Web applications in the browser, or NNTP with a commercialized revival of binary groups. We therefore in this thesis study a subset of these phenomenon: AJAX-based applications, Online Social Networks (OSNs), mobile hand-held devices (MHDs), and NNTP. We focus on traces of network activity that have been recorded from real-world operational networks in order to analyze the trends from a network perspective. Moreover, we study the effects of the trends on the network.

We find that AJAX-based applications add another level of automation to HTTP which reduces the inter-request times per user dramatically, possibly requiring bigger queues in network nodes. For Online Social Networks our results can be used to build an OSN user model that can be used to evaluate new distributed OSN designs. The most popular features in OSNs are messaging, profile browsing, and managing photos. Moreover, the activity periods tend to be short—in the order of five minutes—while authenticated session durations reach days and more. In terms of mobile hand-held usage we find that iPhones and iPods currently dominate the smart phone usage at home and their users prefer multimedia content and mobile applications. Finally, our investigation of NNTP reveals a revival of the Usenet due to fee-based servers offering binary content.

Since we rely on passive measurements obtained at high speed links we need to select suitable measurement equipment. We present a methodology for performance evaluation of different packet capturing systems. Based on the results we make a set of recommendations on how to compose a high performance packet capturing infrastructure. The insights obtained from our performance evaluation are also important for designing systems to secure a network, which e. g., rely on deep packet inspection.

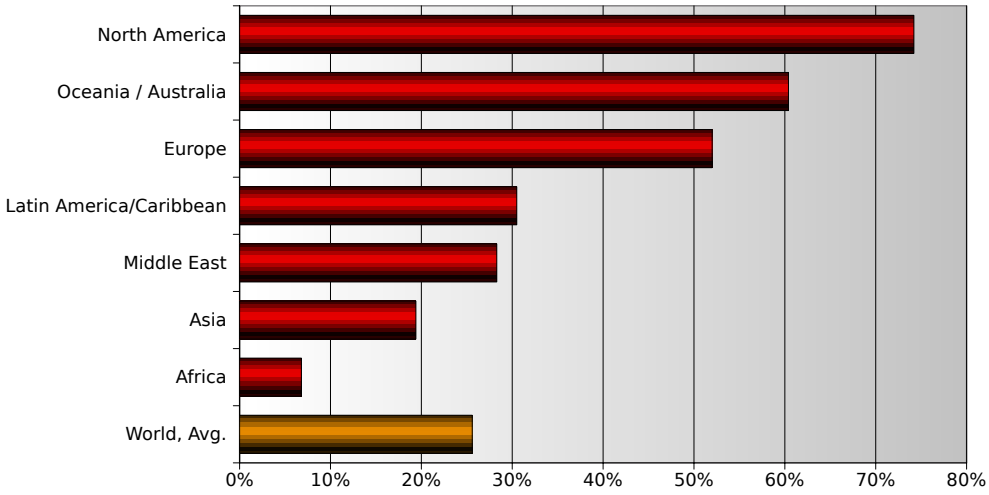


Figure 1.1: **Histogram of Internet penetration rates for various geographic regions.** (Data Source: Internet World Stats [41], based on a world population of 6,767,805,208 and 1,733,993,741 estimated Internet users for September 30, 2009.)

1.2 Aims and challenges

Our main goal is to understand how the Internet is used today. This has two major aspects: What is the user doing and how does this usage translate into network traffic. In order to study what a user—a human—is doing one may think that computer science is not the most applicable discipline. Indeed, social sciences or psychology have a long tradition in studying human behavior. Their basic approach is to select a trial group of humans and ask them to perform certain tasks while monitoring their behavior. A major drawback of this approach is the limited scope and scalability of these experiments. It is rather hard to perform such studies with thousands of participants.

Though, running experiments and measuring the behavior of tens of thousands or even more users is viable on the Internet. Recent statistics [41] report that roughly 1.7 billion people are connected to the Internet—one quarter of the world's total population, see Figure 1.1. Measuring usage on the Internet therefore may enable one to infer characteristics that are based on many more observations and that may therefore be statistically more representative. However, so far collaborations between the Internet measurement community and other disciplines are limited.

For this thesis, we leverage our knowledge and expertise in network measurement to assess Internet usage from passively observed network traffic. Monitoring passively has the additional advantage of minimal bias as the subject under study usually does not know that it is currently monitored. In order to preserve privacy, all

efforts are taken to discard personal information as early as possible and to prevent disclosure[1].

In addition to the ability to characterize human behavior we also want to determine the effect of each trend on the network itself. To this end, passively collected measurements are necessary. But there is one question: Were to place the monitors? Monitors can either be placed on low bandwidth links which connect only a handful of users to the Internet. Or monitors can be put on high bandwidth links that connect many users. In terms of representativeness of the results we prefer the latter.

This directly leads us to the first technical challenge we are facing: Performing measurements in contemporary high-speed networks with an academic budget. This is difficult for several reasons: *(i)* Off-the-shelf hardware is limited in terms of internal bus and memory bandwidth as well as processing power. Commodity servers are powerful machines but data capture and analysis at 10 or even 40 Gigabit per second at line-speed requires even more. *(ii)* Developing specialized hardware is time consuming. Therefore we revisit the measurement tradeoffs and available hardware in detail and design a suitable measurement infrastructure in Chapter 3.

Another challenge is to identify "new trends" on the Internet early on. Again, we rely on Internet measurements. One key for the identification of new trends is to understand the application mix in the Internet. For this purpose, we can use already available studies that report on the application mix [51, 85, 92, 93]. In particular, note the results of Maier et al. [60] as their study is based on the same data sets that we use throughout this thesis.

Once a new trend is identified the next challenge is to infer its usage from the collected network traces. Classification of the traffic that pertains to the trend is the first step. Then all traffic originated from one user is grouped together and usage is reverse-engineered from the data. Experience shows that it pays off to first explore the application under study in depth. Omitting this can lead to re-implementing an analysis multiple times. The final goal is to map the network activity to the actions a user performed.

1.3 Data sources and vantage points

Our approach is to passively observe user activity from a network perspective. For this purpose we need access to network data. In the following we discuss the appli-

[1] For this study, we strictly adhere to all legal regulations. We also limit the group that has access to the anonymized data to the bare minimum. Being aware of the sensitivity of the data for every analysis we first consider the privacy implications of each step. In order to prevent data leakage all sensitive data must reside on specially secured hardware which is not accessible from the Internet.

cability of different available network data sources. Subsequently, we discuss where to monitor the traffic.

Flow data

Flow data summarizes network traffic on a connection level. All packets belonging to a "flow"[2] are aggregated into a record that yields various information, e. g., number of packets, volume of the flow, or the TCP flags that were present in that flow. Flow data is often used for traffic engineering, routing optimization, demand analysis, or network dimensioning. Although flow data is an important data source to understand how much data is flowing from where to where, it usually does not expose enough information to easily attribute a flow with the Internet application that caused the data transfer [14, 67, and references therein]. Furthermore, it gives only a high level picture and does not allow to study what is happening inside a connection. Moreover, due to performance limitations on the routers that export the flow records, flow data is often sampled. This might bias the results as short flows have a higher chance of not being included in the sample than long flows.

Packet header traces

Packet header traces provide more detailed information than flows since per packet information is available. One can think of packet header traces as flows with in-connection timing information. This allows us to determine when there is activity during a connection. It still does not enable unambitious identification of the application causing the connection nor determination what happened on the application layer. Packet header traces are very useful for hardware dimensioning and queueing analysis.

Server logs

Server logs overcome the problem of not knowing what happens on the application layer. It is also easy to determine the application itself as one knows which server is contacted. Unfortunately, operators of servers of novel and popular applications (e. g., Google or Facebook) are usually unwilling to share their server logs with researchers. A solution to this problem is to create ones own novel application and run it on monitored servers. On the one hand this is difficult in terms of providing a scalable infrastructure for a large user base. On the other hand many users need to be attracted by the service to ensure representativity. In addition, not every new trend uses servers, e. g., P2P based applications.

[2]The most common definition of a flow is a set of packets featuring the same source and destination IP addresses, same source and destination port, and the same transport protocol.

Full packet traces

Full packet traces include the maximum of network information that can be collected passively. Thus, full packet traces can be used to gather every other type of data source. The huge drawbacks of full packet traces are confidentiality and their storage requirements. For example, we collect up to six TeraBytes of full packets from a single 1-Gigabit link per day. If one only stores the packet headers this requires an order of magnitude less space[3]. For comparison: Unsampled Flows require two to three orders of magnitude less space than full packet traces, depending on the aggregation scheme and the compression level.

Application layer headers

Application layer headers can include headers of all layers ranging from network over transport to application layer. The payload that is carried by the application layer protocol is discarded. In HTTP, for example, the application layer headers include IP addresses, port numbers, and all HTTP commands and HTTP header information, e. g., a `GET /index.html` command including the URL and all HTTP headers (i. e., `Host:`, `Cookie:`, etc.) that follow the command—but not the HTML page that is requested. This information is sufficient for reverse-engineering the actions and interests of the users. Application layer headers are also well suited for per session traffic characteristics. Application layer headers can be extracted from full packet traces or by directly extracting them from live traffic via an application layer analyzer. For HTTP the storage requirement is roughly 1 % compared to full packet traces.

We use application layer headers because they still contain enough information to characterize the usage and the traffic of the application but allow faster processing and have smaller storage requirements than full packet traces. In addition, discarding the payload significantly reduces the privacy sensitivity of the data.

Location of the monitors

Besides implications on the required performance of the measurement system, the location of the monitors also needs to match the requirements of the analysis. We use the protocol analyzers of the Bro network intrusion detection system (see Section 2.2.3) for extracting the application layer header. In order to correctly parse the network data, Bro needs to monitor both directions of the (TCP) connection. Moreover, missing packets within a connection force the analyzer to stop and skip the analysis for that connection.

Options for placing the monitor are either close to the end host (edge), in the network core, or anywhere in between. Just tracing all traffic of a single end host does not fulfill the requirement of being representative, and thus does not overcome

[3] In our traces we observe an average packet size of 685 bytes. Assuming a header size of 68 bytes we require 1/10 of the size of a full packet trace.

the limitations of surveys and studies on real-life people. Using a monitor somewhere close to the core of the Internet also does not help since routing in the Internet is typically asymmetric. Therefore, it is very unlikely that such a monitor is going to record both directions of the traffic flow. In addition, paths in the Internet change due to routing updates. This may result in connections for which only one part is observed and cause missing packets.

For some analyses it is important to not only monitor the activity of users with one specific Web service, but to observe all activity of a user. For example in the MHD and NNTP analyses we do want to observe all MHD or NNTP traffic that one user initiates, in order to understand which content is popular. It is unlikely that one can monitor all traffic of a subset of the Internet users if the monitor is close to the core of the network. This is because our monitor is placed on a link that is not included in all paths between client and all the different servers. Therefore, it is important to select a vantage point that allows us to monitor both *(i)* a large population thus forcing us to use a monitor closer to the core of the network and *(ii)* all activity of a user forcing us to choose a location close to the edge of the network. A compromise is to place the monitor close to the router at the boundary between network core and aggregation network.

1.4 Structure of the thesis

This chapter motivates the topic of this thesis and introduces our general approach. In Chapter 2 we give an overview on common traffic characteristics and introduce application layer analysis tools. Next, we evaluate the performance of packet capturing systems and design our measurement infrastructure in Chapter 3. Based on the recommended infrastructure we perform our analysis of new trends. We begin with assessing the impact of AJAX-enabled Web applications in Chapter 4. In Chapter 5, we turn to a recent emergence on the Internet: Online Social Networks. Subsequently, we investigate the usage of mobile hand-held devices when used at home via DSL in Chapter 6. As last trend that we study in this thesis, we analyze the revival of binary content in the Usenet in Chapter 7. Finally, we summarize and sketch future work in Chapter 8.

Chapter 2

Background

Before delving into the details of our analyses we begin with an overview of typical characteristics of Internet traffic. In Section 2.1 we introduce the mix of applications and content-types as well as traffic patterns. Next, we briefly outline tools commonly used for traffic analysis in Section 2.2.

Instead of acquainting the reader with each of the studied trends, in this section we refer to the introduction and related work sections in the corresponding chapter. More specifically, see Section 3.1 on limitations of packet capture. A survey of commercial specialized capturing hardware can be found in Section 3.3.4. Related work on AJAX can be found in the prelude of Chapter 4. For background information on Online Social Networks please read the beginning of Chapter 5 and related work in Section 5.7, for mobile hand-held devices see Section 6.3. Finally, find additional information on NNTP in the preface of Chapter 7.

2.1 Internet traffic characteristics

We start by summarizing previous studies on how Internet traffic looks like. We consider four aspects: *(i)* The composition of the application mix, *(ii)* popular content-types, *(iii)* the distribution of traffic over the course of a day, and *(iv)* the distribution of connection sizes.

2.1.1 Application mix

One constant in the Internet over the last 10 years has been its steady growth by more than 50 % each year [36, 72]. Initially, protocols such as FTP, SMTP, and NNTP were popular. Then around 1994, HTTP entered into the picture. Until 2000, P2P protocols such as Napster and Gnutella became popular but were later overtaken by eDonkey and BitTorrent. However, the traffic mix has undergone substantial changes. Therefore, we now revisit previously reported results regarding the application mix of Internet traffic. For this purpose we rely on various studies that report on the application mix between 2007 and 2009 from different vantage points:

- The study by Maier et al. [60], which is based on a subset of the traces studied in this thesis. It was presented at IMC '09.
- Two studies by ipoque [92, 93], which report on different regions in the world (Germany and Middle East). These studies are available for download after registration via a Web form.
- The Arbor report [51] on the ATLAS Internet Observatory presented at a recent NANOG[1] meeting.
- The Sandvine report on "Global Broadband Phenomena" [85].

In order to compare the results we have to summarize and unify the traffic categories as each study uses their own nomenclature (see Figure 2.1). For this purpose we use the following seven categories:

Web All HTTP traffic including One-Click-Hosters (OCHs or Direct Download Providers) but excluding video and audio streaming over HTTP (i. e., Flash-Video).

Streaming All types of streaming in the Internet including streaming over HTTP, RTP, RTSP, RTMP, ShoutCast, etc.

Usenet The article reading and posting system that evolved from UUnet and which uses NNTP as protocol.

BitTorrent/P2P The popular P2P-protocol BitTorrent and all other P2P traffic that is not eDonkey. Note, that the P2P traffic that is not BitTorrent or eDonkey only adds a tiny fraction. Moreover, this category represents all P2P traffic if the study no further subdivides P2P traffic. This is the case for Arbor [51] and Sandvine [85]. Note as well, that the Arbor study [51] reports a table with traffic shares, stating 0.95 % for P2P. This table is annotated with the comment that P2P is more likely to account for 18 % based on payload inspection of a limited data subset.

eDonkey Another P2P protocol, if reported.

Other/known Other identified traffic, for details we refer to the corresponding studies.

Unclassified Traffic that has not been classified. Note, that the Sandvine [85] study does not mention unclassified traffic, which either implies a minute fraction or that it is missing in the plot.

Looking at these statistics we find that all studies report a significant fraction of Web traffic. Indeed, Web is dominant ($> 50\,\%$) in most studies, followed by P2P and streaming. It is noteworthy that Usenet is responsible for a non-negligible fraction

[1] NANOG is the North American Network Operators Group.

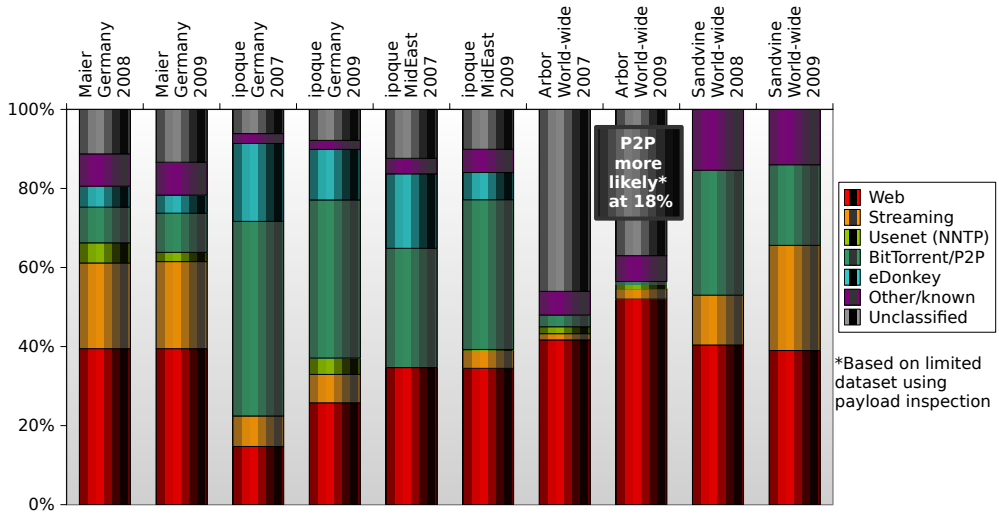


Figure 2.1: **Barplot of the application mix in the Internet (unified categories) for different years, different regions according to several sources [51, 60, 85, 92, 93].** (BitTorrent/P2P contains all P2P except eDonkey.)

in several studies. This is surprising and a good example for the importance of revisiting the application mix periodically in order to identify new trends.

In terms of P2P protocol distribution Figure 2.1 shows that BitTorrent is dominating and the shares of eDonkey are decreasing. Thus, we note that the results of Plissonneau et al. [81] who observed 91 % of the P2P traffic is due to eDonkey in 2004 are no longer applicable. Indeed, the popularity among P2P protocols swapped in favor of BitTorrent. We can also see a general trend: P2P is declining according to all studies. This is also supported by the results of Anderson [6]. He points out that this decline comes with an increase in video streaming. Moreover, most of the studies pointed out that currently One-Click-Hoster (e. g., Rapidshare or MegaUpload) are as important for file-sharing as P2P systems.

Of course there are also trends that do not impact the application mix, for example Online Social Networks (OSNs) such as Facebook. This is due to the fact that OSNs use HTTP and they do not transport large videos, but profile elements. Nevertheless, OSNs remain important, given the huge number of OSN users world-wide.

2.1.2 Content-types in the Internet

Next, we turn to the popularity of content-types in the Internet. Again, we leverage several data sources, namely Maier et al. [60], ipoque [93], and Erman et al. [28]. Here we unify the categories and present results for contents transfered via BitTorrent, eDonkey, and HTTP. See Figure 2.2 for a summary.

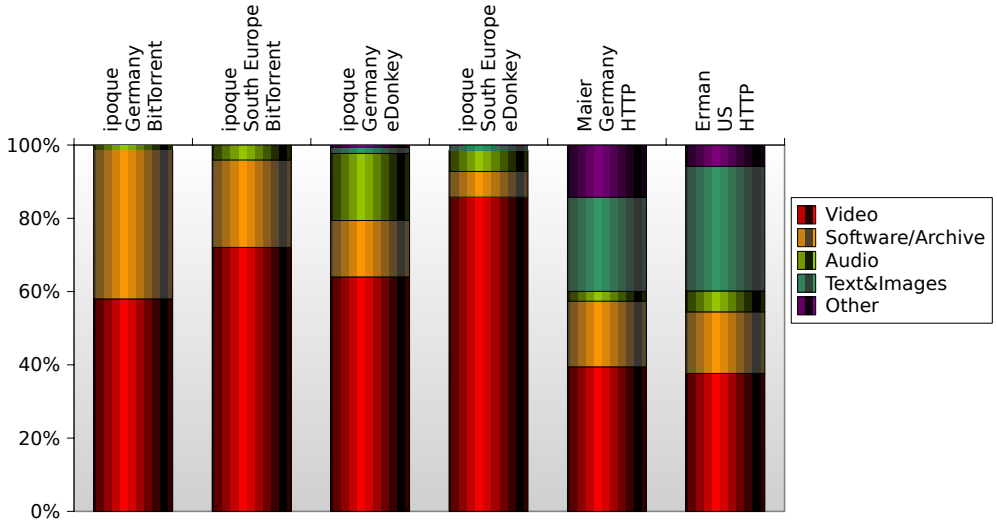


Figure 2.2: **Barplot of content-type popularity in the Internet (unified categories) for different protocols, different regions according to several sources [28, 60, 93].**

We see that videos are the most popular content in P2P systems (BitTorrent and eDonkey). Even in HTTP, videos account for more traffic than any other category. Although HTTP was designed to transfer Webpages (text, e. g., HTML, XML, CSS, JavaScript, and image files) these contribute less than a third of the total HTTP volume.

Overall, a significant fraction of software and data archives is noticeable. According to Maier et al. [60] almost all videos are in flash-video format and are served by video portals such as YouTube. Similarly, almost all archives are served by One-Click-Hosters. This is confirmed by the results of Erman et al. [28].

Shifts in the popularity of content-types can be another indicator of new trends. For example, there were almost no flash-videos observed before the breakthrough of YouTube.

2.1.3 Time-of-day effects

In order to understand when people are active in the Internet we show time-of-day usage plots of link utilization from Maier et al. [60] in Figure 2.3, and aggregated traffic volume from ipoque [92] and Sandvine [85] in Figure 2.4 and Figure 2.5, respectively.

In general, we observe a peak utilization at prime-time around 8 pm and a daily low between 2 am and 4 am. As the data sets of all these studies are primarily collected

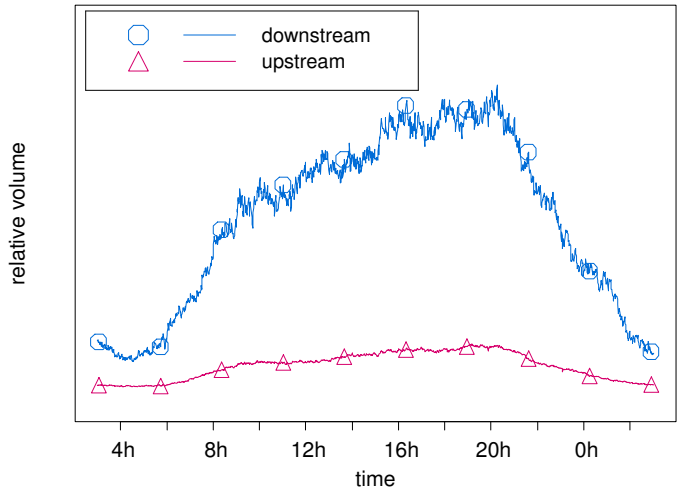


Figure 2.3: **Timeseries of link utilization from Maier et al. [60]**

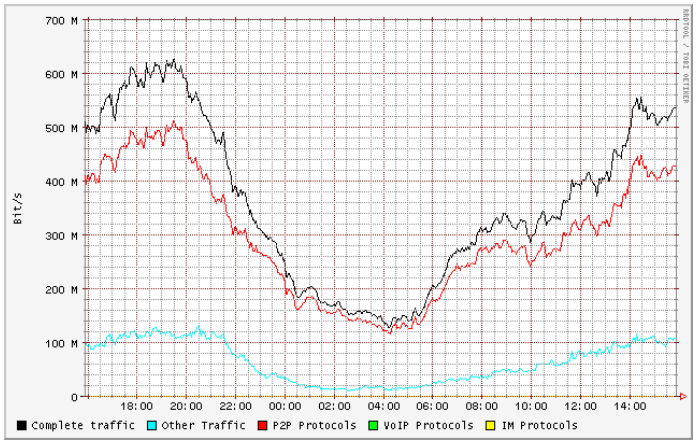


Figure 2.4: **Timeseries of traffic volume from ipoque [92]**

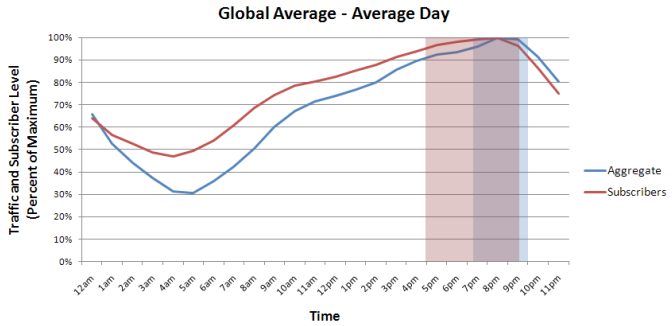


Figure 2.5: **Timeseries of traffic volume from Sandvine [85]**

from residential networks, it not surprising that they all show similar characteristics. The peak usage in the evening hours can easily be explained by the fact that people are usually not at home during business hours. Raising demands just before lunch and in the afternoon may be due to children returning home from school.

2.1.4 Flow sizes and durations

Finally, we focus on flow characteristics. A flow in this context is the set of packets with the same 5-tuple of source and destination IP, source and destination port, and transport protocol.

A reoccurring observation is that the flow size distribution is consistent with a heavy-tailed distribution [21, 55, 80, 116]. Recent work by Basher et al. [11] compared flow-level characteristics of Web and P2P traffic. They found a mean (median) flow size of 21.5 kB (2.53 kB) for Web and 362.4 kB (1.17 kb) for P2P. This is also reflected in their CCDF [11, Figure 2(a)], where more heavier flows are shown for P2P. Calculating the flow size distribution of all traffic for a data set collected in February 2008 at ISP-A[2] we find a mean flow size of 38 kB and a median of 478 Bytes. Figure 2.6 shows the CCDF of these distributions and an exponential distribution with the same mean.

Basher et al. [11, Table 5] further find that flow durations have a mean (median) of 13 seconds (400 ms) for Web and 123 seconds (24 seconds) for P2P. Again, we calculate statistics on the data used earlier for the flow sizes (ISP-A in February 2008), see Figure 2.7: Mean flow duration is 9 seconds and median is 1.1 seconds. Both results are in the same order of magnitude. The differences are likely to be caused by the different composition of the traffic that is covered: Web and P2P on the one hand and complete traffic on the other.

2.2 Traffic analysis tools

Now, we briefly introduce the traffic analysis tools that we use. All tools are Open-Source and freely available.

2.2.1 libpcap and tcpdump

LIBPCAP [42] is a C library for capturing packets, that are transmitted over a link that is attached to the computer running the pcap application. The functions included in LIBPCAP provide a standardized interface for all common (UNIX-based) operating systems, including Linux and FreeBSD. The interface of LIBPCAP can also

[2] See Section 5.2 for details on the monitoring point.

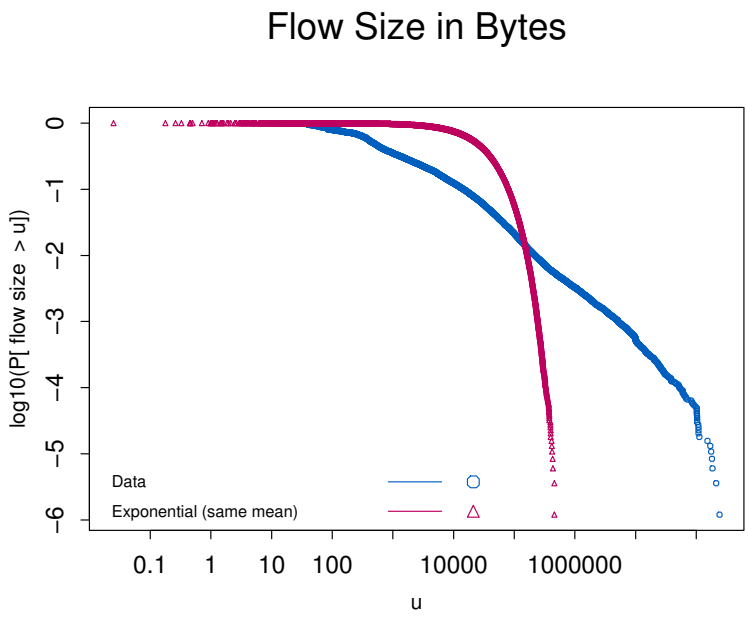


Figure 2.6: **CCDF of flow sizes: Data and exponential distribution with same mean**

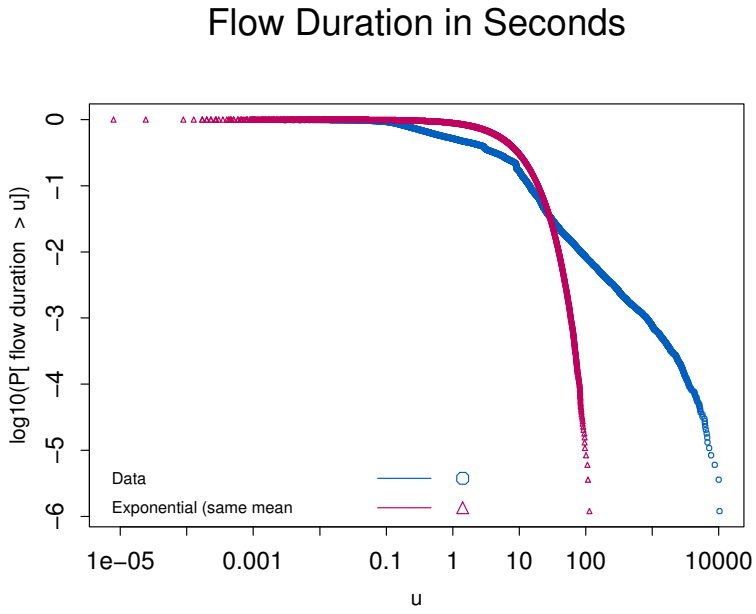


Figure 2.7: **CCDF of flow durations: Data and exponential distribution with same mean**

be used under Windows but the Windows library is called WINPCAP. Many network analysis tools are build on top of LIBPCAP to allow for use on any LIBPCAP supported architecture and operating system.

LIBPCAP defines a packet trace file format and includes functions for reading from such trace files as well as for dumping packets to disk. Therefore LIBPCAP enables both live capture from a network interface and offline analysis from a saved trace file. Furthermore, the pcap trace file format is the de-facto standard for exchanging packet level traces.

One of the most popular tools for network monitoring and network debugging is TCPDUMP (`www.tcpdump.org`). The command-line tool TCPDUMP uses LIBPCAP for data capture. TCPDUMP itself includes various analyzers for parsing network protocols and producing human readable output summaries per packet. It can also be used to collect traces in pcap format.

2.2.2 Wireshark

WIRESHARK [117] is a graphical interface for LIBPCAP with similar capabilities as TCPDUMP. Due to WIRESHARK's information sorting and filtering options it is a very intuitive tool for network debugging. It can read and write pcap format as well as capture data directly from the network. WIRESHARK started as "Ethereal" and was renamed later due to trademark issues.

WIRESHARK's analysis capabilities exceed those of TCPDUMP. For example WIRESHARK can reassemble TCP connections from packet traces and parse the application layer protocols. WIRESHARK also ships some statistical traffic analysis tools, such as conversation lists, response-time analyses, or HTTP request summaries to name a few.

To take advantage of WIRESHARK's extended analysis capabilities via a command-line interface the distribution also includes TSHARK. Moreover, it offers a programmable interface for automated trace editing, converting, or analysis.

2.2.3 Bro

BRO [79] is a network intrusion detection system (NIDS) developed by Vern Paxson [78]. Before attacks can be detected by customizable policy scripts BRO first needs to parse and generate events from the observed network traffic. In order to do so it features a robust TCP reassembly engine and protocol analyzers. These analyzers not only match signatures on the traffic, but try to understand and reconstruct the semantics of the application layer protocol. BRO is organized in several layers (see Figure 2.8, top to bottom):

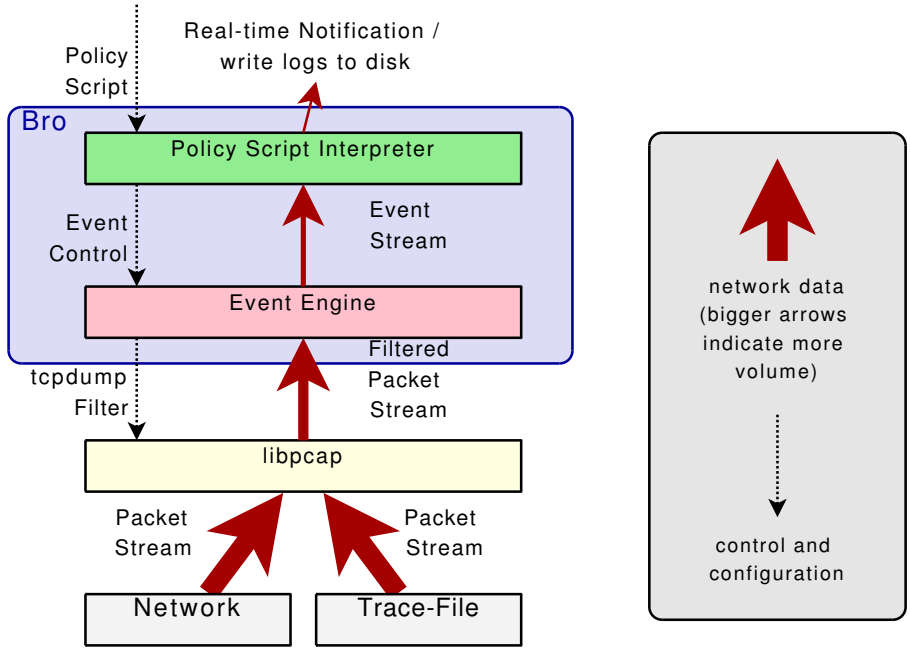


Figure 2.8: **Structural design of Bro**

- The *policy script interpreter* loads the configuration via policy script files. After parsing the files it instructs the event engine which events it is interested in and which analyzers to load. Once events arrive the interpreter executes the analysis as specified in the policy scripts.
- Depending on the events that the policy layer subscribes to and depending on the analyzers that are loaded the *event engine* compiles the capturing filter and starts to capture packets via LIBPCAP. It parses a packet, determines the connection it belongs to, and hands it to the appropriate analyzer tree for this connection. In case of TCP the first packet instantiates a TCP-reassembly for this connection. Then depending on the TCP payload the correct protocol analyzer is started. The analyzer generates application layer events as they are detected and notifies the policy layer.
- LIBPCAP handles all interactions with the operating system and is used as a uniform interface for packet capturing. Via LIBPCAP BRO can read the network data either live from the network or from a pcap trace file.

The analyzer tree is dynamic and can have multiple different protocol analyzers examine the same connection. If an analyzer determines that the current data stream does not comply with the semantics of the protocol that it expects, it can detach itself from the tree with a notification. This dynamic protocol detection [25] allows the detection of traffic of certain protocols even if they do not run on default ports, do not match a signature, or are tunneled within another protocol.

Although Bro was designed as an NIDS, its protocol analysis capabilities make it a perfect tool for traffic classification and application layer analysis. Thus, we use it extensively for all the analyses presented in this thesis. Bro ships a whole set of protocol analyzers including an HTTP analyzer that is capable of extracting HTTP requests, responses, and if required, also HTTP headers. For a complete list of available analyzers please refer to `www.bro-ids.org`.

Chapter 3

Infrastructure for Network Data Capture and Analysis

A crucial component of almost any network measurement and especially any network security system is the one that captures and analyzes the network traffic. Nowadays, almost all organizations secure their incoming/outgoing Internet connections with a security system. As the speed of these Internet connections increases from T3 to 100-Megabit, or even 10-Gigabit Ethernet, the demands on the monitoring system increase as well while the price pressure remains. For example, the Münchner Wissenschaftsnetz (MWN, Munich Scientific Network) [54] offers Internet access to roughly 50,000 hosts via a 10-Gigabit bidirectional uplink to the Deutsche Forschungsnetz (DFN, German Scientific Network). Today, MWN's operators face the challenge of performing packet capture in a 10-Gigabit Ethernet environment using commodity hardware in order to run a security system. We note that most network security systems rely on capturing *all* packets, as any lost packet might cause failing to reveal an attack. Furthermore, most attack detection mechanisms rely on analyzing the packet content, and the security system itself must be resilient against attacks [78]. Therefore, packet capture must be able to handle both the maximum data rates as well as the maximum packet rates.

In Section 3.1 we explain the limitations of packet capturing systems. Section 3.2 presents our results indicating how much traffic a single system is capable of capturing[1]. Next, we present some approaches to overcome the limitations in Section 3.3 and finally propose an infrastructure for data capture in Section 3.4.

3.1 Challenges and limitations of packet capturing systems

Network traffic capture is not an easy task due to the inherent system limitations (memory and system bus throughput) of current off-the-shelf hardware. Expensive alternatives[2] exist as specialized monitoring cards. Our experience shows that PCs

[1] Please note that most of the performance analysis was done two years ago and that the performance of today's systems may exceed our reported numbers. Nevertheless, network load has also continued to increase, e. g., requiring upgrades from 10-Gigabit links to 40-Gigabit links.

[2] Current cost for a 1-Gigabit card is roughly 5,000 €. The costs for 10-Gigabit cards are well beyond 20,000 €.

equipped with such cards are able to capture full packet traces to disk in 1-Gigabit environments. But even these systems reach their limits in 10-Gigabit environments: They lack CPU cycles to perform the desired analysis and/or bus bandwidth to transfer the data to disk.

We approach the question, which system is able to monitor a Gigabit Ethernet interface given the many factors that impact the performance of a packet capture system. These factors include the system, CPU, disk and interrupt handling architecture [66], the operating system and its parameters, the architecture of the packet capture library [22, 23, 42, 120], and the application processing the data [42, 78, 96]. In this chapter, we focus on a first set of factors and only consider simple but typical application level processing, such as compressing the data and storing it to disk. We choose to examine three high-end off-the-shelf hardware architectures (all dual processor): Intel Xeon, AMD Opteron single core, and AMD Opteron multi-core based systems; two operating systems: FreeBSD and Linux with different parameter settings; and the packet capture library, LIBPCAP [42], with different optimizations.

In 1997, Mogul et al. [66] highlighted the problem of receive livelock. Since then, quite a number of approaches have been proposed to circumvent this problem [22, 23, 83, 84]. Most of these approaches rely on reducing the work that needs to be done per captured packet, be it reducing the amount of generated interrupts or saving a copy operation to move the packet towards the analyzing application. For a detailed discussion of the related work we refer the reader to [86, Section 1.2 and 2.2]. Yet, the capabilities and limitations of the current capturing systems have not been examined in the recent past.

3.1.1 Bandwidth limitations

Current PC architectures suffer from two fundamental bottlenecks: Bus bandwidth and disk throughput. In order to capture the data of a fully utilized bidirectional 10-Gigabit link, a system throughput of 2560 Mbytes/s is required. Even for capturing only the packet headers, e. g., the first 68 bytes, one needs 270 Mbytes/s given an observed average packet size of 645 bytes. Moreover when writing to disk, packet capture libraries require the data to pass the system bus twice: Once to copy the data from the card to the memory to make it accessible to the CPU, and once from memory to disk. Thus, this requires twice the bandwidth.

When comparing these bandwidth needs with the bandwidth that current busses offer we notice a huge gap—especially for full packet capture. The standard 32-bit, 66 MHz PCI bus offers 264 Mbytes/s. The 64-bit, 133 MHz PCI-X bus offers 1,066 Mbytes/s. The PCIexpress x1 bus offers 250 Mbytes/s. It is easy to see that none of these busses is able to handle the load imposed by a *single* uni-directional 10-Gigabit link (full packets). Furthermore, the numbers have to be taken with a grain of salt as they are maximum transfer rates, which in case of PCI and

PCI-X are shared between the attached PCI cards. Some relief is in sight: PCI-express x16 (or x32) busses which offer 4000 Mbytes/s (8000 Mbytes/s) are showing a gaining importance.

Next, we survey the bandwidth of the disk systems. The fastest ATA/ATAPI interfaces operates at a mere 133 Mbytes/s; S-ATA (2) offers throughputs of up to 300 Mbytes/s; SCSI (320) can achieve 320 Mbytes/s. Even the throughput of Fiberchannel, up to 512 Mbytes/s, and Serial-Attached-SCSI (SAS), up to 384 Mbytes/s, is not sufficient for 10-Gigabit links. The newest SAS 2 systems (available since 2009) are planned to offer 768 Mbytes/s while SAS 3 (expected 2013) systems may eventually offer 1536 Mbytes/s. Again, there is a huge gap.

Although the described systems and components are two years old, they still represent state-of-the-art components. PCIexpress is commonly available now. Furthermore, there are follow up specifications for PCIexpress 2.0 and 3.0 each doubling the bandwidth. However, network speeds also increased: Backbone links are operated on 40-Gigabit and 100-Gigabit is approaching fast. Thus, the challenge remains that PC component speeds do not scale up to network link speeds.

3.1.2 Performance impacting factors

Let us now turn attention to the selection of suitable commodity hard- and software for the task of packet capturing. For this purpose, we identify the principal factors that may impact the performance and then examine the most critical ones in detail.

Obvious factors include the network card, the memory speed, and the CPU cycle rate. A slightly less obvious one is the system architecture: Does it matter if the system has multiple CPUs, multiple cores, or if hyper-threading is enabled? Another important question is how the OS interacts with the network card and passes the captured packets from the card to the application. How much data needs to be copied in which fashion? How many interrupts are generated and how are they handled? How much buffer is available to avoid packet loss? Yet another question is whether a 64-bit OS offers an advantage over the 32-bit version. With regards to the application we ask by how much the performance is degraded when running multiple capturing applications on the same network card, when adding packet filters, when touching the data in user space, i. e., via a copy operation or data compression, and when saving the data to a trace file on disk. These are the questions that we try to answer in the remainder of the chapter.

3.2 Performance evaluation of packet capturing systems

Next, we want to assess how much data, both in terms of packets and bytes, can be reliably captured by a single machine. Conducting such measurements may appear simple at first glance, but the major difficulty is to find hard-/software configurations that are comparable, as we want to avoid comparing apples to oranges.

3.2.1 Systems under test

To enable a fair comparison, state-of-the-art off-the-shelf computer hardware for all four systems has been purchased at the same time in February 2004, with comparable components. Preliminary experiments have shown that the Intel Gigabit Ethernet cards provide superior results than, e. g., those of Netgear. Furthermore, 2 Gbytes of RAM suffice and we choose to use the fastest available memories. Accordingly, we focus on the system architecture and the OS while using the same system boards, the same amount of memory, and the disk system. Consequently, two AMD Opteron and two Intel Xeon systems[3] have been purchased that are equipped with the same optical network card, the same amount of memory, and the same disk system (IDE ATA based). One Opteron and one Xeon system have been installed with FreeBSD 5.4 and the others have been installed with Debian Linux (Kernel v2.6.11.x).

Once dual-core systems became available, two additional machines had been purchased in May 2006: HP Proliant DL385[4]. In contrast to the other systems these have an internal SCSI controller with three SCSI disk attached. One machine has been installed with FreeBSD 6.1 and the other one with Debian Linux (Kernel v2.6.16.16), both with dual-boot for 32-bit and 64-bit. All systems are equipped with two processors[5] and are capable of symmetric multi-processing (SMP).

3.2.2 Measurement topology

To be able to test multiple systems at the same time and under exactly the same workload we choose the setup shown in Figure 3.1. A workload generator, see Section 3.2.4, generates traffic which is fed into a passive optical splitter. The task of the splitter is to duplicate the traffic such that all systems under test (SUT) receive the same input. The switch between the workload generator and the splitter is used to check the number of generated packets while all capture applications running on the SUTs are responsible for reporting their respective capture rate. In

[3] System details: AMD Opteron 244, Intel Xeon 3.06Ghz, 2 Gbytes RAM (DDR-1 333 MHz), Intel 82544EI Gigabit (Fiber) Ethernet controller, 3ware Inc. 7000 series ATA-100 Storage RAID controller with at least 450 Gbytes space.

[4] Details see previous footnote except of processors (AMD Opteron 277), disk system (HP Smart Array 64xx controller with 3×300 Gbytes U320 SCSI hard disks), and faster memory.

[5] The newer dual-core systems are equipped with four cores in total.

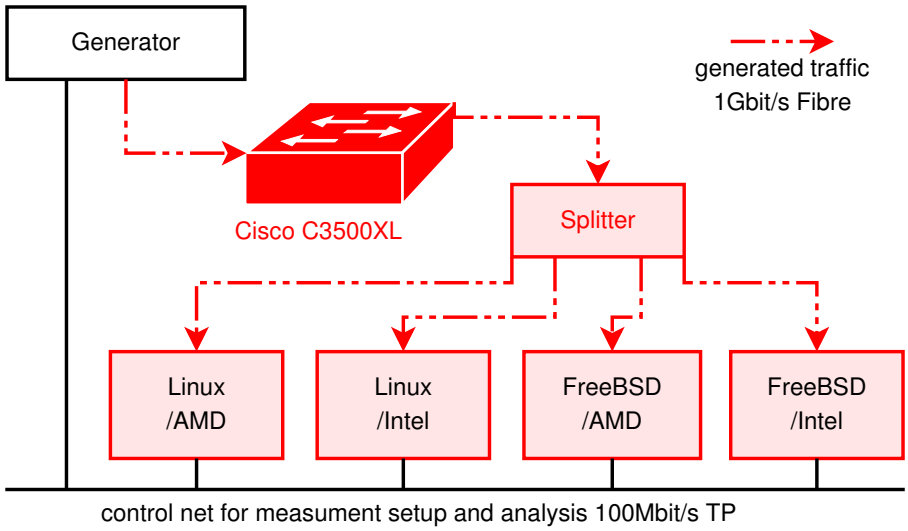


Figure 3.1: **Layout of measurement topology**

later experiments we removed the switch as the statistics reported by the traffic generator itself proved to be sufficient. The control network is a separate network, which is used to start/end the experiments and to collect the statistics.

3.2.3 Experiment sequence

Each measurement varies the traffic rate from 50 Mbit/s to 920 Mbit/s[6] and consists of seven repetitions of sending one million packets at each bandwidth setting to reduce statistical variations. In each iteration we start the capture process (using a simple LIBPCAP [42] based application [86] or TCPDUMP) and a CPU usage profiling application on the SUTs. Then, we start the traffic generator. Once all packets of the run have been sent by the workload generator we stop the packet capture processes.

For each run we determine the capture rate by counting the received number of packets and the average CPU usage (see `cpusage` [86]). The results are then used to determine the mean capturing rate as well as the mean CPU usage for each data rate.

3.2.4 Workload

With regards to the generated traffic, we decided neither to focus on the simplest case for packet monitoring (all large packets) nor on the most difficult one (all small

[6] 920Mbit/s is close to the maximum achievable theoretical input load given the per packet header overheads.

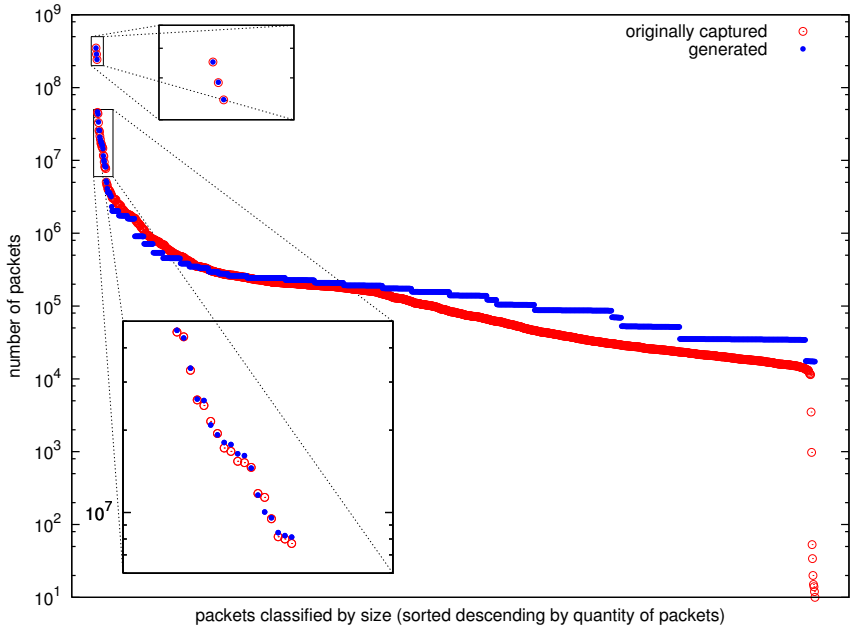


Figure 3.2: **Frequency of packet sizes (sorted by rank, y-axis in logscale)**

packets) since both are unrealistic. The first case puts too much weight on the system bus performance while the latter emphasizes the interrupt load too much.

Instead, we decided to use a traffic generator that is able to generate a packet size distribution that closely resembles those observed in actual high-speed access networks. For this purpose, we enhanced the Linux kernel packet generator (LKPG [73, 74]), which is capable of filling a 1-Gigabit link, to generate packets according to a given packet size distribution.

Figure 3.2 shows the input and the generated packet size distributions of our modified LKPG based on a 24h packet level trace captured at the 1-Gigabit uplink of the MWN [54] (peaks are at 40–64, 576 and 1420–1500 B).

We decided not to mimic flow size distribution, application layer protocol mix, delay and jitter, or data content, as they have no direct impact on the performance of the capture system. Their impact is on the application which is beyond the scope of this chapter. The same holds for throughput bursts, which can be buffered. The intention of the work is to identify the maximum throughput a capturing system can handle. We realize different traffic rates by inserting appropriate inter-packet gaps between the generated packets.

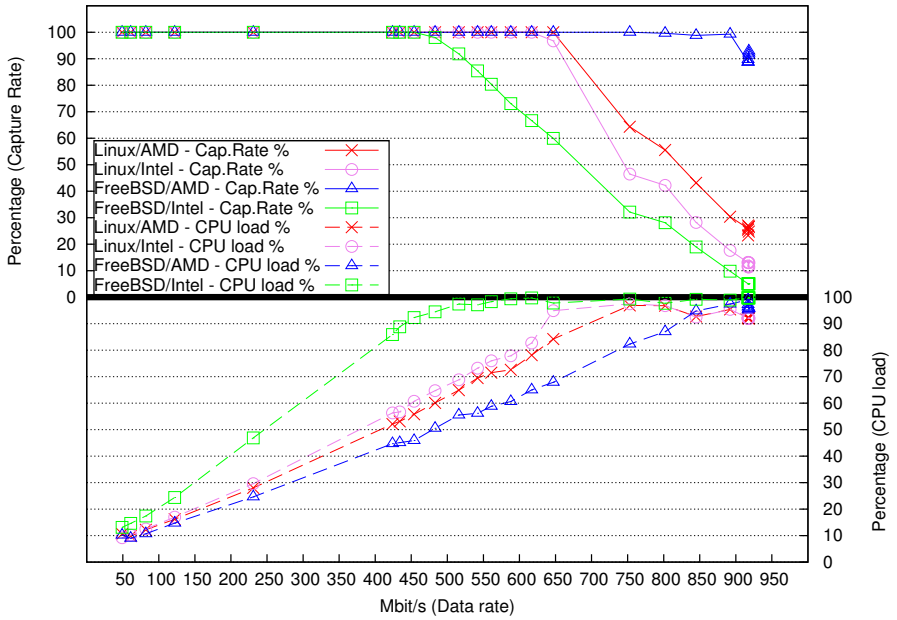


Figure 3.3: **Capturing performance of single processor & increased buffers**

3.2.5 Results

Based on our measurement setup, we now evaluate the performance of our systems starting from the vanilla kernels. We find that it is crucial to increase the amount of memory that is available to the kernel capture engine. Otherwise scheduling delays, e. g., to the capturing application, can cause the overflow of any of the kernel queues, e. g., the network queue for the capture socket. This is achieved by either setting the `debug.bpf_bufsize` sysctl parameter (FreeBSD) or the `/proc/sys/net/core/rmem*` parameter (Linux). Based upon our experience we choose to use buffers of 20 Mbytes. Furthermore, we notice that the systems spend most of their cycles on interrupt processing when capturing at high packet rates. To overcome this problem we try device polling as suggested by Mogul et al. [66]. For Linux this reduces the CPU cycles that are spent in kernel mode and thus increases the packet capture rate significantly. For FreeBSD activating the polling mechanism slightly reduced the capturing performance and the stability of the system. Therefore, we use it for Linux but not for FreeBSD.

Next, we baseline the impact of the system architecture and the OS by comparing the performance of the four systems with single processor kernels. Figure 3.3 (top) shows the capture rate while Figure 3.3 (bottom) shows the CPU usage as we increase the data rate. To keep the plots simple we decide against including the standard deviation. Note, that all measurements have a standard deviation of less than 2 %. As expected, the capture rate decreases while the CPU usage increases

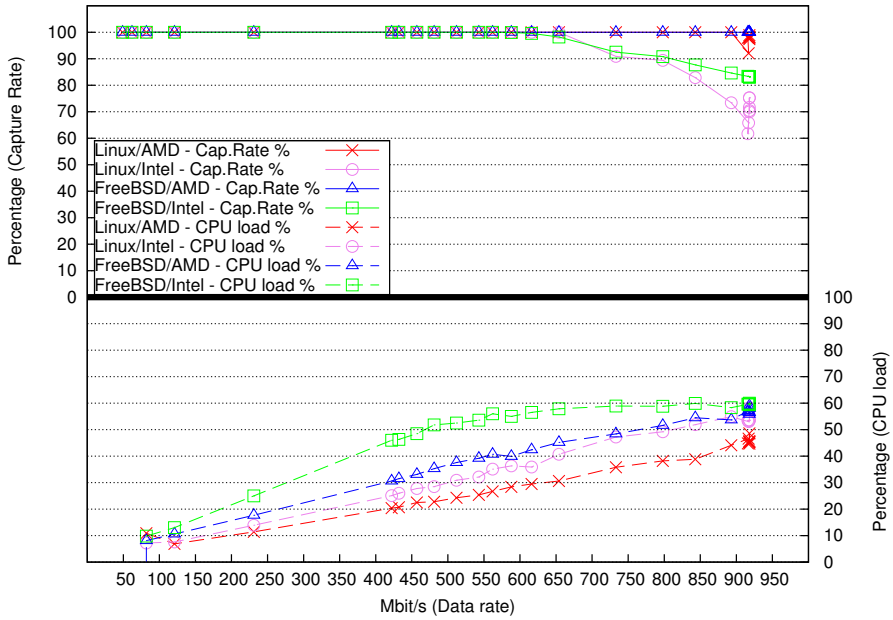


Figure 3.4: **Capturing performance of multiple processors & increased buffers** (50 % CPU load implies one fully utilized CPU.)

as the data rate is increased. FreeBSD/AMD is the only combination which loses almost no packets. All others experience significant packet drops. We note that the systems start dropping packets once their CPU utilization reaches their respective limits. This indicates that the drops are not due to missing kernel buffers but are indeed due to missing CPU cycles. The FreeBSD/Intel combination already starts dropping packets at about 500 Mbit/s. Neither of the Linux systems looses packets until roughly 650 Mbit/s. From that point onward their performance deteriorates dramatically with increasing data rates. The efficiency of the FreeBSD/AMD machine is especially surprising as FreeBSD performs an additional (kernel) packet copy operation and does not use device polling which proved beneficial on Linux systems.

This observation indicates that utilizing the multi-processor system may help overcome the aforementioned problems as the kernel can be scheduled on one processor while the application is scheduled on the other. This almost doubles the CPU resources. Yet, memory conflicts and cache misses have the potential to deteriorate the capture performance. The results shown in Figure 3.4 show the resulting significant performance increase. This indicates that the additional CPU cycles of the SMP architecture clearly top their penalties, e. g., cache misses. In fact, the FreeBSD/AMD combination is able to capture all packets even at the highest data rate. Overall, we note that in our experience FreeBSD outperforms Linux. To test if further increasing the multiprocessor capabilities fosters the performance we enabled Hyper-Threading Technology (only available on Intel machines). This does not im-

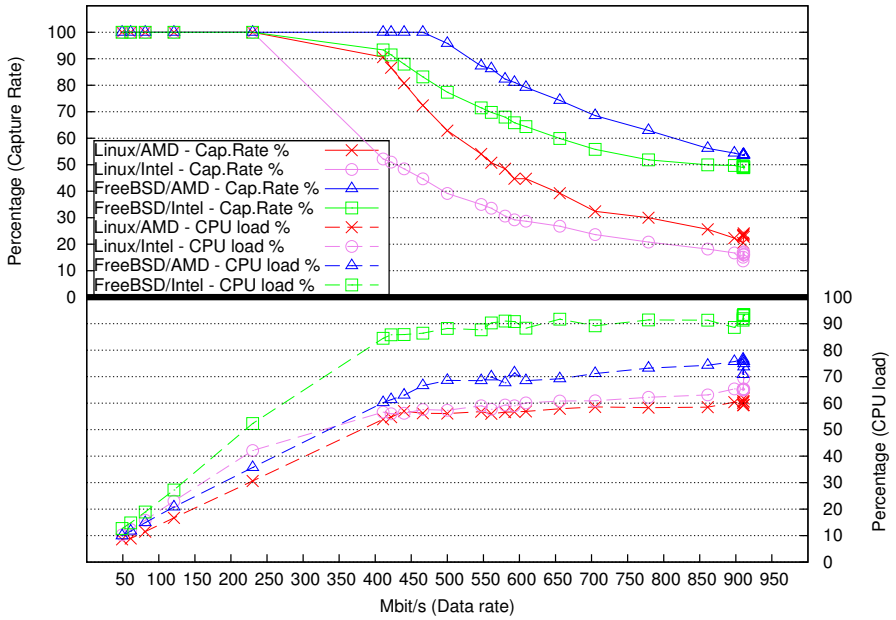


Figure 3.5: **Capturing performance of multiple processors, increased buffers & 50 additional `memcpy` operations on the packet data** (50 % CPU load implies one fully utilized CPU.)

pact the performance, but it is important to keep in mind that in this test only two threads of execution require significant CPU resources, the kernel and the capture application.

As it is common to use filters while capturing, we studied the impact of configuring a 50 BFP instructions filter. We find that even if packets have to pass a long filter before being accepted, the performance is not drastically altered. The performance of the FreeBSD machines remains as is, while that of the Linux machines decreases slightly (up to 10 %) at high data rates (>800 Mbits/s). This indicates that at least for FreeBSD, filtering does not impose high costs on CPU resources. But what if we run multiple capture applications at the same time? In this case the packets have to be duplicated within the kernel and delivered to multiple filter instances. Not surprisingly, we find that the performance deteriorates for all four systems. Since the Linux machines begin to massively drop packets when reaching their CPU limits, we suggest using FreeBSD as the drop rates are less significant.

The next challenge that we add to the application are memory copy operations as they are common in network security applications. To investigate an extreme position, we instructed the application to copy the captured packet 50 times within user-space. In Figure 3.5 we see that all systems suffer under the additional load. The Linux machines yet again experience larger problems. The next challenge is to compress the content of each packet using a standard compression routine, LIBZ.

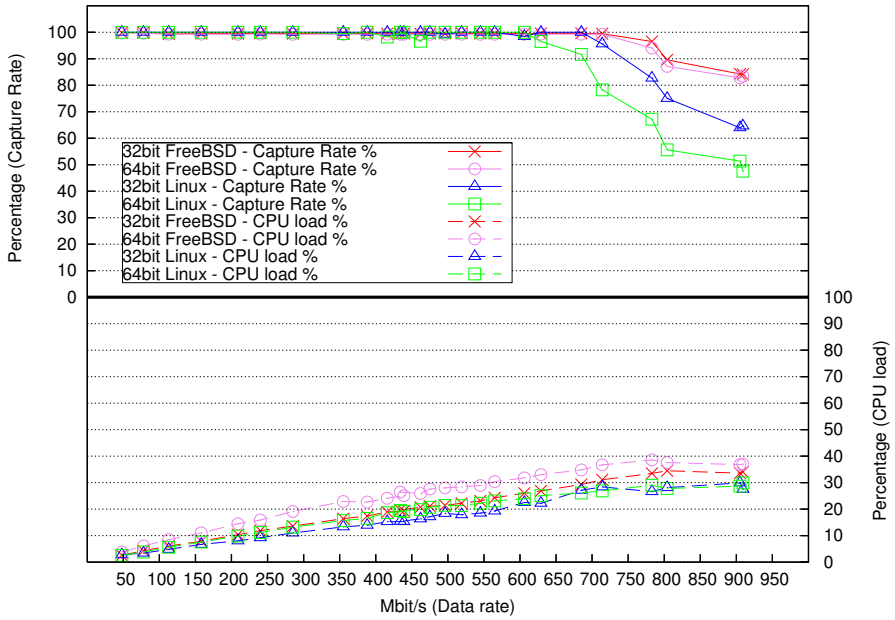


Figure 3.6: **Capturing performance of multiple *dual-core* processors, increased buffers & writing full packets to disk** (25 % CPU load implies one fully utilized CPU.)

For the first time the Intel systems outperform their AMD counterparts. This suggests that Intel processors and not AMD's have better hardware realizations for instructions used by LIBZ.

Recapitulating, we find that the multiprocessor Opteron system with FreeBSD 5.4 outperforms all other systems. Its maximum loss rate is less than 0.5 %. One explanation is that AMD has a superior memory management and bus contention handling mechanism. These results motivate us to only purchase Opteron dual-core machines for the next tests. To baseline the new system we repeat the above experiments, except the ones with additional application load. These experiments show that both machines can capture every single packet on FreeBSD as well as Linux running either the 32-bit or the 64-bit version of the respective OS.

The newer machines clearly outperform the older ones. Therefore, we next examine if either system is able to capture full packet traces to disk. For this we switch to using TCPDUMP as application. From Figure 3.6 we see that Linux is able to write all packets to disk up to a data rate of about 600 Mbit/s independently of using a 32-bit or 64-bit kernel. FreeBSD consistently drops roughly 0.1 % of the packets even at the lowest data rates. This indicates a minor but fixable principle problem in the OS. FreeBSD only begins to drop a significant number of packets when the data rate exceeds 800 Mbit/s. Keep in mind that Linux captures only about 65 % of the packets at the highest data rate (32-bit). While under FreeBSD

the difference between 32-bit and 64-bit mode is negligible (up to a capture rate of 83 %), Linux in 64-bit mode deteriorates drastically. It records only half of the generated packets. The poor performance of 64-bit Linux might be due to the increased memory consumption for longer pointers within the kernel.

3.2.6 Summary

To answer the question which system is able to support packet monitoring best we present a methodology for evaluating the performance impact of various system components. We find that our AMD Opteron systems outperform our Intel Xeon ones and that FreeBSD 6.1 outperforms Linux 2.6.16. While multi-processor systems offer a lot, the benefit of adding hyper-threading and multi-core is small. Moreover, it appears that for the task of packet capture the 64-bit OS versions are not quite ready yet. The newer systems clearly outperform the older ones and can even capture full packet traces to disk as long as the data rate is less than 600 to 700 Mbit/s. All in all, multi-processor Opteron systems with FreeBSD clearly outperform all other systems.

Obviously, our goal must be to understand not only the packet capture performance but also the characteristics of highly complex security screening applications such as Bro. A summary of our current and future results is available on the project Website [91].

3.3 Overcoming limitations of packet capturing systems

In this section we present and propose different approaches for overcoming the limitations of packet capturing systems. Among the proposals are splitting the capturing load across multiple machines (see Section 3.3.1) or multiple processors (see Section 3.3.2), as well as using specially developed network adapter drivers that are optimized for packet capture (see Section 3.3.3). In Section 3.3.4 we survey specialized packet capturing hardware. Also note the findings in Section 3.2.5 that tuning specific operating system parameters can significantly improve capturing performance.

3.3.1 Using link trunking to split traffic over multiple links

One way to overcome the performance limitations of current commodity hardware for packet capturing is to distribute the load across multiple machines. Instead of designing custom hardware for this purpose we propose to use a feature of current Ethernet switches. Most switches are capable of bundling a number of lower speed switchports. Once the bundle has been configured it can be used like a normal

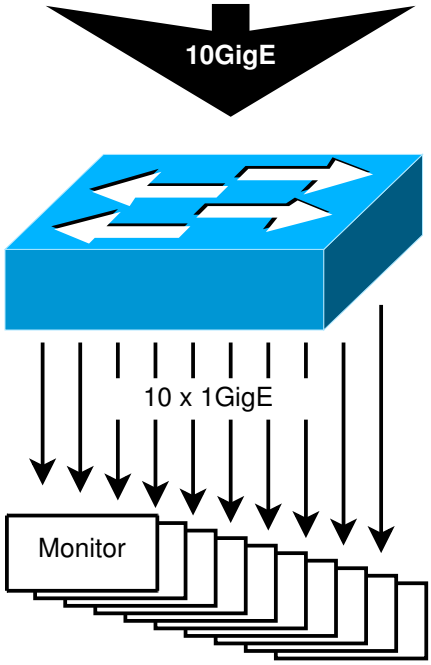


Figure 3.7: **Example of a setup using port bundling to split traffic across monitors**

interface. Therefore, it can be used as a monitor interface for any other interface. Accordingly, the switch forwards all traffic that it receives on a high bandwidth interface, e. g., 10-Gigabit, on a bundle of low bandwidth interfaces, e. g., 1-Gigabit. These lower speed interfaces can then be monitored individually. Figure 3.7 shows a schematic view of this setup. There are three major drawbacks to this solution: *(i)* Time stamps are slightly distorted as the packets have to pass through an additional switch, *(ii)* it is hard to synchronize the system time of the monitors, and *(iii)* even though the overall capacity suffices the individual capacity of each link may not be sufficient. This depends on the load balancing scheme in use.

We tested this mechanism by using a Cisco 3750 switch to move from monitoring a single 1-Gigabit Ethernet link to eight (which is maximum for this vendor) 100-Mbit links. We use the EtherChannel feature and configure such a link bundle across eight 100-Mbit links. Then this link is used as a monitor link for a 1-Gigabit input link. It is crucial to use the appropriate load balancing method for this link. Common options include load balancing based on MAC addresses (simple switches), IP addresses and/or MAC addresses (e. g., Cisco 3750), combinations of IP addresses and/or (TCP/UDP) port numbers (Cisco 6500 Series). Note that per packet multiplexing is not a sensible option as this can lead to an uneven distribution across the monitors. Given that it is common to use a switch for monitoring traffic traveling between two routers the MAC address variability is limited. There are usually only

two MAC addresses in use, one for each endpoint. Accordingly, we have to rely on at least source and destination IP addresses or better yet IP addresses and port numbers for load balancing which inconveniently rules out the cheapest switches. Still, depending on the application one may want to ensure that all packets from an IP address pair or a remote or local host are handled by a single monitor. This ensures that all packets of a single TCP connection arrive at the same monitor. With such a configuration load balancing can be expected to be reasonable as long as there are sufficiently many addresses that are monitored. If the EtherChannel feature is not supported by a specific switch one should check if it offers another way to bundle links. A good indication that a switch is capable of bundling is the support of the Link Aggregation Control Protocol (LACP).

3.3.2 Parallelizing pcap-trace based analyses

Once we are able to read all relevant packets into memory or onto disc the next challenge is to process the data. For this purpose we use several tools (see Section 2.2), such as BRO. Common tasks of these tools include TCP stream reassembly or signature matching. On first glimpse these tasks appear easy, but once they are to be done on several Terabytes worth of data, 150 million connections, or 4000 billion packets, the dimension of the challenge, especially with regards to CPU resources, becomes visible. Recent server processors usually offer multiple cores, and server boards typically host multiple processors. Hence, servers with up to eight cores are common and are available for less than 2,000 €at the time of writing.

Since most trace analysis tools are not yet multi-threaded[7], they cannot fully utilize the processing power of such a system. In order to take full advantage of all cores we have implemented SPBC[8], a small C program based on LIBPCAP, which implements the key idea from Section 3.3.1, namely connection preserving packet distribution to multiple outputs. Since we are only distributing the traffic to multiple instances of the analysis tool within a single machine this approach does not require a switch or multiple network interfaces. The program uses a hash function to determine where to redirect the output. So far, we implemented hashing on the *5-tuple* of source and destination IP, source and destination port, and the transport protocol, as well as the *3-tuple* of source and destination IP, and the transport protocol,

In our experience the tool works well for both offline trace analysis and online live measurements. The imposed overhead is low. Moreover, it can be tuned to the traffic that needs to be analyzed. If the trace contains a small number of hosts

[7] Current versions of BRO, SNORT, or WIRESHARK are single-threaded. The upcoming release (3.0) of SNORT is announced to support multi-threading.

[8] "split pcap by connection" (SPBC) is a program that uses LIBPCAP's default functions to open files for reading and writing. As the underlying system calls operate on file-handles these "files" can also be named pipes, or any of the standard file-handles. This allows a direct pipeline to multiple analyses processes, and if required, even for a live capture.

the *5-tuple* hash can be used to ensure decent load-balancing. For traces covering many thousands of hosts the *3-tuple* hash or a hash on the downstream IP address is feasible which allows detection of inter-connection effects (e. g., a DDoS attack on a downstream host) as well.

Certainly, this approach has limitations and drawbacks. Any analysis that needs full coverage of the vantage point will suffer. In addition, after the analysis the output often has to be joined together.

3.3.3 Tuning network adapter drivers for capturing

As reported in Section 3.1 and [86, 89], the capturing stacks in Linux and FreeBSD are not designed for high performance packet capture. One reason for that is that these stacks are not supposed to interfere with the normal TCP/IP stack. Therefore, these stacks impose overhead that is unnecessary if they were designed purely for packet capturing. To exploit this we design and evaluate two approaches for improving packet capture for Linux and FreeBSD. We note that changes to the kernel code are required since most of the capturing stack is implemented in the kernel or in the network driver.

The first approach is to circumvent user space when writing a trace to disk. This has the potential to save two copy operations, one to user space and than one to the I/O subsystem. Therefore, we expect it to outperform the default capturing stack. In an undergraduate project, Pronchev [82] implemented this extension to the Linux Netfilter framework. He also compared his extension to default TCPDUMP and finds that his direct approach has slightly higher throughput. However, we point out that the disk system in his experimental setup was the bottleneck, explaining the small benefits.

Another approach is to eliminate copy operations in the FreeBSD stack. In his Diplomarbeit[9] Fiveg [33] implemented a "true" zero copy driver where the DMA[10] transfer copies the packet directly into a user space shared memory buffer. He finds that this approach significantly reduces the system load and allows higher capturing bandwidth than the standard approach. Note, that there is another project called "Zero-Copy BPF" [63, 111] which also avoids copy operations, but still allows the normal network stack to work. The "Zero-Copy BPF" has been released with FreeBSD 8.0 only one month before the completion of Fiveg's thesis, and could therefore not be evaluated.

[9] A Diplomarbeit is the German equivalent to a master's thesis.

[10] A direct memory access (DMA) transfer is a possibility to copy data from the computer's periphery to the main memory. As compared to regular interrupt context transfers, DMA does not require the CPU to perform the copying, instead the target memory address is loaded to the peripheral device and an interrupt is generated once the data is available.

3.3.4 Specialized hardware for packet capture

Given the difficulties of using commodity hardware for high speed packet monitoring it is not surprising that specialized monitoring hardware is available. Endace [27] is well-known for their DAG[11] network monitoring cards. There are several advantages to these cards:

- Accurate packet time-stamping on the card itself. This is an important feature since OS based capturing stacks time-stamp the packet during the interrupts or only once the packet has been delivered to user space. Considering interrupt coalescing and scheduling issues these time-stamps may not be that accurate.
- Large on-card buffers are used to gather data in order to allow for transferring to memory in larger chunks. This reduces the risk of packet loss due to a blocking kernel and reduces the DMA transfer overhead.
- Contrary to commodity hardware, these cards are available for all link-layer types. It is difficult to get a commodity non-Ethernet card for a PC.

Of course these benefits come at a substantial increase in costs. Endace also offers complete monitoring probes, called Ninja Boxes. These are basically well-selected off-the-shelf server components and one of their DAG cards. Endace is not the only one selling specialized monitoring hardware, although they are the only ones actively supporting research. WildPackets [114], Napatech [68], and Solera Networks [97] offer alternatives to Endace. In addition to the commercial vendors there are research-driven non-profit projects such as NetFPGA [71] or OpenFlow [75] that can be leveraged for high performance packet capture.

3.4 Recommendation

Given the above insights we conclude this chapter with a set of recommendations on how to compose a high performance packet capturing infrastructure. From our results in Section 3.2 we learn that it is best to avoid systems with a front side bus (FSB) architecture, such as Intel systems prior to the Nehalem architecture. It is also beneficial to use FreeBSD when capturing with regular network cards[12]. When using Endace cards the influences of the architecture and the operating system are not significant.

A major bottleneck for throughput is still the storage speed, especially for interfaces to external storage. We recommend avoiding external storage. A seemingly better

[11] Endace call their cards "DAG" network monitoring cards after the former DAG research project at University of Waikato, in Hamilton, New Zealand, see `http://dag.cs.waikato.ac.nz`. In 2001, Endace Measurement Systems was formed to commercialize the DAG project.

[12] In our experience Intel cards perform best.

approach is to use a machine with a large chassis and many cheap internal disks. The internal disks can be SATA and do not have to be top-notch for two reasons. First, the throughput can be improved by simply using many disks. Second, the seek times are not important for linear read and write. If the throughput is limited by the RAID controller[13] plugging in another controller and using a software RAID-0 (striping) for combining both helps further.

When targeting a deployment with many different vantage points in one location, consider buying a tap selector or aggregator instead of purchasing another set of monitors for the second vantage point. One final suggestion: It is beneficial in terms of costs and operation to have a trace cruncher with huge (external, as speed does not matter) storage, many processors/cores, and a lot of main memory to process traces and perform analysis, while the monitors do not need much CPU power or memory for the task of recording to disk.

[13] We had positive experience with 3Ware [58] controllers.

Chapter 4

Impact of AJAX-based Applications on the Network Traffic

The World Wide Web [47] is one of the most popular applications of the Internet and runs primarily over the HTTP protocol. While HTTP (Hyper Text Transfer Protocol) [32] constitutes the session layer or messaging protocol of the Web, HTML (Hyper Text Markup Language) describes the content and allows authors of Web content to connect up Web pages through hypertext links or *hyperlinks*; an idea made popular by Tim Berners-Lee in the early 1990s and widely used today. In its classical form, users reach other pages or access new data by clicking on hyperlinks or submitting Web-based forms. In this basic HTTP request-response model each clicked link or submitted form results in downloading a new Webpage in response to the respective request.

The recent popularity of asynchronous communication enabled Websites has caused a significant shift from the classical HTTP request-response model of the Web. This asynchronous communication is commonly executed through AJAX (Asynchronous JavaScript and XML)[121], a compendium of technologies that enable Web browsers to request data from the server asynchronously, i. e., without requiring human intervention such as clicking on a hyperlink or on a button. Consequently, HTTP requests are increasingly becoming automated rather than being human-generated. In this chapter, we use the terms AJAX and "Web 2.0" interchangeably to refer to Web applications that use this new paradigm on the Internet.

Contemporary Webpages often contain embedded request-response functions comprising a JavaScript application engine that automatically executes in the background to asynchronously pre-fetch large quantities of data from the server. This intelligent pre-fetching is often used to mask the round trip and transmission latency of Internet connections to give the user a 'smoother' Web application experience. We highlight the differences in Figure 4.1. The JavaScript engine builds a local pre-fetched cache based on the user's interaction with the Web application and executes parts of the application logic in the client's Web browser itself instead of at the Web server. The prediction algorithms of any automated pre-fetching scheme usually result in significantly larger downloads as compared to user-initiated Web browsing due to inaccurate guesses on behalf of the prediction algorithms about which data to pre-fetch. Even when the prediction is accurate, HTTP traffic inter-request times

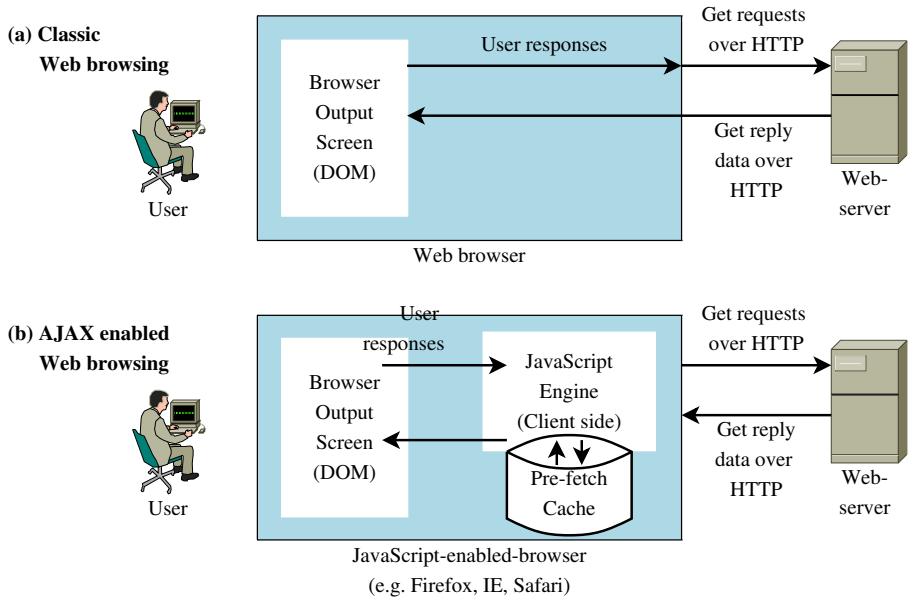


Figure 4.1: **Structural design of classical and AJAX-enabled Web applications.**

are no longer lower-bounded by human response times (order of seconds) and may instead depend on the JavaScript code logic of the Web application on the client machine.

Many popular Web applications have adopted Web 2.0 technologies. One of the most popular and early adopters of AJAX is Google Maps. Its success encouraged the use of AJAX for building other interactive Web applications. For example, many Web-email offerings have transitioned into Web 2.0 applications in order to rival the look and feel of desktop email clients. Furthermore, some social networking Websites use AJAX technologies to offer rich and interactive user experiences. In this chapter we explore the traffic characteristics of the most popular representatives of these AJAX-based applications in our environment and contrast their characteristics to those of the overall HTTP traffic.

Related Work

A good overview of traditional Web protocols is given in the book by Krishnamurthy and Rexford [47]. One of the early works on characterizing the effect of HTTP traffic and HTTP pre-fetching is by Crovella [59]. It highlights the beneficial and unwanted effects of pre-fetching HTTP data and hence further substantiates the importance of our analysis of Web 2.0 applications and their global effect on the Internet. There has been a vast literature on Internet Web caching, e. g., [1, 10, 18]. However, the

underlying motivation for using caching in all these studies has been on reducing the overall download latency of popular Websites and not facilitating low latency interactive Web 2.0 applications.

There are few studies focusing on the characteristics of AJAX-based traffic, although there exist several discussions, blogs and Websites about the end-user perceived latency of AJAX-based applications (e. g., [20]). The novel aspect of our work is that we focus on the behavior of two large user populations and investigate multiple AJAX-enabled applications.

Contributions

In this chapter, we highlight the changing characteristics of Web traffic by comparing the traffic patterns of HTTP and Web 2.0 applications. For this we rely on several HTTP traces from large user populations in Munich, Germany, and Berkeley, USA, from which we extract popular AJAX application traffic.

From the statistical analysis of Web 2.0 traffic in comparison to all HTTP traffic extracted from the traces we show that the former's characteristics significantly differ from the latter's. Our work focuses on the number of transferred bytes, the number of HTTP requests issued, and the times between subsequent requests (inter-request times). For example, Web 2.0 traffic has shorter inter-arrival times due to the underlying human-independent automated data pre-fetching schemes.

Our work complements the efforts of the Web developer community towards a better understanding of the Web 2.0 application characteristics. Some of our results may motivate the Web developer community to design Web applications that are friendlier to the underlying network, for example, by reducing the number of automated HTTP requests when possible.

The rest of the chapter is organized as follows. We give a brief overview of the applications studied in this work and then describe our data collection process in Section 4.1. In Section 4.2 we present the results of our statistical analysis comparing AJAX traffic with the HTTP traffic. Finally, we conclude in Section 4.3.

4.1 Methodology

In order to determine which Web 2.0 applications to study we first examine the popularity of different applications (Section 4.1.1). Google Maps is among the most popular Web applications and a nice example of an AJAX-enabled application. Therefore, we provide a high level overview of its communication patterns in Section 4.1.2. Finally, we detail how we extract application characteristics from our data sets in Section 4.1.3. Similar extraction methodologies (skipped for brevity) are used for the other AJAX applications.

Table 4.1: **Overview of the anonymized data sets**

trace	start date	duration	size	#req total	#req GMaps
UNI-1	Feb 24th 2007	>32h	2.4 TB	30,0 M	222 K
LAB-1	Mar 3rd 2007	∼9h	214 GB	2,0 M	82 K
UNI-2	Oct 11th 2005	24h	2.5 TB	119 M	43 K

4.1.1 Data sets

We use packet level traces collected from two independent networks: the Münchener Wissenschaftsnetz (Munich Scientific Network, MWN) in Germany, and the Lawrence Berkeley National Laboratories (LBNL) in the USA. Both environments provide high speed Internet connections to their users. The MWN provides a 10 Gbps link capacity to roughly 55,000 hosts at two major universities and several research institutes, transferring 3-6 TB a day. LBNL utilizes a 1 Gbps upstream link, transferring about 1.5 TB a day for roughly 13,000 hosts. We base our analysis on three traces from network port 80 (the HTTP port). Two of these traces, UNI-2 and UNI-1, are from MWN while one trace, LAB-1, is from LBNL. See Table 4.1 for information about the traces including: size, duration and start dates, total number of HTTP requests, and number of HTTP requests related to Google Maps.

We rely on packet level traces of large user populations as they provide the most detailed data. From these traces we reconstruct the HTTP request-response stream of all connections. While one could use a variety of tools [47], we utilize the HTTP analyzer of Bro [79], a network intrusion detection system. Bro's policy script `http.bro` together with the policy scripts `http-reply.bro` and `http-header.bro` enable TCP stream re-assembly, basic HTTP analysis, and HTTP request-response analysis. We augmented the `http-header.bro` script to extract the times when the HTTP requests were issued. The resulting output file consisted of one-line summaries of each HTTP request containing (TCP) connection ID, number of request in the connection, session ID, transferred bytes, three time-stamps (request issued, cookie seen, request finished), requested hostname (servername[1]), prefix of the requested URL, and the HTTP status code for this request. Note that the number of transferred bytes does not include the HTTP header size. We only include requests for which we successfully record start and end times.

In order to determine the most popular AJAX-enabled Web 2.0 applications we first identified the 500 most popular Web servers[2] in the UNI-1 data set. We then

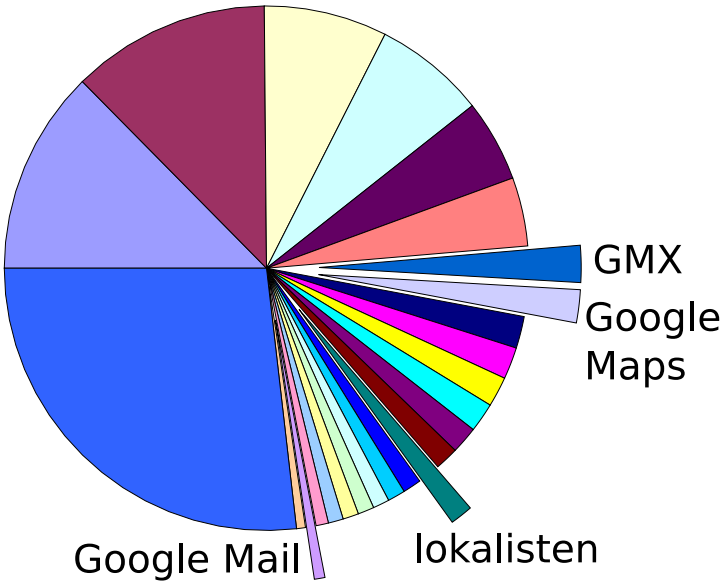
[1] We use server and host interchangeably in this discussion.
[2] We extract the name of the Web server from the hostname in the HTTP request.

grouped these into multiple categories for better visualization. The first set of categories contained the servers that are hosted by the two universities and the other research institutes (MWN). The next categories contained all request related to advertisements (Ad Server) and news Websites (News). Manual inspection showed that neither category contained many AJAX related requests. Some of the services offered by Google, including Google Maps and Google Mail, use AJAX while others like Google search, Google images, and Google Earth do not. Accordingly, we separate them into Google Maps, Google Mail, Google Earth, and all others (Google). Another popular Web email service in Germany that is also AJAX supported, is provided by GMX. Some categories include just a single popular site (Site 1, ..., Site 5), others are well known Websites, e. g., Ebay and MSN. Figure 4.2 shows a pie chart of the number of requests per category for the UNI-1 data set. We find that GMX is the most popular AJAX-based application with 2.27 % of the requests followed by Google Maps which contributes 2.04 %. Another AJAX-enabled social networking Website is `lokalisten.de` with 1.4 %. Although Google Mail only accounts for 0.65 % of the requests we include it as our fourth application since this gives us two AJAX-enabled Mail applications by different providers. In terms of bytes the contributions are smaller, e. g., Google Maps with 1.41 %. But all of the applications considered in this chapter are among the top 500. We refer to these both most popular and AJAX-enabled applications as "Selected-4" in subsequent discussions.

4.1.2 Google Maps communication

Google Maps is one of the first Web applications to popularize AJAX technology. Consequently, it is widely considered as the canonical example of an AJAX application. AJAX uses the Document Object Model (DOM) [24] of the Web browser such that it is no longer necessary to reload the entire Webpage each time it is updated. In this way it increases interactivity, speed, and usability.

Google Maps maintains multiple connections to different servers in the Internet that serve as back-ends for the Google Maps application. All connections use HTTP as the session protocol and take advantage of the advanced features of HTTP 1.1 [32] such as persistent HTTP connections for efficiency and pipelining for reducing latency, leading to multiple HTTP requests per TCP connection. In the context of Google Maps, most of these connections are used to fetch image tiles of the map. The others are used for control messages and for the initial transfer of the AJAX application (JavaScript code), the transfer of other GUI related pictures, and user queries. The connections carrying tile images can be identified by the servers they connect to.



Category	Percent	Category	Percent
LMU	12.60 %	Ebay	1.67 %
Ad Server	12.26 %	Site 3	1.45 %
MWN	7.60 %	`lokalisten.de`	1.40 %
Google	6.90 %	Bav. State	1.14 %
TUM	5.05 %	`web.de`	1.09 %
News	4.23 %	Site 4	1.01 %
GMX	2.27 %	Site 5	1.00 %
Google Maps	2.04 %	StudiVZ	0.97 %
Yahoo	1.97 %	MSN	0.93 %
Site 1	1.96 %	Microsoft	0.76 %
Google Earth	1.85 %	Google Mail	0.65 %
Site 2	1.81 %	Youtube	0.55 %
		other	26.83 %

Figure 4.2: **Pie chart of fraction of requests per hostname categories for the top 500 hostnames.** (Top 500 hostnames account for 53 % of the total requests. Percentages are relative to the top 500.)

4.1.3 Application characterization methodology

In this section, we discuss how to extract application specific data from our data sets. For brevity reasons we focus on Google Maps traffic.

One of the challenges of identifying Google Maps traffic is that Google offers all its services (e. g., Google Maps, Google Search, Google Video, etc.) on the same back-end server infrastructure and uses a uniform key for all services. Therefore, the browser can reuse existing TCP connections to Google servers to issue Google search queries, image or video queries, as well as Google Maps queries. Thus, separating Google Maps traffic from other Google services requires some effort. Moreover, to capture the user's interaction with Google Maps, we are not only interested in individual HTTP requests but also in the full set of HTTP requests within a Google Maps "session". Meaning all requests that are issued when a user connects to `maps.google.com` and then interacts with the application, e. g., by entering some location, by moving the map, or switching the zoom level. Accordingly, we group these requests to a Google Maps "session".

To identify Google Maps related requests among the very large number of HTTP requests within our traces we check if the hostname contains the string `maps.google`. To find the other requests by the same user we take advantage of Google's own session book-keeping mechanisms. Google uses cookies to mark all requests of a session by embedding a unique hash of its session ID[3]. We use this ID as our session ID as well and gather all other requests of this Google Maps session using the session ID. Unfortunately, there maybe additional requests to other Google services among the identified requests. We exclude these if they do not contain a Google Maps specific URL prefix. We found that /mt (map), `/kh` (satellite), `/mld` (route planning) and `/mapstt` (traffic) are related to the kind of map that is requested. `/maps`, `/mapfiles` and `/intl` are used for meta information. `/` and `favicon.ico` are not restricted to Google Maps use. A similar methodology is used for the other Selected-4 applications.

For comparison purposes, we also group requests of the complete HTTP traffic (ALL-HTTP), including requests of the Selected-4, into Web sessions. In this case, we cannot take advantage of cookies yielding session identifiers. Therefore, we group those requests that come from the same client IP go to the same server (IP) on the same server port. This aggregates connections from different client side ports.

For both Selected-4 sessions and ALL-HTTP sessions we use a timeout[4] of 10 minutes. We compute per connection and per session statistics including number of transferred HTTP payload bytes, number of requests, their durations, and inter-request times (IRTs) for the Selected-4 applications as well as ALL-HTTP traffic.

[3]The hash is located after the string `PREF=ID=` in the cookie.

[4]If the time between the end of a reply and the start of the next request is larger than 10 minutes a new session is started.

4.2 Characteristics of AJAX traffic

In this section, we present the results of a statistical analysis of the characteristics of both ALL-HTTP and Selected-4 traffic. Almost all connections and sessions are usually comprised of multiple requests. However, we find significant differences in the session characteristics, including session life times, transferred bytes per session, number of requests within sessions, and inter-arrival times of HTTP requests within sessions.

Most of the data is presented as probability density functions (PDF) although complementary cumulative distribution functions (CCDFs) are also shown. In order to capture the multiple orders of magnitude in the data we plot all CCDFs on a log-log scale and compute the PDFs of the logarithm of the data in order to be able to use a logarithmic X-axis. In addition, Table 4.2 presents the number of request and number of sessions, and Table 4.3 the mean and median values.

In our analysis we concentrate on the UNI-1 data set and only use the UNI-2 and LAB-1 data sets to highlight some of the noticeable differences. Note, that the 2005 data set (UNI-2) was collected during Google Maps beta testing phase.

Figure 4.3 shows the CCDF of the number of bytes transferred in a single HTTP connection for ALL-HTTP and all Selected-4 applications for the UNI-1 data set. ALL-HTTP connections are clearly consistent with a heavy-tailed distribution over several orders of magnitude with a median of 332 Bytes and a mean of 58 KB. Some connections are clearly used to transfer a huge number of bytes, e. g., due to downloading some large image or video file embedded within a HTTP page, or a big software package, or when HTTP is used as transport protocol for P2P protocols, such as Bittorrent.

The tails of the AJAX-based Selected-4 applications are not as heavy. Yet, except for Google Mail the curves lie on top of the ALL-HTTP traffic for most of the plot which is reflected in the statistics as well, e. g., the median and mean for Google Maps is larger, i. e., 25 KB and 204 KB, respectively.

To further explore the differences in the body of the distribution we show the PDFs for Google Maps and Mail as well as for ALL-HTTP traffic in Figure 4.4. In general we note that the Selected-4 applications (see for example, Google Maps) transfer more bytes than ALL-HTTP connections. This probably stems from multiple larger image/JavaScript library transfers, when, for example, Google Maps users pan and zoom their map. In particular, only 39.6 % of the UNI-1 Google Maps connections comprise of connections that transfer less than 10 KB, whereas 81.8 % of the ALL-HTTP connections from UNI-1 transfer less than 10 KB. Similar observations hold for the LAB-1 data set. Moreover, we note that the shape of the ALL-HTTP connection has not changed substantially over the years if compared with results from 1997 [31].

Table 4.2: **Number of requests and sessions per application for ALL-HTTP and Selected-4 in UNI-1.**

Application	#requests	#sessions
ALL-HTTP	30,000 K	1.4 M
Google Maps	221 K	1127
`lokalisten.de`	128 K	3822
Google Mail	140 K	1020
GMX	288 K	6101

Table 4.3: **Mean/median of application characteristics for ALL-HTTP and Selected-4 in UNI-1. IRTs are inter-request(-arrival) times.**

Application	bytes per connection	bytes per session	#req per connection	#req per session	IRTs in a connection	IRTs in a session
	mean					
ALL-HTTP	57890	278K	4	13	2.34	17.23
Google Maps	204476	2288K	18	197	1.39	1.54
`lokalisten.de`	31856	129K	8	34	0.38	4.52
Google Mail	9742	371K	4	138	23.02	31.84
GMX	14163	95K	7	47	0.53	4.29
	median					
ALL-HTTP	332	688	1	2	0.0987	0.2035
Google Maps	25199	161675	4	21	0.0288	0.0076
`lokalisten.de`	1678	7854	3	7	0.0347	0.0406
Google Mail	3	27932	1	23	4.3735	9.2202
GMX	428	6863	3	29	0.0400	0.0489

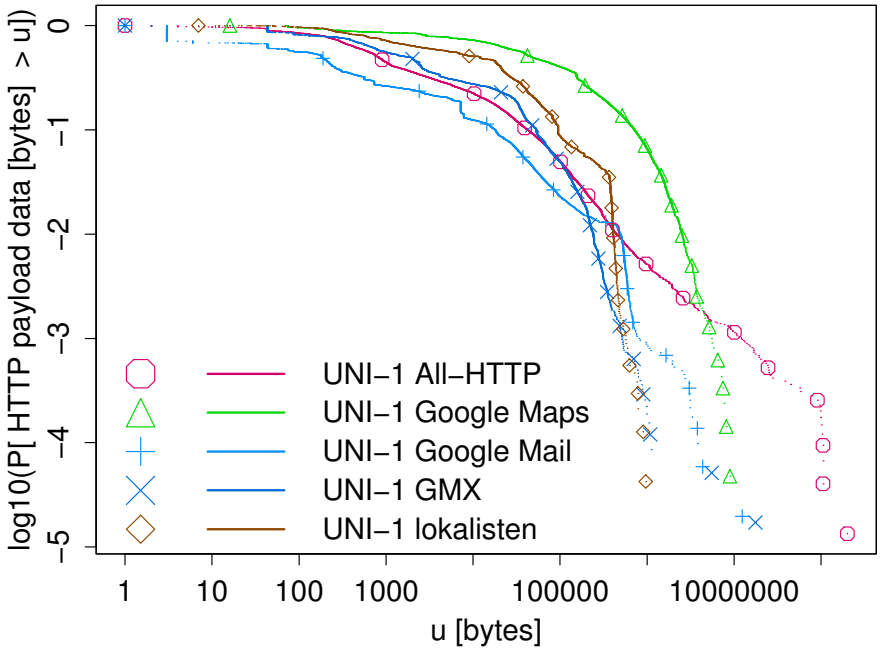


Figure 4.3: **CCDF of HTTP payload bytes per connection for ALL-HTTP and Selected-4 in UNI-1**

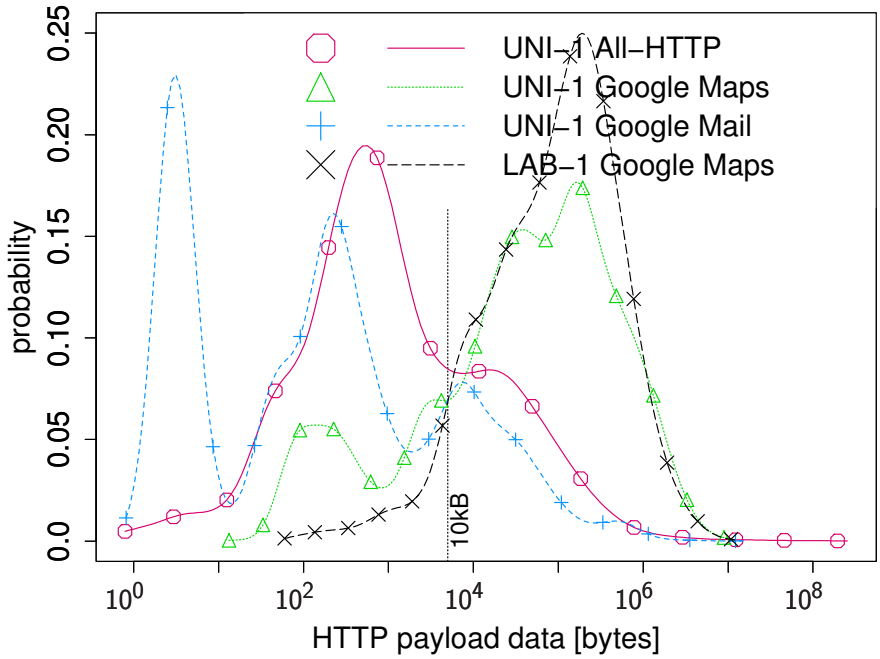


Figure 4.4: **PDF of HTTP payload bytes per connection for ALL-HTTP and Google Mail in UNI-1 & Google Maps in UNI-1 and LAB-1**

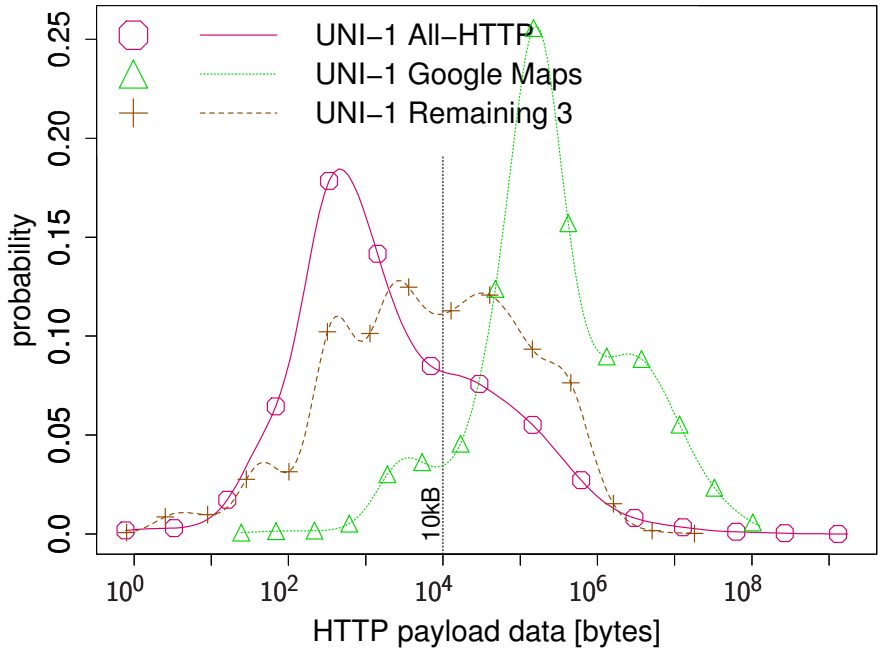


Figure 4.5: **PDF of HTTP payload bytes per session for ALL-HTTP, Google Maps and Remaining-3 in UNI-1**

Google Mail differs and shows a clear spike for 3 bytes requests. This is due to periodic server polling by the client-side AJAX engine of Google Mail. Once we move from HTTP connections to HTTP sessions (Figure 4.5), this artifact is removed and the probability mass of all Selected-4 applications clearly lies to the right of that for ALL-HTTP traffic. This is reflected in the median but not in all means. But recall that the mean is dominated by the very large transfers within the ALL-HTTP traffic.

We next move to the number of HTTP request within a session. Figures 4.6 and 4.7 show the CCDF and PDF for ALL-HTTP and Selected-4 sessions in the UNI-1 data set. These figures highlight the "chatty" nature of the Selected-4 applications—on average they issue many more requests than ALL-HTTP traffic whose first fifty percent of the sessions are limited to two requests. Part of these additional requests are due to the Web 2.0 characteristics of the Selected-4 applications while the others are likely due to longer session duration. Interestingly, a look at the PDF reveals that Google Maps issues more requests than the email or social networking applications. A likely explanation is that Google Maps implements pre-fetching more aggressively.

The typical duration of an ALL-HTTP session (Figure 4.8) is shorter than for AJAX-enabled applications. Half of the ALL-HTTP sessions last between 0.008 and 2.13 seconds (5 %–55 % quantile across all sessions) while 50 % of Google Maps sessions

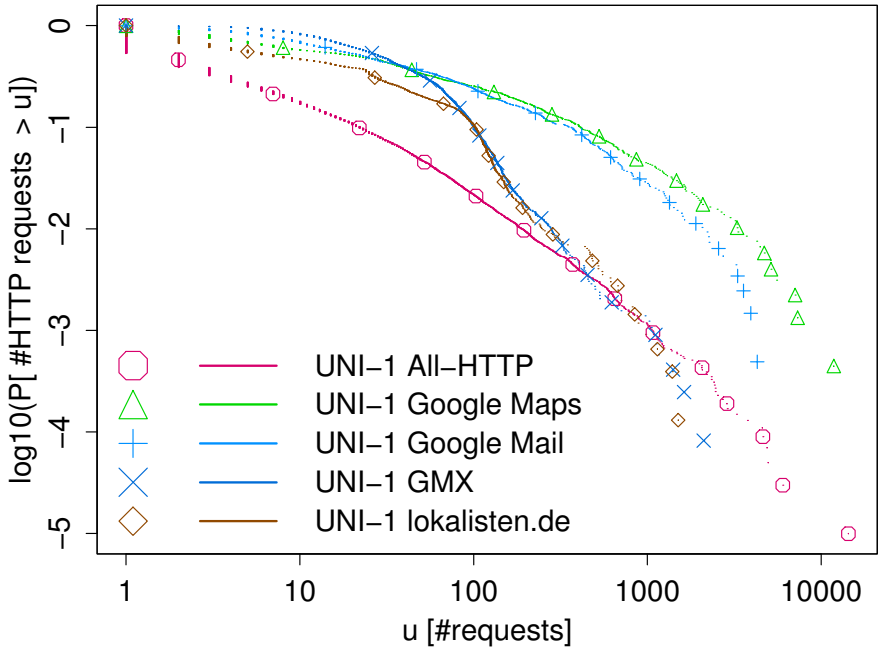


Figure 4.6: **CCDF of requests per session for ALL-HTTP and Selected-4 in UNI-1**

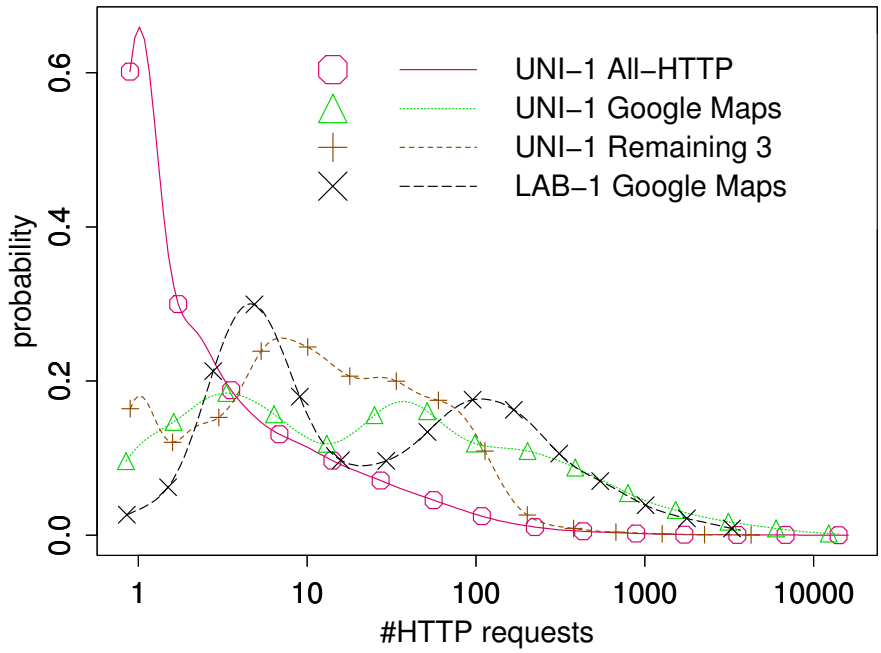


Figure 4.7: **PDF of requests per session for ALL-HTTP and Google Mail in UNI-1 & Google Maps in UNI-1 and LAB-1**

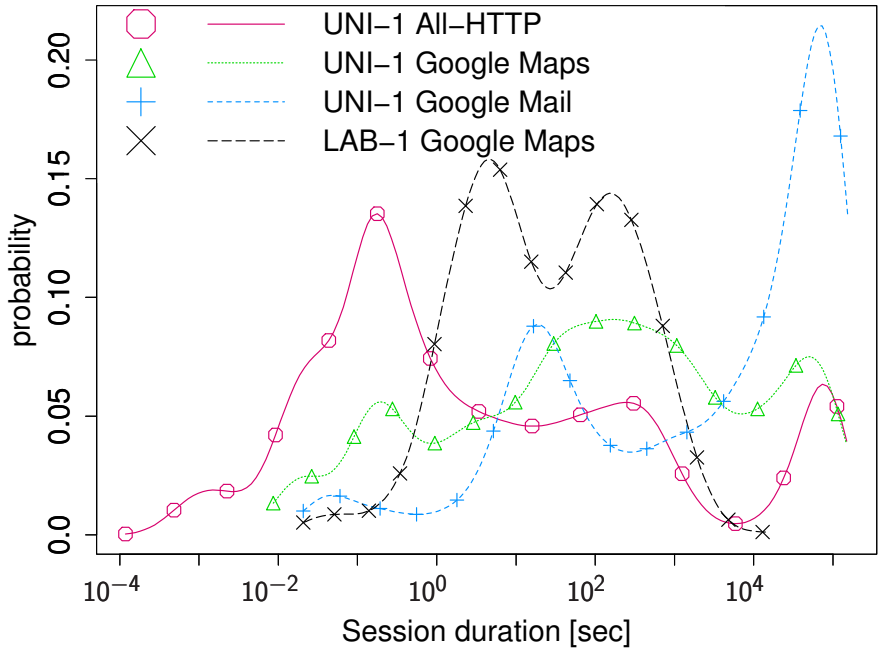


Figure 4.8: **PDF of session durations for ALL-HTTP and Google Mail in UNI-1 & Google Maps in UNI-1 and LAB-1**

in the UNI-1 data set last between 13.04 seconds and 2 hours and 9 minutes (30 %–80 % quantile across all sessions). On the other hand, the first period only accounts for 20.7 % of the Google Maps session while the second only accounts for 23.87 % of the ALL-HTTP traffic. One reason for the longer session duration may be that these specific applications are able to keep the users attention longer than a typical Website. Overall, these characteristics indicate that AJAX-enabled applications last longer and are more active than ALL-HTTP sessions.

Finally, Figures 4.9 and 4.10 show the inter-request times between requests within a session. The most interesting feature of this density graph is that Google Maps' inter-request times are very similar and significantly shorter, i. e., more frequent, than for ALL-HTTP for both UNI-1 and LAB-1. As such the traffic pattern is burstier. Moreover, there has not been a major change for ALL-HTTP from 2005 to 2007. The majority of requests is clearly automatically generated, as they are executed within 1 second (see support line; >1 second corresponds roughly to human-issued browser request) in all sessions. Google Maps is again the most extreme application. Most likely this is caused by the utilization of pre-fetching for supporting the dynamic features of Google Maps.

Moreover, we note that different service providers can use the AJAX capabilities in different manners. GMX and Google Mail are both Web-based email applications. Yet, the inter-request times differ dramatically. The reason for this is that

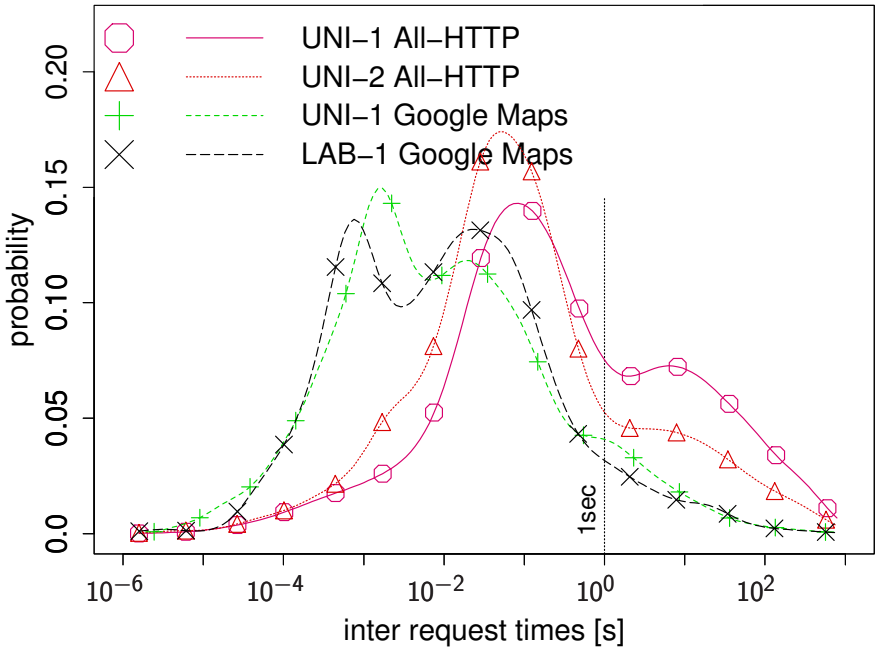


Figure 4.9: **PDF of inter-request times within each session for ALL-HTTP in UNI-1 and UNI-2 & Google Maps in UNI-1 and LAB-1**

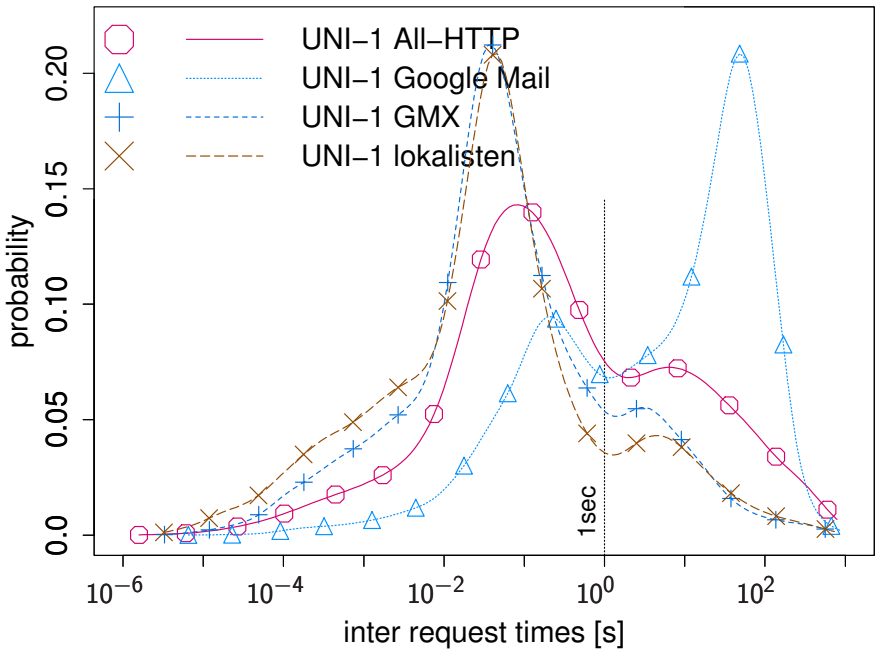


Figure 4.10: **PDF of inter-request times within each session for Google Mail, GMX and `lokalisten.de` in UNI-1**

Google Mail uses a polling interval of roughly 120 seconds (the 3 Bytes requests from Figure 4.4). Once these are removed the densities are quite similar again.

4.3 Summary

The overall transition of the Web from a hyperlinked document repository into a real-time application platform has ramifications for the underlying Internet over which Web traffic is transferred. In this chapter, we highlight characteristics of some popular Web 2.0 applications, in particular—Google Maps, Google Mail, `lokalisten.de`, and GMX Mail. We report that these applications are heavy (bytes transferred), chatty (many more requests), and greedy (actively pre-fetching data). Our analysis of their traffic patterns suggests that their characteristics translate into more aggressive and bursty network usage as compared to the overall HTTP traffic.

End users have come to expect contemporary Web applications to be as responsive as locally installed software applications which imposes high QoS requirements. Yet, treating this new HTTP traffic as relatively deterministic flows (i. e., in the same way as streamed media) is bound to fail due to the inherent variability.

Web application developers have embraced data pre-fetching, HTTP connection persistence, HTTP pipelining, and other advanced features to mask network latency from end users. The results in this chapter may help Web application developers in understanding how their applications affect Internet traffic and how their applications can be designed for more efficient operation.

Chapter 5

Usage of Online Social Networks

Online Social Networks (OSNs) such as Facebook, MySpace, LinkedIn, Hi5, and StudiVZ, have become popular within the last few years. OSNs form online communities among people with common interests, activities, backgrounds, and/or friendships. Most OSNs are Web-based and allow users to upload profiles (text, images, and video) and interact with others in numerous ways. The contemporaneous rise of Web 2.0 technology and user-generated content has resulted in over half a billion users being present on the OSN ecosystem. Facebook alone adds over 377,000 users every twenty-four hours and is expected to overtake MySpace in the total number of users in 2009 [7].

This sheer number of users makes OSN usage interesting for different entities: *(i)* ISPs have to transport the data back and forth and provide the connectivity, *(ii)* OSN service providers need to develop and operate scalable systems, and *(iii)* researchers and developers have to identify trends and suggest improvements or new designs. The questions this chapter aims at answering therefore include *Which features of OSNs are popular and capture the users attention?*, *What is the impact of OSNs on the network?*, *What needs to be considered when designing future OSNs?*, *Is the user's behavior homogeneous?*

A recent study [46] explores the properties of OSNs worth examining, presents methodologies available, and discusses various challenges associated with measuring them. Earlier work studied the graph properties of the online communities, high level properties of snapshots of individual OSNs, and issues related to anonymization and privacy. However, they can only capture the state of an OSN as inferred by some specific measurement technique, e. g., crawling. Furthermore, there are no known studies which document how users interact with various OSNs beyond those that rely on surveys [2, 26, 101] and interviews [43]. Moreover, such techniques are limited in scope and cannot capture OSN macro-level properties such as overall volume of traffic, its dynamics, etc. or micro-level properties such as what happens within an OSN when users interact with it. To analyze such properties one has to capture the interactions of the user with the OSN over time which is impossible via crawling.

This chapter focuses exclusively on these understudied properties by examining actual user clickstreams. We extract clickstreams from several anonymized HTTP

header traces[1] from large user populations collected at different vantage points within large ISPs across two continents. We focus on OSNs whose primary content are user maintained profiles. We chose Facebook, LinkedIn, Hi5, and StudiVZ [30, 40, 57, 100] because they are popular, well known, and well represented in our traces. We present a methodology that allows us to reverse engineer user interactions with OSNs from network traces.

Unfortunately, currently available clickstream data sets are very limited. In principle, there are three ways of gathering such data, either on the server, on the client side, or at a proxy/aggregator. As server-side data is considered proprietary, datasets are limited. For search engines there are some examples [95, 98, 99]. However, none of the server-side data sets can include the full clickstream as the full clickstream consists of all user accesses to all Web pages related to the OSN or the search query. Previous work based on client-side data gathering has focused on Web search clickstreams [44] or on asking volunteers to interact with the OSN [43]. Other approaches include surfing the Web using additional browser plug-ins, e. g. [112], or enhancing HTTP proxies with extended logging functionality, e. g. [8]. However, ISPs servicing residential customers do not necessarily use proxies nor do volunteer interactions with an OSN necessarily correspond to their natural behavior. Recently, Benevenuto et al. [12] have analyzed clickstream data from a Brazilian social network aggregator.

Using our methodology we are able to track the beginning and ending of a user's interaction with an OSN as well as various intra-OSN actions performed by the user. We apply this methodology to each of the four selected OSNs and validate the results using traces of manual interactions with these OSNs. We present results on feature popularity *within* OSNs, OSN session characteristics, and on dynamics within sessions. For example:

- We find that users commonly spend more than half an hour interacting with the OSNs. For Facebook users, we verified that while users interact with the OSN, only a minority of them accesses any non-Facebook sites.
- While we selected OSNs based on the criterion that they feature profiles, profiles are only the most popular feature within LinkedIn and StudiVZ. Within Facebook and Hi5 profiles are among the popular features besides downloading photos and exchanging messages. In addition, the most popular features in terms of clicks usually do not contribute the most to the traffic volume. With regards to volume, photos play a major role, especially with regards to uplink bandwidth.
- We find that the number of accesses to profiles within the session is highly skewed. While there are some sessions with many accesses (> 100) most users

[1] All IP addresses are anonymized and HTTP content is excluded. Furthermore, we apply anonymization to any other field that has the potential to contain user related data before processing the data.

only access a handful. Indeed, in terms of unique profiles the number is even lower. This indicates that the richness of the friendship graph is not a good indicator of how many profiles will actually be accessed during a session.

This results in an in-depth understanding of what happens *within* an OSN and also allows us to look for similarity *across* OSNs. Our work complements the efforts of the OSN community towards a better understanding of how OSNs are used. Service providers benefit by knowing OSN features that are of significant interest to their own users and by understanding what else on the Web is important to their users. This allows them to influence and improve their own service offerings. Researchers can propose improvements or simplifications for existing OSNs or design OSNs with novel features. From an ISP viewpoint OSNs currently contribute a lot less than peer-to-peer applications in terms of bytes. However, OSNs might add features that increase the per-user bandwidth demand. Given this potential for traffic explosion (e. g., when video becomes popular within OSNs), it is imperative to understand the network-level dynamics of OSNs.

The remainder of this chapter is structured as follows: In Section 5.1 we give an overview of OSN features and introduce our terminology. After giving a description of the data sets we use in Section 5.2, we discuss our analysis methodology, Section 5.3. We then present the results of our macro analysis with regards to feature popularity analysis in Section 5.4, and session characteristics in Section 5.5, followed by our micro-level analysis regarding the dynamics within OSN sessions in Section 5.6. Finally, after reviewing related work in Section 5.7, we summarize our experience and suggest future research directions in Section 5.8.

5.1 OSN features and terminology

Before delving into the user session and clickstream analysis we introduce the terminology we use in the rest of the chapter by discussing which features OSNs offer and how a sample OSN session is seen from a network perspective.

5.1.1 OSN features

Most OSNs include features for creating user accounts and authenticating users. A user's basic profile includes entries for age or home town. OSNs offer a variety of different features that are commonly accessible only to those users that are logged in. Users can update their profiles (contact information, photographs, information about hobbies, books, movies, music, etc.), browse other users' profiles by searching and subsequently obtaining lists of their friends and narrowing them via categories like schools or work sites. They can add friends, invite new friends, join groups or

networks, communicate with other users via OSN-internal email services, writing on other users' "walls" or on discussion forums, and adjust their privacy settings.

Several OSNs offer a platform to build third-party applications; these are hosted on external servers by the individual application writers and allow OSN users to exploit the social graph and download and interact with each other through the application. Popular applications are of the social utility variety (e. g., dating) or games. There are also several applications that are internal to the OSN. We do not explore specific application features in this chapter.

5.1.2 A "sample" Facebook session

Next, we show how a sample OSN session is seen from a network perspective. Users must login before accessing any Facebook feature and this starts the *OSN session*; after logging in the user is *authenticated*. At the end of the session a *logout* results in the user becoming *offline*. The time between login and logout is an *authenticated OSN session*, while the time before logging in and after logging out is an *offline OSN session*. A subsequent logging in ends the current offline OSN session and starts another authenticated OSN session. The overall time from a first contact of the OSN site or a logout to another logout is an *OSN subsession*.

Once authenticated, a user starts using the Facebook features—we label these as *actions* or *clicks*. Our sample OSN session (shown in Table 5.1) has six actions (a)–(f); each of which corresponds to a user click. We group actions into *categories*—e. g., the category "photos" includes managing, uploading, displaying, commenting on, etc. of photos. Such interactions result in multiple HTTP request response pairs (*rr-pairs*) seen on the network. Table 5.1 lists all rr-pairs to `www.facebook.com`. OSNs sometimes use HTTPS, HTTP over SSL, rather than HTTP for rr-pairs carrying account credentials or CAPTCHAs. For example Facebook uses HTTPS for the login action—rr-pair 2 in Table 5.1. These are not easily observable and thus a HTTP rr-pair only log does not include this request due to the SSL encryption. But, we do observe traffic over a HTTPS connection at time 29.121.

Not all *actions* correspond to a single rr-pair. For example, the "open friend list" action generates rr-pairs 4–6. Rr-pair 4 is directly triggered by the user's mouse click while rr-pairs 5 and 6 are generated by Facebook's AJAX-based user interface. We call the first kind of rr-pairs *active* and the others *indirect*. The indirect requests also include requests for loading embedded images, JavaScript snippets, etc. (not shown in Table 5.1). An action refers to the active and its associated indirect requests. Therefore, the number of bytes in an action is the sum of bytes of all its requests.

Indirect requests are not limited to the main domain `www.facebook.com`. They can also be directed to other locations, e. g., Facebook utilizes a CDN; most Facebook images are retrieved from `static.ak.fbcdn.net`. In addition, Facebook pushes information to their users via the servers `channel.[a-Z0-9]*.facebook.com`.

Table 5.1: **Example of a Facebook interaction: Action and rr-pairs of the requests to `www.facebook.com`**

Time [sec]	Action/Click No.	Proto	Method	URI
0.000	a) open `www.facebook.com`			
9.944	1	HTTP	GET	/
27.696	b) login, enter password			
29.121	2	**HTTPS**	POST	/login.php?
31.012	3	HTTP	GET	/home.php?
45.513	c) open friend list			
47.631	4	HTTP	GET	/friends/?ref=tn&quickling[version]=141637;0&_ecdc=check
48.672	5	HTTP	GET	/friends/ajax/friends.php?membership=1&_ecdc=check
56.441	d) select profile of a friend			
59.199	7	HTTP	GET	/profile.php?id=XXX&quickling[version]=141637;0&_ecdc=check
95.921	e) write "`posted something on the wall`" on friends wall			
97.947	8	HTTP	POST	/ajax/profile/composer.php?_ecdc=false
102.841	f) logout			
105.029	8	HTTP	GET	/logout.php?h=c909dd2db7b0a83b238ea70321d2041b&ref=mb
105.341	9	HTTP	GET	/index.php?lh=c909dd2db7b0a83b238ea70321d2041b&

Table 5.2: **Overview of anonymized HTTP header traces**

ISP/ID	start date	duration	sites	size	rr-pairs
ISP-A1	22 Aug'08 noon	24h	all	>5 TB	>80 M
ISP-A2	18 Sep'08 4am	48h	all	>10 TB	>200 M
ISP-A3	01 Apr'09 2am	24h	all	>6 TB	>170 M
ISP-B1	21 Feb'08 7pm	25h	OSNs	>15 GB	>2 M
ISP-B2	14 Jun'08 8pm	38h	OSNs	>50 GB	>3 M
ISP-B3	23 Jun'08 10am	>7d	OSNs	>110 GB	>7 M

Section 5.1 shows that we can in principle identify OSN sessions and the associated clickstream from a network perspective by passively monitoring the rr-pairs.

5.2 Data sets

In this section we describe the anonymized data sets gathered at different vantage points representing actions of tens of thousands of OSN users within two large international ISPs. We focus on OSNs that allow users to maintain profiles and have different communication mechanisms with Facebook as an obvious candidate. Hi5 and LinkedIn are also popular in the U. S. StudiVZ is the most popular OSN in Germany after YouTube (which focuses primarily on video and has limited internal communication mechanisms) and Facebook. We thus chose these four OSNs to study. Accordingly, the sites monitored for the OSNs were: `facebook.com`, `thefacebook.com`, `fbcdn.net`, `fbcdn.com` (Facebook), `hi5.com`, `hi5modules.com`, `hi5networks.com` (Hi5), `linkedin.com`, `lmodules.com`, `linkedinlabs.com`, `linkedin.custhelp.com` (LinkedIn), and `studivz.net`, `studivz.de`, `studivz.ivwbox.de`, `imagevz.net` (StudiVZ).

We had access to multiple sets of anonymized HTTP header traces (see Table 5.2) from two commercial ISPs, ISP-A (traces ISP-A1, ISP-A2, and ISP-A3) and ISP-B (traces ISP-B1, ISP-B2, ISP-B3). Each site connects more than 20,000 DSL users to the Internet via at least a 1 Gbps uplink. The monitoring infrastructure uses Endace DAG network monitoring cards [27] for traffic capture. The data anonymization and HTTP header extraction is performed immediately on the secured measurement infrastructure via the HTTP analyzer of the Bro IDS [78, 79]. While some traces include all rr-pairs some only include rr-pairs to the OSN sites (see Table 5.2). To resolve the sites to appropriate IP addresses we did DNS resolutions at multiple vantage points to exclude biases due to DNS load balancing or traffic flow optimizations. Unfortunately, this restriction to OSN sites means that we do not have HTTPS flow data available for some of these traces. To compute the HTTPS flow records we use custom software with a 15 second inactivity timeout. Table 5.2 gives an overview of

the data traces including when they were gathered (local time zone, all in 2008/09) and approximate numbers on their overall size before HTTP header extraction and the number of rr-pairs that they contain.

In general, we observe that only a subset of the 20,000 DSL users actually use any OSN during the trace collection period. Overall we identified roughly 2500 (6000) users at ISP-A (ISP-B) who use any of the OSNs under study. We observed significant activity for Facebook and StudiVZ in ISP-A and Facebook, Hi5, and LinkedIn in ISP-B—well beyond 1000 users and 100,000 rr-pairs.

As we cannot show all plots for all traces we pick representative plots for presentation, usually either from trace ISP-A2 or ISP-B3 as these are the ones with the largest number of rr-pairs. Unless stated differently, similarities and differences between OSNs for one trace are also observable in other traces from that ISP.

5.3 Approach

To understand how users interact with OSNs, we extract **OSN clickstreams** from various anonymized HTTP header traces (see Section 5.2). From this main data source we identify:

OSN session clickstreams for the subset of users who interact with an OSN under study (Both ISPs).

All HTTP request/response pairs for all users who interact with any Web server (ISP-A only).

Standard browsers, proxies, or social network providers cannot provide us with this kind of data. After a short summary we present our general analysis methodology and its validation in more detail.

In summary, our methodology relies on identifying those HTTP request response pairs (rr-pairs) that are part of an OSN session and then grouping them into sessions utilizing the OSN session cookies. Within a session, we separate the time that a user is actually authenticated (online) from the period he is offline. Next, we separate direct user actions, called active rr-pairs, from follow-up requests, called indirect rr-pairs. Finally, we associate a category, e. g., photo, profile, or home, with each rr-pair. To cross check our methodology we rely on a set of manual traces. The manual traces include both, the action that the user performed on the OSN site and the resulting rr-pairs. Therefore, the manual traces serve as ground truth for the validation of our methodology.

5.3.1 Methodology

After using the typical tricks to extract clickstreams [44], e. g., the HTTP analyzer of the Bro IDS [78, 79], we need to group the clicks into OSN sessions and identify which of the OSN features are actually used. In principle, the approach outlined below can be used for any Web-based service that requires login and offers multiple different features. However, understanding how users interact with the OSN is of particular interest due to their popularity, their diversity, their complexity, and their continual evolution. Indeed, as an OSN becomes more popular it get closer to certain scalability limits of different parts of the Web service infrastructure. As such, some OSNs delegate static content not only to separate servers but to separate domains while others rely on CDNs to increase scalability. Moreover, due to changes in feature sets they may restructure their software interface, e. g., as Facebook did in early September 2008 [29]. Therefore, the analysis software needs to be easily customizable and highly flexible.

OSN session handling

Web-based service sessions are much more complex than the simple one outlined in Table 5.1: Sessions from different users overlap, users may utilize multiple OSN sessions in parallel with different user names, users may not logout at the end of a session, or the trace may start in the middle of an OSN session, etc. We next address how we identify an OSN session.

Since OSNs require login they must track their users via the appropriate HTTP mechanism: Cookies. They can be set via standard `SET-COOKIE` HTTP response headers and then the client returns the cookie in subsequent HTTP requests via the `COOKIE` request header. OSNs typically rely on a *session cookie* which is assigned at the start of the first OSN session. Even if it is changed during the session we can track such changes. In some OSNs, these cookies even persist if the user logs back in after a logout. We can thus group rr-pairs by the anonymized IP address and the anonymized session cookie. However, OSNs are not standardized. Hence, each of them uses different kinds of cookies. Most OSNs use separate cookies in addition to the session cookie to refer to the OSN user and their login, typically an email address. We refer to these as *user cookie* and *login cookie*.

Within such a group of rr-pairs we have to identify logins and logouts to distinguish authenticated and offline periods in a session. We refer to this process as *state handling*. Most OSNs appear to use separate scripts for handling login and logout; these update the user and the login cookies.

For sessions that start and end in the middle of our traces we can identify a login/logout process by looking for the specific URI (e. g., Facebook logout URI is `/logout.php?`). Some OSNs, including Facebook, LinkedIn, and StudiVZ, use

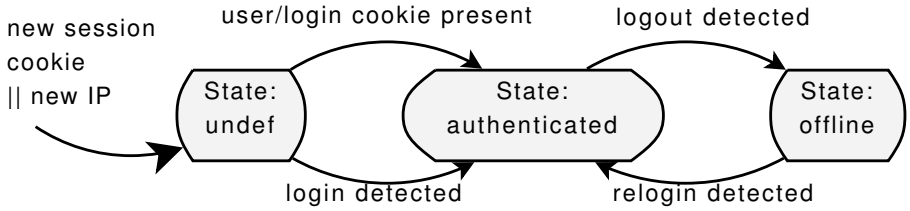


Figure 5.1: **State handling diagram for OSN sessions**

HTTPS (see Table 5.3) instead of HTTP for these scripts. The remainder of the OSN session is typically not encrypted. Thus, we augment our anonymized HTTP header traces with flow traces[2] of HTTPS activity on port 443. Note, after login and logout, users are usually redirected to a welcome/goodbye page. By checking for the specific URIs of these pages in combination with prior HTTPS activity we can identify such logins/logouts.

For sessions that start before the beginning of the network-based observation we can check if they are authenticated or offline by looking for the presence of the user/login cookie in the HTTP headers. If they are present the session is authenticated, else offline. For sessions where no logout was detected within the network-based observation we have no way to infer their ultimate duration; thus we assume that they ended with the last observed request. For these sessions we underestimate their durations and volume. Given the fact that we do not have HTTPS data for ISP-B and the smaller user count for LinkedIn and Hi5, we only have login/logout data worth reporting on ISP-A for Facebook and StudiVZ: For StudiVZ, we hardly observe any sessions that are missing the login, due to automated logout after an idle timeout. However, we observe sessions that do miss their logout. The numbers for StudiVZ and Facebook are in the same order of magnitude, and around 13–19 %. For Facebook, we find that 7–8 % of the authenticated sessions are missing both, login and logout, while 10–15 % are missing only their login. The principle state handling is shown in Figure 5.1.

Rr-pairs classification

Table 5.1 underscores our experience that it is possible to infer the OSN feature (action) associated with each rr-pair by inspecting the URI. This is true for all active rr-pairs and we built suitable patterns. Classification of rr-pairs is complex due to embedded objects and rather than finding patterns for all, we use the HTTP `REFERER`

[2] A flow summarizes a stream of packets that are selected by some criteria such that the time between packets of a flow never exceeds a specific timeout value. Our criterion is the five-tuple consisting of anonymized IP addresses, transport protocol, and port numbers. We use a timeout value of 15 seconds.

Table 5.3: **OSN specific information: cookies and login/logout procedure**

OSN	session cookie	user cookie	login cookie	login `https`	logout `https`
Facebook	`datr=`	`cuser=`	`login_x=`	✔	✘
Hi5	`JSESSIONID=`	`Userid=`	`Email=`	✘	✘
LinkedIn	`bcookie=`	`leo_authi_token=LIM:`	n/a	✔	✔
StudiVZ	`PHPSESSID=`	`UserID1=`	n/a	✔	✘

header if set. These requests all include an active rr-pair in their referrer chain. However, not all rr-pairs include the OSN session cookie or a referrer; especially in requests to sites that are hosting static helper objects and scripts. We associate these with the last active rr-pair. This approach can lead to misclassification if the same IP address is involved in multiple parallel OSN sessions. However, if there are multiple concurrent ongoing sessions the likelihood that another user issues a request while the first user is still retrieving all embedded objects is low. If we are unable to find an action for an rr-pair, it is classified as *UNKNOWN*.

OSN specifics

The OSN-specific parts include a combination of relatively simple pieces (e. g., identifying the OSN sites and cookies) and some more complex classifications of the rr-pairs (e. g., profile analysis, specific login and logout determination). Section 5.2 lists the OSN-specific sites while Table 5.3 lists the specific cookie names that we use for each of the four OSNs.

The first step in the rr-pair classification is to determine if it is hosted on the main site. The next step is to determine if a rr-pair is active or indirect. Most requests not on the main site are indirect. However, not all other requests are active. For example, requests triggered automatically by AJAX should not be classified as active. To identify those, we use the manual traces (see Section 5.3.2) to develop specific patterns. For Facebook we initially consider all URIs with the following patterns as active: `.*\.php.*`, `.*/\$`, `.*/\?.*`. We then use specific patterns to exclude some URIs, e. g., `^/js_strings.php`, which enables word completion for input fields. We again rely on pattern matching to identify the action/click that caused the active rr-pair via a configuration file which lists the OSN-specific patterns. Table 5.4 gives example patterns for all OSNs.

At this point we have identified an action for each rr-pair. However, given that the number of actions is of the order of 200–300, we group actions into categories.

Table 5.4: **Examples of OSN specific patterns for action classification for home and sending a message**

OSN	category	action	method	pattern
Facebook	home	index	GET	^\/index\.php(\?(.*))*$
Hi5	home	home	GET	^\/friend\/displayHomePage\.do$
LinkedIn	home	home	GET	^\/home(\?*(.*))$
StudiVZ	home	start	GET	^\/Start(.*)$
Facebook	messaging	send msg	POST	^\/inbox\/(\?*(.*))$
Hi5	messaging	send msg	POST	^\/friend\/mail\/sendMail\.do$
LinkedIn	messaging	send msg	POST	^\/msgToConns(\?*(.*))$
StudiVZ	messaging	send msg	POST	^\/Messages\/WriteMessage(.*)$

We distinguish between the following categories: profile, photos, friends, home, offline, apps, messaging, search, video, groups, advertisement, osnspecific, other, and UNKNOWN. While most of these are obvious, note that home includes the login and logout rr-pairs, and osnspecific actions are, e. g., notes for Facebook and account migration for StudiVZ. Whenever the category of an active rr-pair is profile, we extract further details on profile accesses. Specifically, we count how often a user accesses his own profile, the profile of another user, and of how many other users. To account for changes in the OSN architecture we distinguish between actions that have been verified via a manual trace and rr-pairs for which the category is guessed given the knowledge about the structure of the OSN site and the URI.

5.3.2 Customization and validation approach

We need to be able to customize our approach to a set of specific OSNs and validate it. Therefore, we create *manual* traces for which we know the ground truth—by recording the actions while passively monitoring our interactions with the specific OSN. When collecting the manual traces we make a good faith effort to explore the feature set of the OSN that users might execute.

Manual traces enable us to identify the site names that belong to an OSN such that we can narrow our trace collection process to the relevant subset of traffic. We can also identify the various cookies that are used to track user sessions by the OSN, check if the OSN uses HTTPS for login/logout and identify corresponding handshakes for the state management, and construct the signatures and patterns to identify active rr-pairs that correspond to the user actions within the OSN. Manual traces also help in validating our approach. We claim to successfully characterize

Table 5.5: **Overview of manual traces.**

OSN	traces	size	actions	rr-pairs
Facebook	11	32 MB	344	5036
Hi5	6	50 MB	368	4413
LinkedIn	8	106 MB	411	6363
StudiVZ	11	27 MB	354	3990

a new OSN only when the analysis script can correctly identify the authenticated/offline OSN sessions and classify the active rr-pairs according to the user actions in manual traces. OSNs may reorganize their script architecture during our analysis. This may require readjusting the rr-pair classification patterns and new manual traces.

The different manual traces were collected via TCPDUMP on our local machine for each OSN. These included a trace that covered login, some basic actions, and logout (Section 5.1.2 discusses an example trace), one that covered all the actions offered via the initial menu, including profile, search, messaging, photos, ..., one that covers changing a user's global and privacy settings, one that registers a new user, and a trace that has a long break of over half an hour between basic actions. Finally, we included a trace that tests the state handling by including multiple login/logout steps; we used multiple browser windows and/or multiple browser instances in parallel to simulate activity of two different users from the same IP address. Table 5.5 summarizes these manual traces. Typically, our manual traces cover roughly 95 % of the actions recorded in the traces, see Section 5.4.2.

5.3.3 Validation

For our manual traces, we know exactly what and how many actions were performed in a session. Knowing this ground truth we applied our methodology to the manual traces; Figure 5.2 shows the inferred OSN usage for Facebook as a stacked barplot. For each category, we distinguish between verified and guessed requests and within each class of requests between active and indirect.

We observe a perfect agreement between the ground truth and our reverse-engineered user actions and their groupings into categories. In particular, Figure 5.2 shows that we have no rr-pairs in the category UNKNOWN. Also, there are no rr-pairs in either of the two subclasses "active–guessed" or "indirect–guessed". As such, although these classes are shown in the legend they do not show up in the stacked barplot. The plot shows a large number of indirect requests. This is not surprising given the large number of embedded rr-pairs. When monitoring actual user behavior, we can

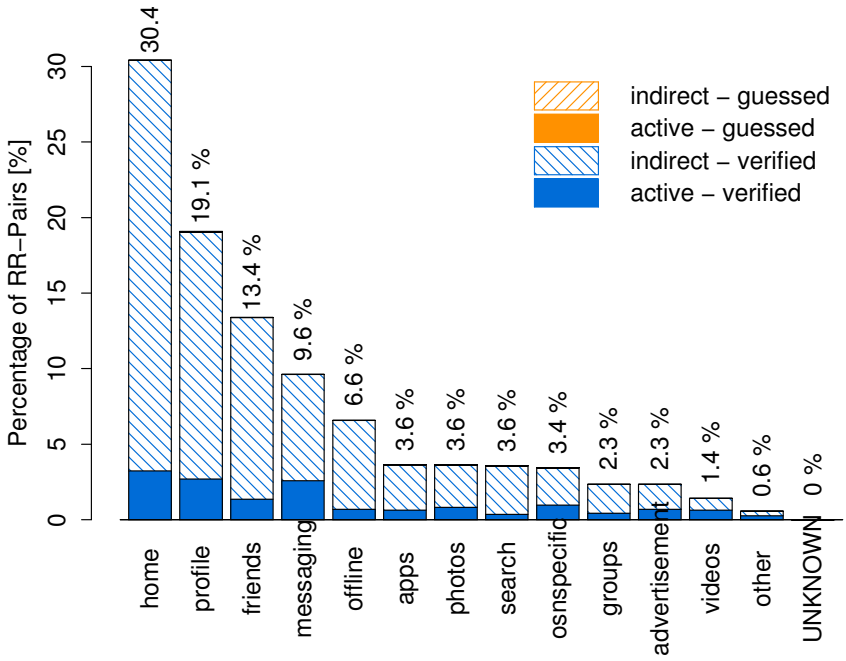


Figure 5.2: **Barplot of category popularity by rr-pairs for manual Facebook traces**

expect to see less of these indirect requests because they can be cached and users, contrary to us, usually do not flush their browser cache before visiting an OSN.

Manual inspection of the traces confirmed that all rr-pairs are correctly assigned to the appropriate category. The same findings (not shown) hold for the manual traces from Hi5, LinkedIn, and StudiVZ. However, some Web pages, e. g., `updatestatus.php` are used as both: active and indirect. That is why, some requests that are active are classified as indirect and the other way around. Since there are only a handful of these we are able to correctly identify 98 % of the requests as active or indirect.

5.3.4 Lessons learned

Starting from passive measurements it is possible to extract user clickstreams. However, we learned several lessons along the way:

- Reverse-engineering user interactions with OSNs from HTTP traces is non-trivial: OSNs differ in their software architectures, and the use of Web 2.0 features complicates matters significantly due to short and multiple interactions that differ qualitatively and quantitatively from “normal” Web traffic.
- We started the adaptation of our methodology with Facebook and then added LinkedIn and StudiVZ. Finally, we extended the capabilities to Hi5. With each

step the required customization time decreased since we were able to better isolate the OSN-specific elements. By now the major bottleneck is gathering the manual traces for validation (3–6 hours), the adjustment of the cookie and session handling including validation (2–4 hours), and the classification of the rr-pairs according to their features (2–6 hours depending on the feature richness and how intuitive the Web site is organized). Note, that this is a one-time cost per OSN.

- The number of patterns (see Table 5.4) needed for each OSN is relatively large. All in all, we have 253, 218, 206, and 299 patterns for Facebook, Hi5, LinkedIn, and StudiVZ, respectively. However, even if an OSN restructures its Web site they usually do so in small steps. For example, the reorganization of Facebook in 2008 only added around 50 patterns.

- If an OSN restructures its service the patterns have to be updated. Detection is via the drastic increase in the number of UNKNOWNs. This can be reduced by capturing new manual traces and then updating the patterns. For the Facebook update this only took about 1 hour as there were no major changes to the session handling which would have required collecting more complicated manual traces.

- Analyzing data from multiple ISPs, each with their own security mechanisms in place, requires careful synchronization of the analysis software to ensure comparability of the results.

- Given our experiences with OSNs it should in principle be possible to adjust the methodology to other WEB 2.0 sites. However, it is not possible to do the same kind of analysis for online shopping sites as these most likely use HTTPS rather than HTTP.

- The Tamper Data plug-in for Firefox turned out to be very useful to understand how an OSN is handling its sessions as it is able to display all requests while browsing even those sent via HTTPS.

5.4 Feature popularity

A typical question that is of interest to both OSN providers as well as ISPs, is which OSN features are so fascinating to the users that they spend so much time on the site. Does it differ across OSNs? In addition, both are interested in popularity shifts among features, e. g., from photos to videos, or the impact of novel features, e. g., live streaming. The ISP needs to care as it might impact bandwidth demand and the OSN needs to care as it might impact server resources. Therefore, we now explore the popularity of different features provided and supported by the different OSNs.

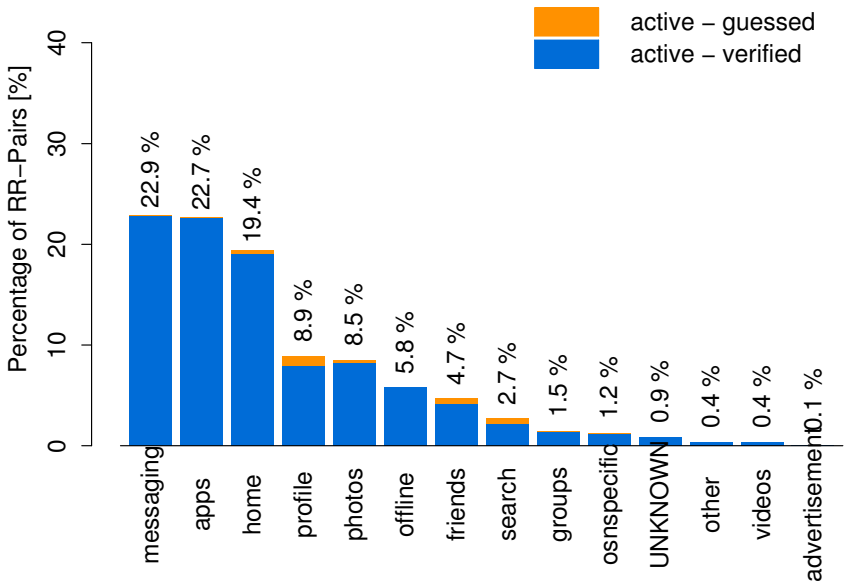


Figure 5.3: **Barplot of category popularity by active requests for Facebook in ISP-A2**

5.4.1 Clicks/active requests

Figures 5.3 and 5.4 show the histograms of the distribution of active rr-pairs (clicks) according to categories for Facebook for ISP-A2 and ISP-B3. We observe that the popularity of features differs by location. Within ISP-B, custom applications (apps) are more popular than within ISP-A, but the opposite is true for the profile category. This is consistent across the traces for the two ISPs. At the same time, we also note strong similarities: Messaging, home, profile, and photos are crucial categories at both locations. Comparing this with the relative popularity of features within Hi5 for trace ISP-B3 (see Figure 5.5), we see that for Hi5 users photos are more important. Similar observations hold for all traces at both ISPs. In addition, the profiles together with friends play a role in Hi5. Surprisingly, even though we selected OSNs whose primary content are user maintained profiles, the Facebook and Hi5 users' main interests are not on profiles as highlighted by their clickstream, which are spread among messaging, apps, photos, and eventually profiles.

While StudiVZ tried to clone Facebook, its users do use different features (plot not shown). Here, profile is the most common category in all ISP-A traces with more than 25 % of the active rr-pairs within ISP-A2. This is followed by home with 18 % and then by friends with 15 %. Within LinkedIn the focus on profiles and friends across all traces is even stronger. Within ISP-B3 we have 31 % of requests related

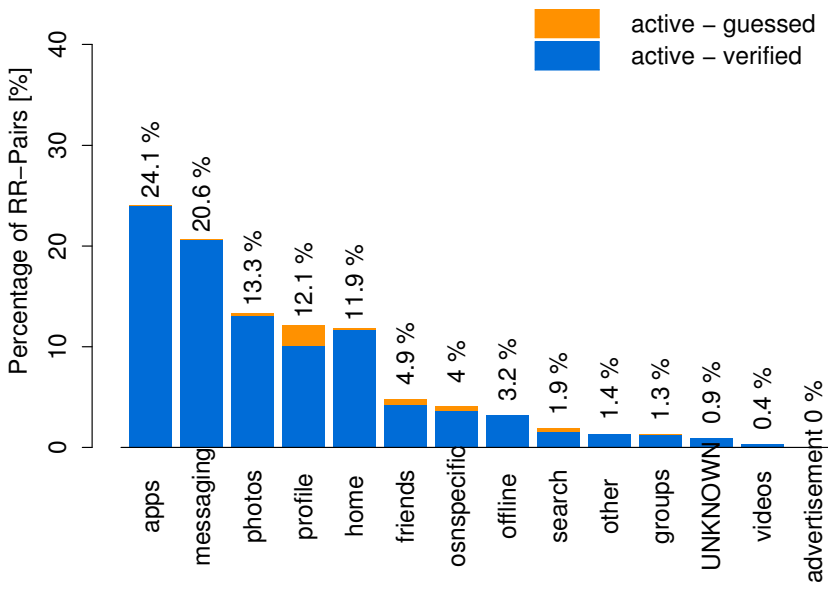


Figure 5.4: **Barplot of category popularity by active requests for Facebook in ISP-B3**

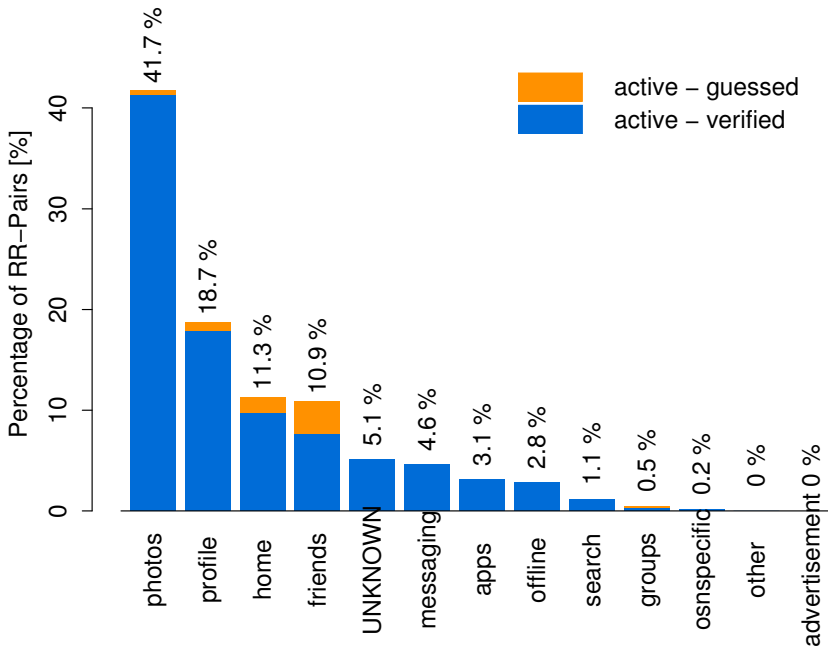


Figure 5.5: **Barplot of category popularity by active requests for Hi5 in ISP-B3**

to profiles and 22 % related to friends. It is intriguing to see that the relationship between requests in the profile and the friends categories are roughly the same for StudiVZ, Facebook, and LinkedIn. Note, that the percentage of UNKNOWNs is small.

5.4.2 All OSN requests

Figures 5.6 and 5.7 show the histograms of the distribution of all rr-pairs rather than active requests for Facebook and ISP-A2. Again, we see that the number of UNKNOWN rr-pairs is small—well under 5 %. Next, we note that the number of guessed rr-pairs, which are rr-pairs classified with a pattern that could not be verified using the manual traces, is also well below 3 %, except for LinkedIn where there are multiple guessed patterns in the friends category. Over all traces and both ISPs we find that the number of UNKNOWNs and guessed rr-pairs is each less than 7.5 %. Note the drastic difference in the distribution of the observed requests per category from the manual traces (recall Figure 5.2). This highlights how difficult it can be to approximate actual usage patterns of OSNs via active crawling, even though we acknowledge that our manual traces were collected for the purpose of exploring all OSN features.

Note how including the indirect requests drastically changes the relative importance of the categories. For Facebook, messaging and apps decrease by 50 % while home increases. Also, photos rather than profiles is now the top category in StudiVZ. Such shifts are important to keep in mind when partitioning OSN sites among different servers, e. g., one for handling the active requests (typically PHP scripts) and others for handling the embedded objects.

LinkedIn differs from the other OSNs in that it has maintained its strict focus on profiles. In LinkedIn (plots not shown) profiles and friends account for more than 50 % of the rr-pairs. Crosschecks for the other traces (not shown) show that this effect is not due to the different locations but rather the different focus of LinkedIn users.

To understand the impact of popularity shifts on traffic we have to compare the histogram of the request distributions to those of the byte contributions (see Figure 5.7 for Facebook and ISP-A2). Across all traces and all OSNs we see that the relevance of photos increases, as one might expect. Moreover, they contribute the most to the upload activity. The only other categories with some upload activity are apps for Facebook, profile for StudiVZ, and friends for Hi5. In addition, the relative importance of the guessed and UNKNOWN categories decreases even further for all OSNs. Extrapolating this, enabling the use of higher quality photos or even videos which contribute a tiny number of requests may change the bandwidth demand of OSNs quite drastically. Currently, they still contribute no more than 0.3 % of the

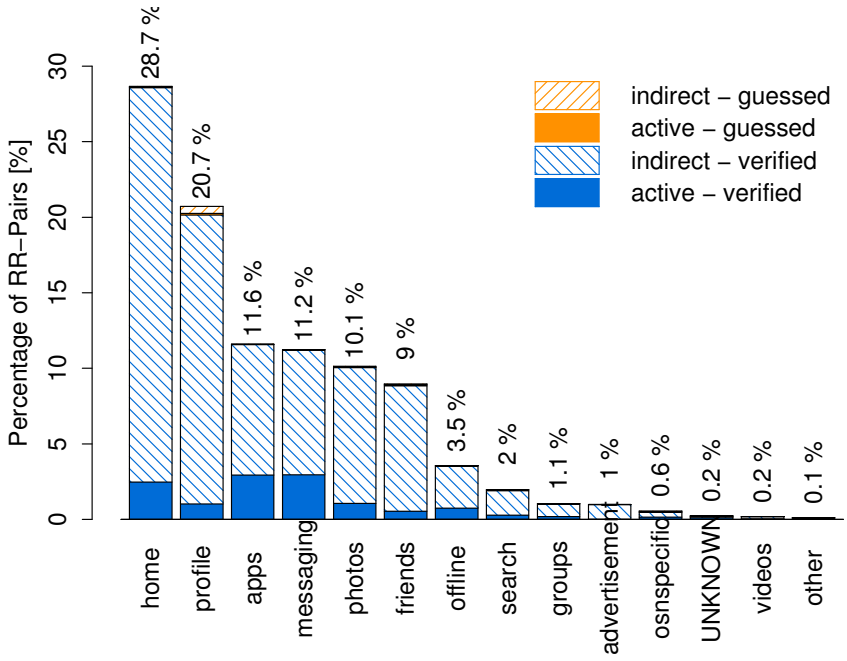


Figure 5.6: **Barplot of category popularity by rr-pairs for Facebook in ISP-A2**

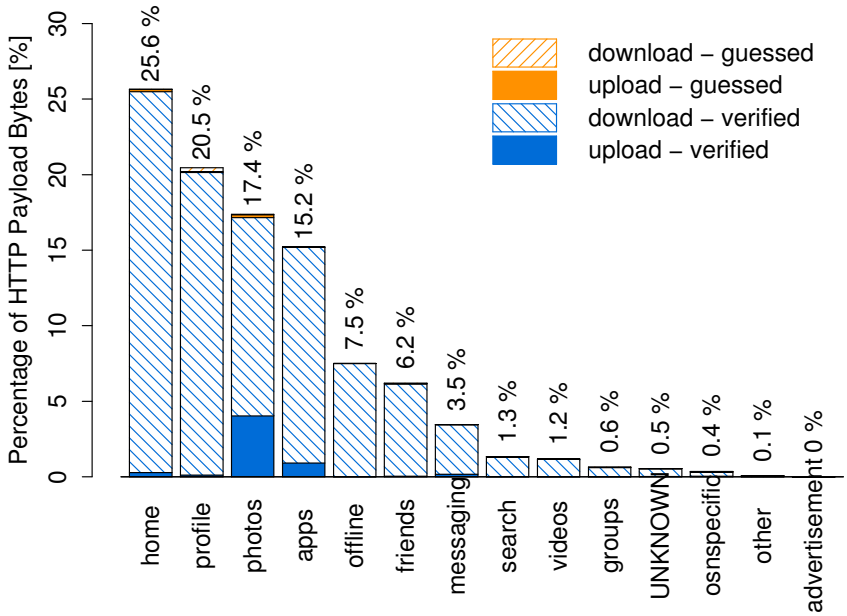


Figure 5.7: **Barplot of category popularity by bytes for Facebook in ISP-A2**

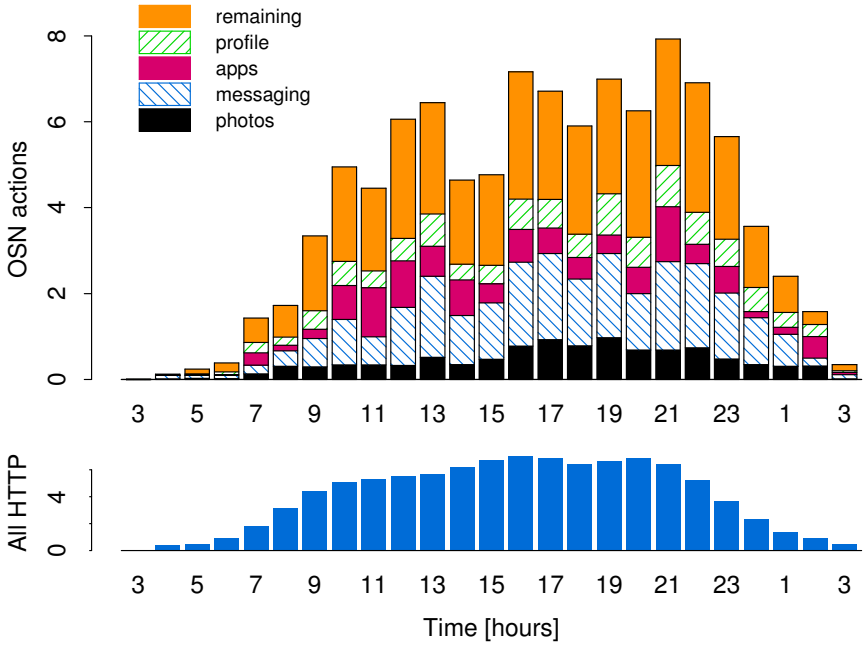


Figure 5.8: **Barplots of daily usage patterns for Facebook and all HTTP in ISP-A2: Facebook clicks (top) and total HTTP rr-pairs (bottom)**

requests within ISP-A but more than 1 % of the bytes. Recall, how the popularity of YouTube has increased the bandwidth demand.

5.4.3 Difference across time

To assess if OSN user activity differs from other HTTP activity we plot histograms of the relative frequency of rr-pairs and clicks in Figure 5.8. We see the expected time of day behavior for residential customers. We observe a similar trend for Facebook users. However, there are some notable differences. For example, messaging increases during lunch time and in the early evenings. Also, photos and apps are much more popular in the afternoon to early evening.

5.4.4 Differences between users

So far, we have only considered how the overall user population of an OSN behaves. Next, we ask if users of a given OSN behave in a similar manner or if some users only use one or two of the different features offered by the OSN. To this end, we rely

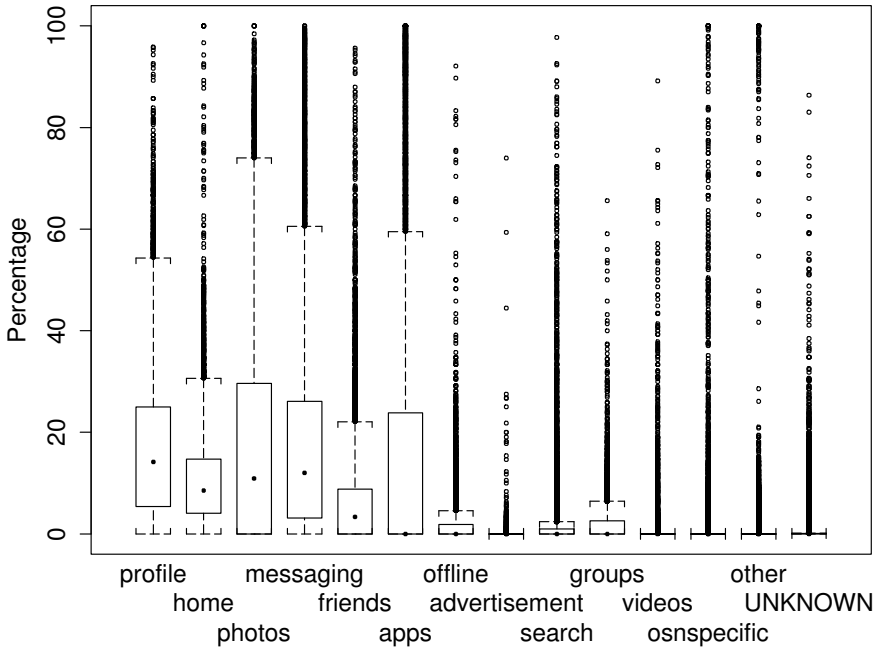


Figure 5.9: **Boxplot of category popularity by active requests per authenticated session for Facebook in ISP-B3**

on boxplots[3] for plotting the percentage of user requests for each category within an authenticated session. This means, a user that has 10 % and 90 % of his requests within home and profiles, respectively, will contribute one data point at 10 % for the boxplot for home and 90 % for the boxplot for profile.

Figures 5.9 and 5.10 show the resulting plots for authenticated subsessions with at least 20 active requests for Facebook for traces ISP-B3 and ISP-A2. We can see that among all the considered subsessions, the features profile, home, messaging, and friends matter. As captured by the whiskers and outliers, for some of the sessions, customer applications play a major role. More than 50 % of the users do not use any custom apps. But there is a substantial fraction which is almost exclusively focused on apps. We find that messaging is more crucial within ISP-B. The same observations hold for the other traces. In general, we note that some of the differences, e. g., with respect to apps usage, can be traced to the behavior of a few users while other differences, e. g., with respect to messaging, are caused by the overall user population.

Our previous observation that users use different features when interacting with dif-

[3] Boxplots are used to display the location, the spread, and the skewness of several data sets in one plot: The box shows the limits of the middle half of the data; the dot inside the box represents the median; whiskers are drawn to the nearest value not beyond a standard span from the quartiles; points beyond (outliers) are drawn individually.

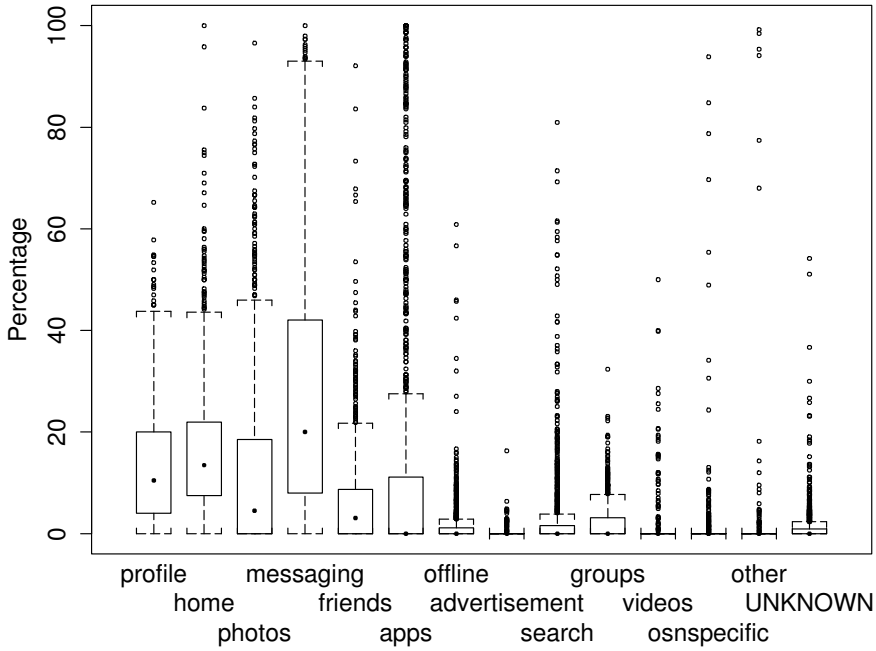


Figure 5.10: **Boxplot of category popularity by active requests per authenticated session for Facebook in ISP-A2**

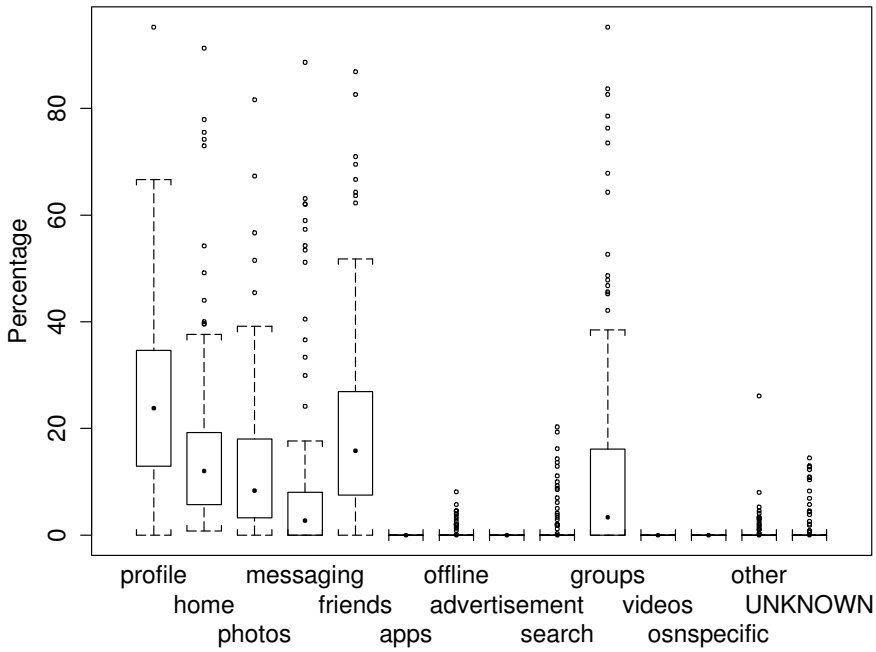


Figure 5.11: **Boxplot of category popularity by active requests per authenticated session for StudiVZ in ISP-A2**

ferent OSNs (Figures 5.3–5.5) also holds on a per session basis, e. g., see Figures 5.10 and 5.11 for StudiVZ and Facebook for ISP-A2. User actions of StudiVZ and Hi5 users (not shown) appear to be more homogeneous then those of Facebook users as indicated by many outliers in Figures 5.9 and 5.10 when compared to those in Figure 5.11. Over all traces and both ISPs we see that OSN users consistently use the popular features of the OSN such as profiles, photos, messaging, etc.

One implication of this analysis is that there are some typical OSN users that use the popular features of that OSN. However, OSN usage is also heavily influenced by some users who like a specific feature a lot, e. g., the application feature of Facebook or the groups feature of StudiVZ.

5.4.5 Profile usage

Lastly, we are interested in a further breakdown of a popular category such as profile. Whenever a user issues an active request for a profile we record if it is for his own, for another profile that is publicly accessible, or for an OSN internal profile. Figure 5.12 shows a stacked barplot of the relative number of accesses to the different profile categories. We note strong differences between the OSNs that we observed across the different traces. In Facebook, LinkedIn, and StudiVZ the majority of the requests are to profiles of friends. Only about 25–35 % are to the users' own profile. Within Facebook, about 10–15 % of the accesses are to public profiles and 20–25 % in LinkedIn. This feature is not available in StudiVZ and Hi5. One reason why LinkedIn might have a larger number of requests to public profiles is that some people use the public LinkedIn profile as their professional home page. Hi5 differs from the other OSNs in that most profile requests involve the users own profile. In part this is due to the way how Hi5 organizes its site. The profile page is a major component of the navigation.

The distribution across the different profile categories holds not just for the overall traces across the ISPs but also roughly within each user's OSN subsessions. This is shown in Figure 5.13 for Facebook and ISP-A2, where for a subset of the authenticated sessions[4], a stacked barplot is used to depict the number of accesses to a user's own profile, to public profiles, or to another user's internal profile. We see the skewed nature of the resulting profiles per session distribution. A few users access a lot of profiles (> 100) while others access only a few. The plot also shows that the distribution of profile accesses across the three categories roughly translates to the sessions themselves. This is also confirmed by the boxplot of the same data (not shown). We find that the variation across sessions with respect to the percentages of own, public, or internal profiles does not vary by much.

We next compare the number of profile accesses to the number of accesses to unique profiles. We find that while the average number of profile access is 6 for Facebook and

[4] We selected those with the largest number of profile requests per subsession.

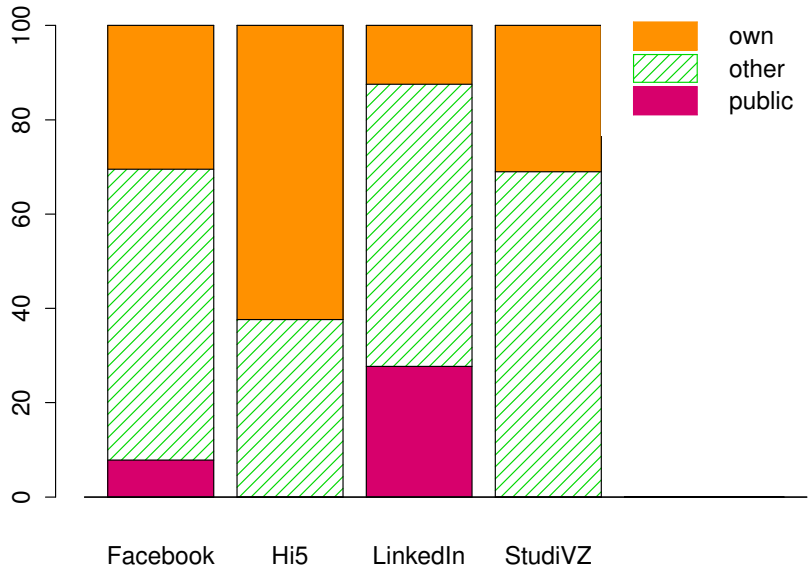


Figure 5.12: **Barplot of profile type popularity for all OSNs in ISP-A2**

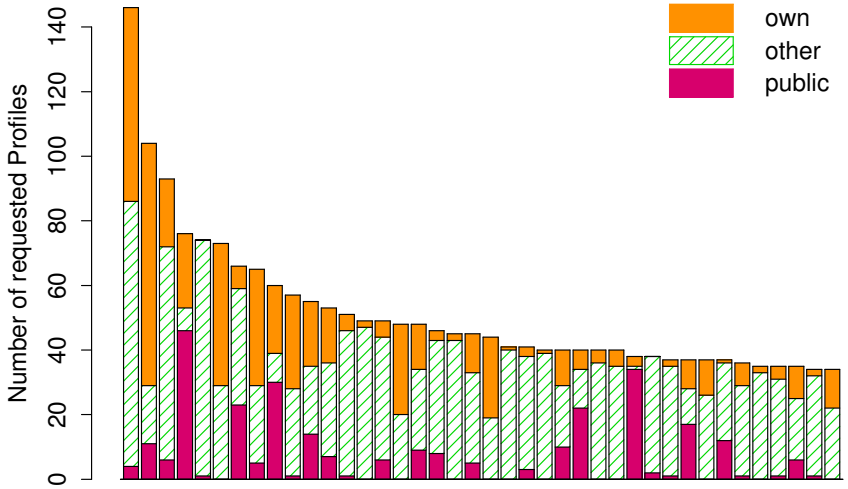


Figure 5.13: **Barplot of requested profiles per subsession for Facebook in ISP-A2**

ISP-A2 the average number of unique profile accesses is only 3. The median reduces from 3 to 2. Moreover, the distribution across the different profile classes changes to predominantly Facebook internal profile. This is highlighted by Figure 5.14 which again plots a stacked barplot for the subsessions with the largest number of unique profile requests. One contributor is that the user's own profile only counts once. Overall, these numbers are drastically lower than the size of the friendship graphs may indicate. Golder et al. [37] report that the mean/median number of friends is 144/180 while Joinson et al. [43] report 85/124.

5.5 OSN session characteristics

While OSNs are not yet universal they are rapidly adding users and change the way that users interact with each other. We now ask if their general traffic characteristics differ from other Web services in terms of top-level characteristics such as session sizes and durations. For example, this is useful for developing test cases for evaluating the performance of new P2P-based OSNs, such as PeerSoN [15].

5.5.1 Bytes per OSN session

We start by examining how much OSNs are contributing to the total traffic in terms of volume. We find that the contribution is still relatively small. Figure 5.15 plots the Cumulative Complementary Distribution (CCDF) of the number of bytes per OSN subsession for Facebook and StudiVZ for ISP-A2 on a log-log scale. All plots for all OSNs and both ISPs show that bytes per session are consistent with a heavy-tailed distribution and not with an exponential one. This implies that a small fraction of all OSN sessions is responsible for most of the bytes imposed on the network by this OSN. However, the tail of the distributions is by far not as heavy as those of all HTTP services. Indeed, we find that a Weibull distribution with a shape parameter of 0.5 yields a visually reasonable fit for Facebook. The tail of the sessions from all Web services are not well matched by a Weibull distribution. Here, a Pareto distribution with an α between 1.1 and 1.3 visually fits better.

The plots indicate and inspection of the other traces confirms that "heavy hitters" in Facebook impose the most load, followed by StudiVZ, and then LinkedIn users. We point out that a typical Facebook session size is between 200 KB and 10 MB, a typical StudiVZ session size between 50 KB and 5 MB, and a typical LinkedIn session size between 10 KB and 1 MB. Figure 5.15 highlights that there are some differences between the load imposed by authenticated vs. overall OSN sessions especially for StudiVZ. Typically, we see a shift in the probability distribution to a smaller number of bytes and sometimes a mode at OSN specific values, e. g., for StudiVZ at 10KB, occur. Note, that such volume demands are well within the capabilities of mobile data-service offerings.

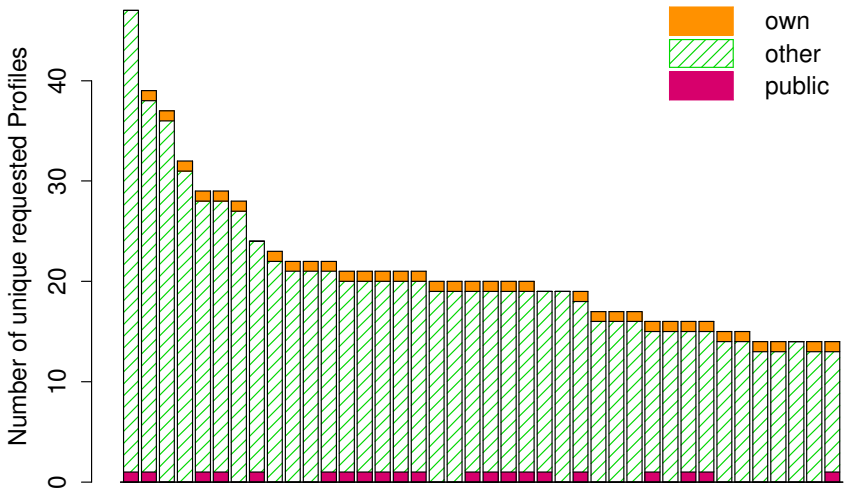


Figure 5.14: **Barplot of unique requested profiles per subsession for Facebook in ISP-A2**

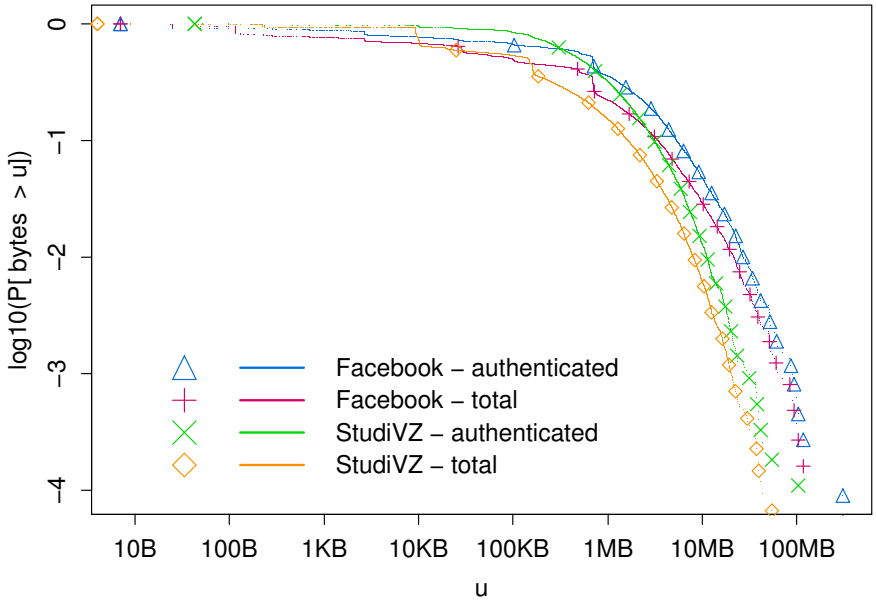


Figure 5.15: **CCDF of bytes per OSN subsession for Facebook and StudiVZ in ISP-A2**

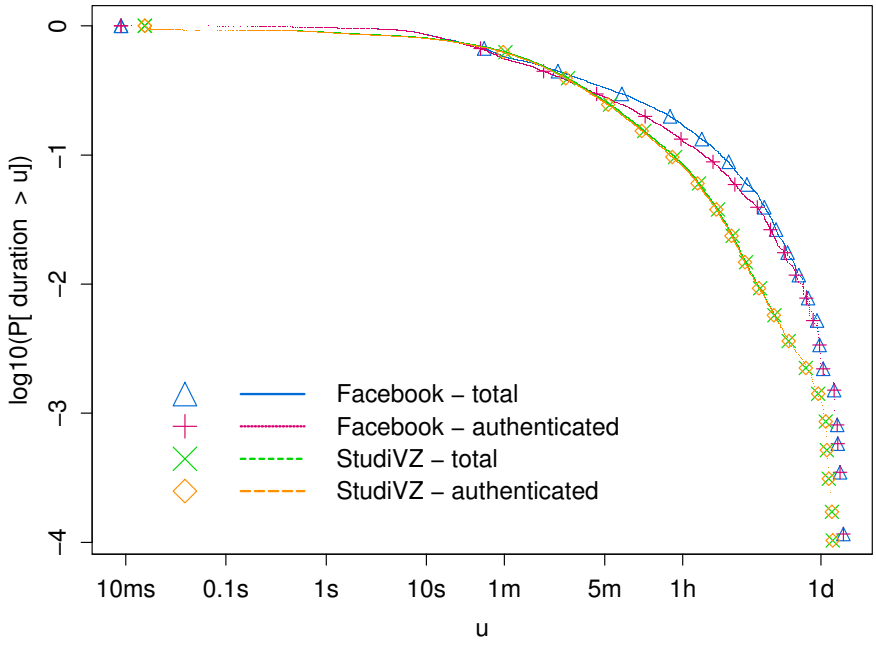


Figure 5.16: **CCDF of subsession durations for Facebook and StudiVZ in ISP-A2**

5.5.2 OSN session durations

Figure 5.16 plots the CCDF of the duration of OSN subsessions and authenticated sessions for Facebook and StudiVZ for ISP-A2, respectively, on a log-log scale. The plot shows that OSN sessions exhibit high variability, with many lasting a very short period of time and a few lasting for hours, with a mean of about 40 minutes for both, authenticated and total OSN subsessions. The mean durations for other OSNs and traces is roughly of the same order. Only the mean duration of Facebook usage at ISP-B is significantly longer. Note, the total OSN subsessions last just slightly longer than the authenticated OSN subsessions which is the case for both ISPs and all OSNs. This agrees with our expectation that users do not spend much time on the OSN site without logging in. Over all traces and both ISPs we find that the session duration distributions are not consistent with an exponential distribution, but have significantly heavier tails.

Figure 5.17 plots the Probability Density Function (PDF) of the logarithm[5] of the same durations as shown in Figure 5.16. We again see that the total duration is only slightly longer than the authenticated duration. Indeed, more than 10 % of the

[5] Coupled with a logarithmic scale on the x-axis, plotting the density of the logarithm of the data facilitates direct comparisons between different parts of the graphs based on the area under the curve.

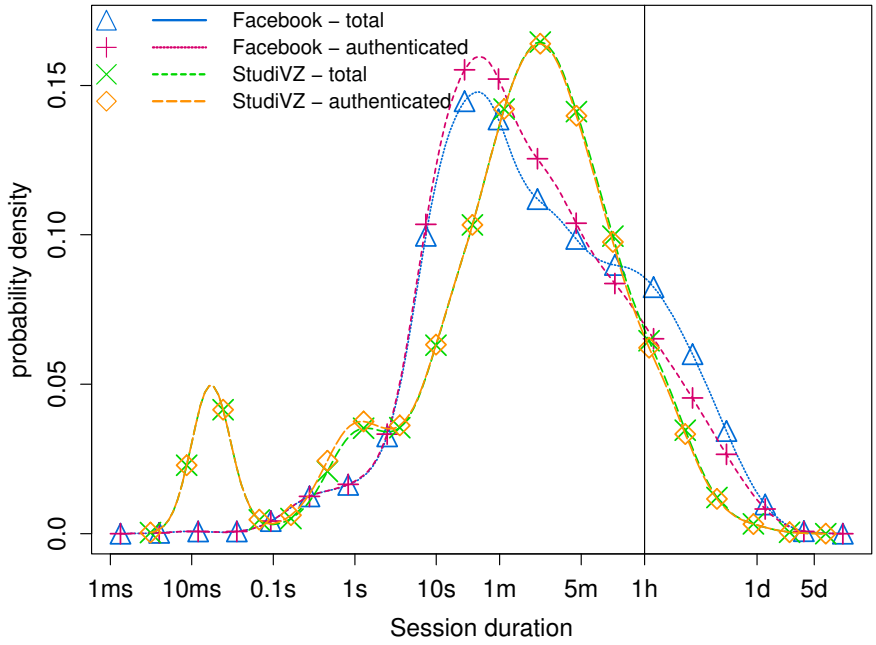


Figure 5.17: **PDF of subsession durations for Facebook and StudiVZ in ISP-A2**

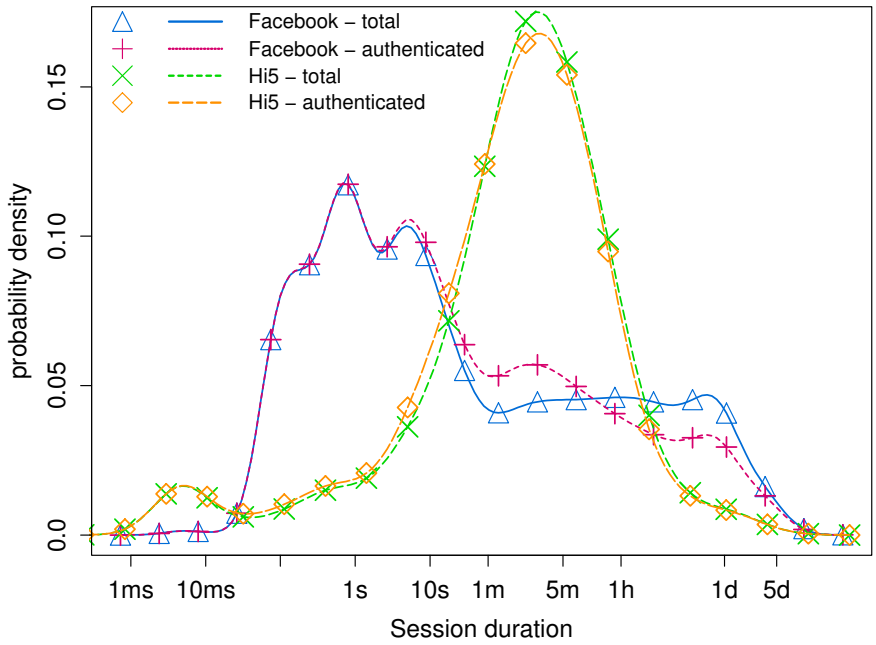


Figure 5.18: **PDF of subsession durations for Facebook and Hi5 in ISP-B3**

authenticated sessions and 12.5 % of the total sessions last longer than one hour. We also observe a peak between 5 sec and 2 min which appears to come from sessions where users only briefly check in with the OSN. Indeed, we find that most of them consist of only one or two actions. Comparing the durations of Facebook sessions to those of StudiVZ within the traces at ISP-A, we see there are more longer as well as shorter lasting sessions in Facebook than in StudiVZ. In Facebook we have a clear mode between 10 sec to 1 min while the mode is at 1–5 minutes for StudiVZ.

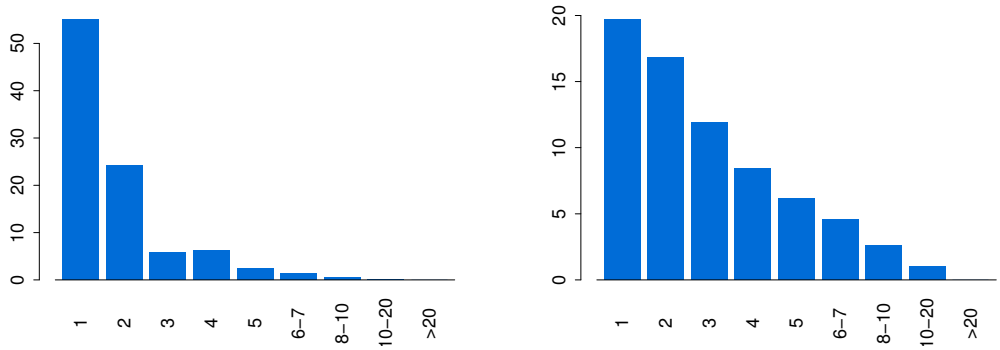
Figure 5.18 shows the PDF for Facebook and Hi5 for the 7-day trace ISP-B3. We again see that the usage of different OSNs by users of the same ISP differs. For Facebook we notice more shorter connections as well as many longer lasting ones than for Hi5 across all ISP-B traces. The session durations for LinkedIn (not shown) for all traces fall between those of Facebook and Hi5. Some Facebook sessions last as long as 24 hours and thus increase the mean duration for Facebook at ISP-B. But there is also a significant fraction between one to ten minutes.

When comparing the durations of Facebook across ISPs, i. e., by comparing Figure 5.17 with Figure 5.18, we see that within ISP-B Facebook has more shorter connections as well as many longer lasting ones. It appears, that users at this ISP stay active on Facebook for significantly longer time periods. We observe this difference across all traces. We also note the spike at 1 day—likely due to automatic renewal of IP addresses which ends the session. Maier et al [60] show that within ISP-A there is a large fraction of users that use idle disconnects[6]. This explains the shorter durations. On the other hand, the shape of the distributions for Hi5 and StudiVZ are similar even though these are different OSNs at different ISPs.

5.5.3 Number of subsessions within a session

Lastly, we explore how often a single user has multiple subsessions with an OSN. Figures 5.19(a) and 5.19(b) show histograms of the number of subsessions per anonymized IP address for Facebook within ISP-A2 and ISP-B3. We observe that it is common to have multiple sessions per IP. This can have multiple reasons, e. g., multiple computers using a single DSL line via NAT box; multiple users using the same computer; and reuse of IP addresses to different DSL subscribers. Within ISP-A we see a smaller number of subsessions per anonymized IP address than in ISP-B. Given that IP addresses are assigned dynamically one can expect the number of sessions per IP address to increase with the length of the trace. We, for example see this across OSNs within ISP-B when we compare results from ISP-B2 to those of ISP-B3. Nevertheless, this effect does not fully explain the difference between Figures 5.19(a) and 5.19(b). There are also differences between OSNs. For example,

[6] An idle disconnect is a feature of home routers (NAT boxes) that disconnects its DSL connection when the user is idle for some time. Usually, such routers automatically reconnect the DSL connection whenever there is new activity. This usually implies a change in the IP address for the line.



(a) Subsessions per IP: ISP-A2 (b) Subsessions per IP: ISP-B3

Figure 5.19: **Histograms of authenticated Facebook subsessions per IP address in ISP-A2 and ISP-B3**

at ISP-A there are more sessions per IP address for Facebook than for StudiVZ in both traces. Moreover, at ISP-B Hi5 has a higher likelihood of two sessions than Facebook across all traces at ISP-B.

To differentiate between multiple subscribers and multiple users, we also checked how many subsessions a single session (using the same session cookie) has. While there are many sessions that have only a single subsession, there are a number of users (as identified by the session cookie within Facebook) that repeatedly login/logout from Facebook. Indeed, 15 % of all Facebook sessions recorded at ISP-A include more than 3 subsessions. In addition, we have observed a session with more than 387 subsessions. Within ISP-B2 and ISP-B3 a single IP address which has many authenticated sessions skews the results. For other OSNs we see a smaller number of subsessions per session, i. e., for Hi5 and LinkedIn we see less than 5 % of the sessions with more than 3 subsessions. Unfortunately, this kind of analysis cannot be extended to StudiVZ as StudiVZ assigns a new session cookie for each authenticated subsession.

5.6 Dynamics within OSN sessions

Next, we delve into the OSN sessions and ask how users behave within a session. This is crucial for deriving detailed models for OSN evaluations.

5.6.1 Active vs. inactive time

In principle, we notice that the durations of Facebook sessions are longer than those of other OSNs. We next examine if the users are actually continuously interacting

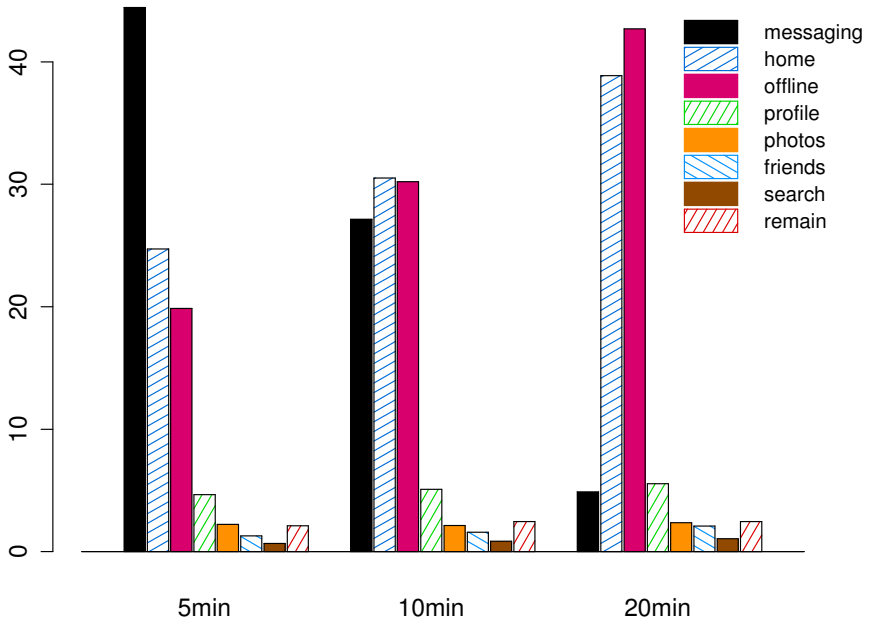


Figure 5.20: **Barplot of action popularity after inactivity periods for Facebook in ISP-A2**

with the OSN or if this duration is an artifact of the session handling. Therefore, we compute *OSN action sequences* by grouping all actions from the same OSN session as long as the time between actions never exceeds a timeout value of 5 minutes. Then, we use this information to calculate the percentage of time users are active on the OSN and the percentage of time they are inactive. An inactive user is authenticated with the OSN but is currently not interacting with it.

When we consider all sessions users are typically active during the whole session. If we only consider those sessions that last more than 1 minute, only 50 % of the users remain active the whole time. Once we consider only those sessions that last for at least 40 minutes we find that hardly any users continuously interact with Facebook.

We notice that during active periods, Facebook users usually do not visit any other sites. Only 7 % of all sessions for ISP-A2 visit other sites. Among the most popular domains are `msn.com` and photo community sites. While the users are inactive on Facebook, sites such as Google, YouTube, and Apple are popular. Other users additionally visit other OSNs, news sites, the ISP home page, etc.

Next, we explore the features with which users resume their usage of Facebook after a period of inactivity. Figure 5.20 shows the relative histograms for inactivity breaks of length 5, 10, and 20 minutes. The distribution changes drastically. After

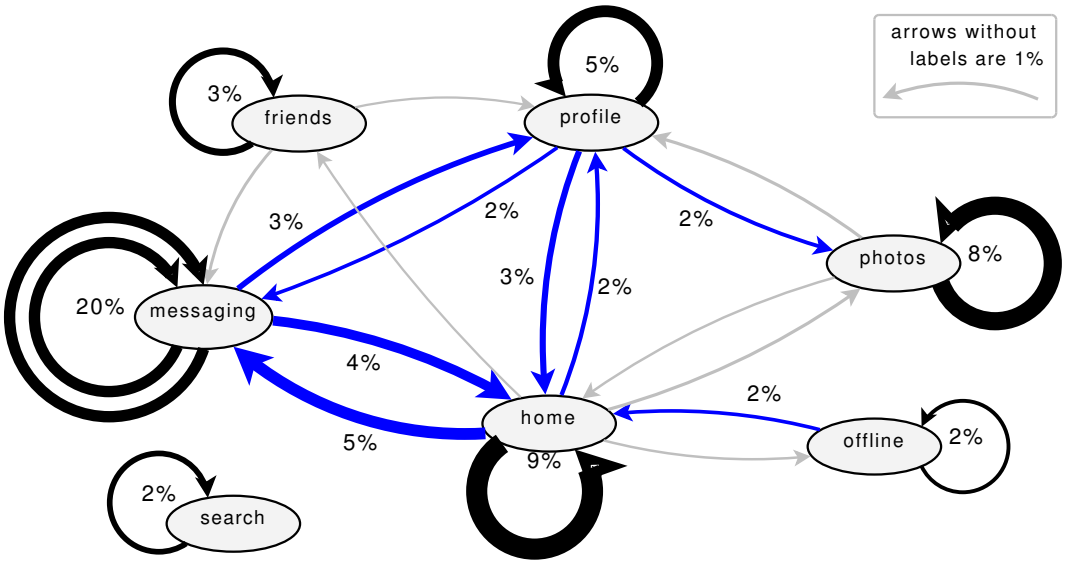


Figure 5.21: **Category transitions of click sequences for Facebook in ISP-A2 (only transitions > 1 % are shown)**

a short break, messaging is dominant. However, after a 10 minute break, home and offline take over in importance. This trend continues if we consider 20 minute breaks. However, for longer breaks the plots stay the same as for 20 minutes. In addition, we note that the relevance of photos, profile, search, and friends also increases slightly.

5.6.2 Feature sequences

Given the above observation regarding which features are popular after a break, we now explore the relative popularity of feature sequences within the user clickstreams for Facebook sessions with at least ten active requests. The corresponding transition diagram is shown in Figure 5.21 for trace ISP-A2. We find that the prominent feature categories, seen in Figure 5.3, are also dominant here. A few categories account for 50 % of the transitions and they indicate that users tend to stay within their feature category, i. e., from messaging to messaging (20 %), from home to home (9 %), from photo to photo (8 %), and from profile to profile (5 %). However, it is also interesting to see that from home, messaging is the most likely category followed by profile. From profile, the users switch to home, photos, or messaging.

For StudiVZ we see a similar trend—users tend to stay within their feature categories. But for Facebook the dominant features are profiles (9 %), friends (7 %), and photos (7 %). The transitions between features are from friends to profiles (6 %) and from profiles to home, photos, and friends (4–6 %).

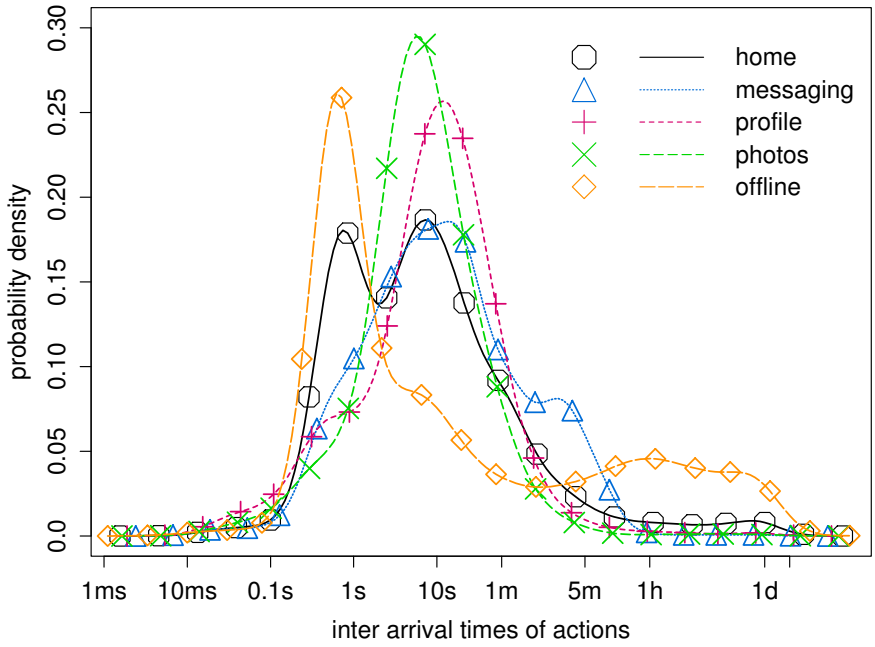


Figure 5.22: **PDF of inter-action times within Facebook categories in ISP-A2**

Then, we explore how much time a user needs for such transitions. Figure 5.22 shows a density of the inter-arrival time for selected feature groups. We focus on home, messaging, profile, photos, and offline as they are among the most popular features and their inter-arrival time distributions differ. The distributions for offline differs the most from the others. This is not that surprising given the previous discussions about periods of inactivity. Messaging also has some larger inter-arrival times. One possible explanation is that users need some time to compose their texts. Profiles typically have shorter inter-arrival times and photos are still shorter. A user is likely to stay within the same category while exploring different profiles/photos. For photos this is usually simplified by photo album features. Home has a bimodal distribution. The very short inter-arrival times are likely due to triggered actions. The other mode is likely due to its use as navigation site.

5.7 Related work

In the past, social network analysis was the domain of sociologists and anthropologists [110]. Their typical tools are surveys and interviews which have the drawback that they can usually only capture a small user base. Nowadays, with the advent of Online Social Networks, the networking community is capable of gathering large-

scale data sets from OSNs, e. g., by crawling the OSN, by passively monitoring user interactions, or by collaborating with an OSN operator.

In this chapter, we examined actual traffic across multiple OSNs, both at a macro as well as at a micro-level, to understand user behavior. Therefore, our work is related to the efforts that rely on surveys [2, 26, 101] and interviews [43]. Our findings regarding the feature popularity match well with the findings of Joinson [43]: OSN users focus on messaging, browsing profiles, and sharing of photos. Hence, one can conclude that the users impression about how they use OSNs agrees with how they actually use them. In addition, we have a significantly larger user base, explore multiple OSNs, and explore within session characteristics.

Other efforts use network traces to characterize individual OSNs or individual features of OSNs. For example, Gill et al. [34] study patterns of access to YouTube from a campus perspective. Zink et al. [122] also study YouTube using passive traces from a campus network. They explore the popularity of video clips and show that local and global popularity of video clips differ which supports the case of local caching. Nazir et al. [69] focus on a specific feature of Facebook: Third party applications. By offering different kinds of applications on their own servers they are able to monitor and characterize their usage. By further studying the interactions between Facebook, external applications, and the OSN users Nazir et al. [70] identify some potential performance bottlenecks within the Facebook server infrastructure. Using network data of an extended time period from Facebook messages and pokes sent by college students Golder et al. [37] are able to characterize another piece of Facebook: The messaging activity. We in contrast focus on the larger picture and can examine transaction sequences.

By collaborating with specific OSN operators Chun et al. [19] are able to compare the structural characteristics of the activity network with the friends network relying on guestbook logs from Cyworld. Kumar et al. [49] explore how path properties, including diameter and density, of the social network change over time for Flickr and Yahoo 360! based on timegraphs from these networks. As such, both of these studies focus on the graph properties rather than on how users use the OSN.

A few studies have tried to explore the differences and commonalities between OSNs. For example, Backstrom et al. [9] study how groups form in social networks across LiveJournal and DBLP; Kumar et al. [49] explore how path properties, including diameter and density, of the social network change over time for Flickr and Yahoo 360!; and Mislove et al. [65] explore the degree and cluster coefficient of the embedded networks and confirm the power-law, small-world, and scale-free properties of Online Social Networks for Flickr, LiveJournal, YouTube, and Orkut. While the previously mentioned studies focus on the graph properties of the online communities Krishnamurthy and Wills [48] characterize privacy settings and their usage across Facebook, MySpace, Bebo, and Twitter while Cha et al. [16] study how the popularity of video content changes with the age of the content for YouTube and

Daum UCC. However, all of these studies can only study the static relationships in the OSN rather than how the users actually interact with the OSN.

By crawling specific OSNs other studies have focused on the topology of the OSN. For example, the study by Mislove et al. [64] focuses on the growth of the Flickr's network and if it adheres to the preferential attachment property. This is complemented by Cha's work on social cascades in Flickr [17] and Liben-Nowell's work on the relationship between geography and online friendships in LiveJournal [56]. Gjoka et al. [35] crawl Facebook user profiles to study high-level characteristics of application users and the growth patterns of applications. Lampe et al. [52] crawl Facebook to determine the popularity of specific profile elements. We in contrast examine how many profiles in general and how many unique profiles are actually accesses by users within an OSN session.

Recent work has focused on understanding how users interact with the OSN or with other users using crawls of the OSN sites. For example Viswanath et al. [109] repeatedly crawled the walls of specific Facebook users and used the differences to determine communication patterns. Valafar et al. [107] examine interactions within Flickr between the photo owners and their fans. Torkjazi et al. [105] find that a large fraction of the MySpace user IDs are inactive or deleted by inspecting the last login time.

In concurrent work, Benevenuto et al. [12] also analyzed clickstream data to develop models of OSN user behavior. Their data source is a Brazilian social network aggregator. Their analysis focuses on Orkut as it is the most popular OSN within their data set. Overall, their study identifies similar trends to ours.

5.8 Summary

We successfully reconstructed OSN clickstreams from anonymized HTTP header traces obtained from passively monitored network traffic with tens of thousands of users at different ISPs. We present a customizable methodology for identifying OSN sessions and user actions within the OSN. We apply our methodology to four OSNs: Facebook, Hi5, LinkedIn, and StudiVZ. Our methodology enables us to extract OSN usage information across a wide range of features, from coarse information like session duration to minute details about the kinds of profiles the user accesses. For example, we find:

- Users tend to stay within the same activities (feature category). Moreover, we find that users are "trapped" in some categories, e. g., photos and messaging.
- While user sessions can be quite long (typically > 30 minutes), we find, e. g., for Facebook, that for long sessions (> 10 minutes) users do not continuously interact with the OSN.

- While we selected the specific set of OSNs based on the criterion that they feature profiles, they are the most popular feature only within LinkedIn and StudiVZ. With regards to transfer volume, photos are currently the most important category, although the volume is significantly lower than that of typical Web sessions.

We are able to identify the features that are important to the users and point out differences from other Web services. In addition, we gathered some insights on how to generate workloads for evaluating novel OSNs.

The next steps involve customizing our methodology for a larger set of OSNs. Moreover, we are planning to dig even deeper into the intra-session characteristics. In addition, we are starting a collaboration with researchers in the social sciences to better understand the implications of our observations.

Chapter 6

A First Look at Mobile Hand-held Device Traffic

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 Wi-Fi capabilities. Their increased graphics and processing powers make these devices all-in-one PDAs and media centers. Contemporary MHDs can be used to surf the Web, check email, access weather forecast and stock quotes, and navigate using GPS based maps—to 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 [119].

In this chapter, we cast a first look at Internet traffic caused by mobile hand-held devices. We use anonymized residential DSL broadband traces, spanning a period of 11 months, to study their behavior and their impact on network usage. We are thus able to observe behavior of MHDs when they are connected via Wi-Fi at home and compare their traffic patterns to the overall residential traffic characteristics. Some devices (most notably iPod touch and iPhone) require Wi-Fi connectivity rather than cellular connectivity for certain services, such as downloading large files. 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 focus in this chapter only 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 6.1 we present our data sets and methodology, Section 6.2 presents our results. In Section 6.3 we discuss related work before we conclude in Section 6.4.

Table 6.1: **Overview of anonymized 24 hours packet traces.**

				MHD HTTP Traffic	
Name	Start date	Size	# MHDs	Volume	% of HTTP
ISP-A2	Thu 18 Sep '08 4am	>4 TB	>200	>2 GB	0.1 %
ISP-A3	Wed 01 Apr '09 2am	>4 TB	>400	>9 GB	0.4 %
ISP-A4-a	Fri 21 Aug '09 2am	>6 TB	>500	>15 GB	0.6 %
ISP-A4-b	Sat 22 Aug '09 2am	>5 TB	>500	>15 GB	0.7 %

6.1 Methodology

In this section we describe the anonymized data sets of residential DSL connections and our methodology for analyzing them.

6.1.1 Data sets

We base our study on multiple sets of anonymized packet-level observations of residential DSL connections collected at aggregation points within a large European ISP. The monitor, using Endace monitoring cards, operates at the broadband access router connecting customers to the ISP's backbone. Our vantage point allows us to observe more than 20,000 DSL lines. The anonymized packet-level traces are annotated with the anonymized DSL line card port id. This enables us to uniquely distinguish DSL lines since IP addresses are subject to churn and as such cannot be used to identify DSL lines [60]. While we typically do not experience any packet loss, there are several multi-second periods (less than 5 minutes overall per trace) with no packets due to OS/file-system interactions.

We use several 24 h traces collected over a period of 11 months which gives us the the opportunity to track mobile device usage changes over time. Table 6.1 summarizes characteristics of the traces, including their start, duration, size, and number of observed MHDs. We note that while the number of observed DSL lines is about the same in each trace, the number of observed MHDs has increased significantly.

The data anonymization, classification, as well as application protocol specific header extraction is performed immediately on the secured measurement infrastructure using the Bro NIDS [78] with dynamic protocol detection [25].

6.1.2 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 the MHD behavior in a controlled environment. These manual traces include both, the action that the user performed and the resulting packet substream.

Among the currently popular MHD devices are Symbian based phones, BlackBerrys, iPhones and iPods, Windows Mobile based phones, and Google Android phones [113]. 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.

Given that multiple devices can be sharing a single IP address we have to utilize other information to identify MHDs. We rely on user-agent strings and IP TTLs (see Section 6.1.3). More specifically, we take advantage of HTTP user-agent strings, as HTTP dominates the protocol mix for MHD devices and most mobile applications, including weather and stock quotes, use HTTP.

From our manual traces we extract a list of HTTP user-agent strings for each device and OS combination. 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. Yet, since most rely on standard libraries, we can add patterns for these. 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 end of user-agent strings. Since the version numbers used by the iPhone OS and Mac OS X are disjoint we can distinguish these. 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.

Our analysis of the manual traces also reveals that Apple devices send DNS requests for `iphone-wu.apple.com` occasionally and subsequently connect to this site using HTTP. This site seems to be related to YouTube videos. Starting with iPhone OS 3 Apple devices also resolve and connect to `configuration.apple.com`.

[1]Manual trace collection was performed on 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.

6.1.3 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. 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 6.2.1 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 OSs. 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 OSs. While we cannot distinguish iPhones/iPods from Macs usage or Windows Mobile from Windows usage we can use IP TTLs to separate the other combinations. Our observations show that the majority of home OSs 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[2] 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 now use Bro's DPD [25] 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.

6.2 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 our results on iTunes and AppStore usage.

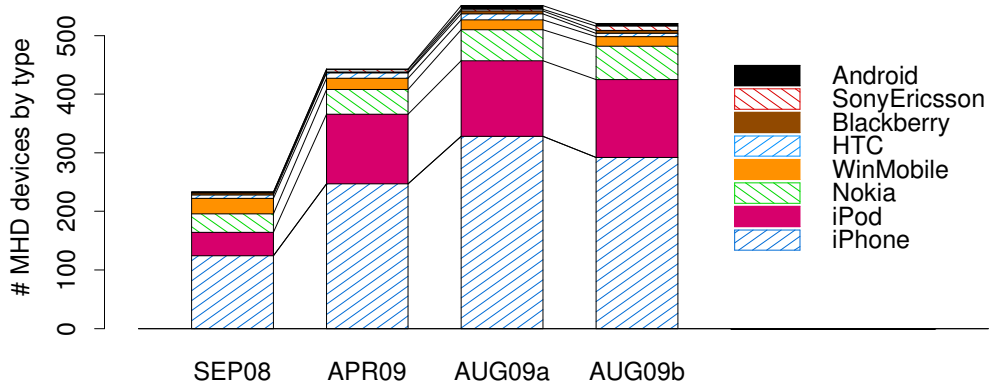


Figure 6.1: **Barplot of MHD device type popularity**

6.2.1 MHD pervasiveness

On a significant number of the DSL lines we observe traffic from MHDs (see Table 6.1). Indeed, in the most recent trace, ISP-A4, 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 % [28, 60].

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 ISP-A2 and ISP-A4. The HTTP traffic volume from MHD grew sixfold while the overall traffic volume increased only slightly and the HTTP volume increased by 22 % at our vantage point.

Figure 6.1 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 [113]. One possible explanation is that Apple users *(i)* find their device very convenient even for home use *(ii)* are looking for a multimedia device that "also works as a phone". Indeed, the iPod Touch is basically an iPhone without phone capability. We note that starting from ISP-A3 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 time. 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

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

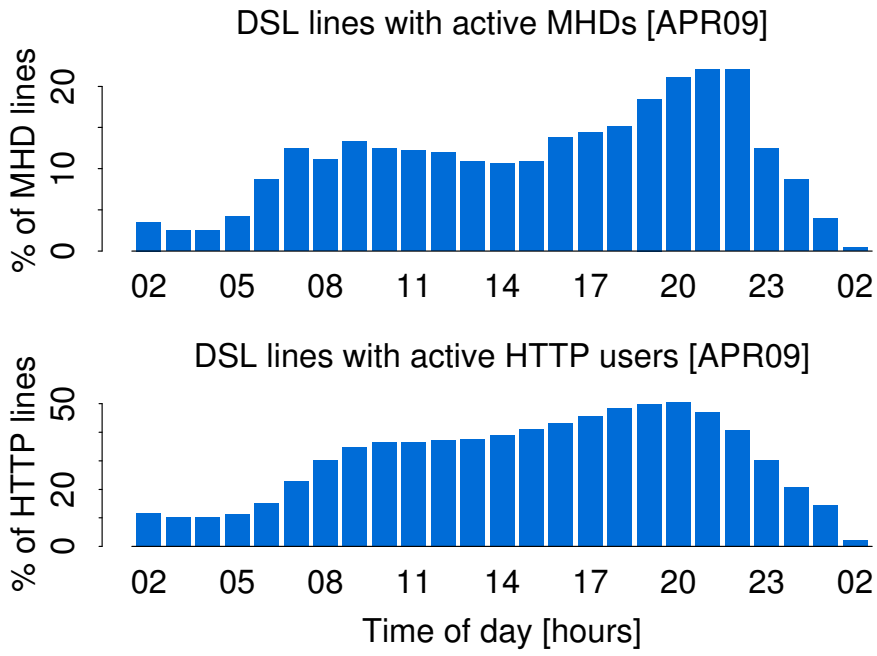


Figure 6.2: **Histograms of active lines per hour in ISP-A3: Lines with MHD activity (top) vs. lines with HTTP activity (bottom)**

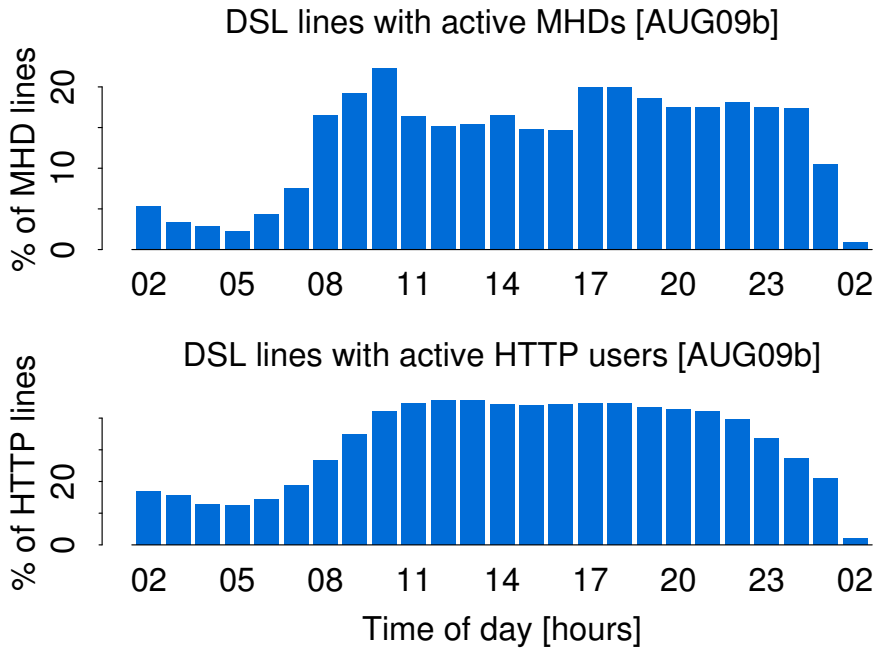


Figure 6.3: **Histograms of active lines per hour in ISP-A4-b: Lines with MHD activity (top) vs. lines with HTTP activity (bottom)**

HTTP traffic per hour for ISP-A3 and ISP-A4-b in Figure 6.2 and Figure 6.3. 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 (ISP-A4-b 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, the weather, or even their calendar 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 at the same time or whether they are used contemporaneously. To asses 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. Using this information and timeouts of one second, one minute, and five minutes we compute the number of DSL lines with mixed activity (MHD and non-MHD). We find that 33–39 % of MHD lines exhibit mixed MHD/non-MHD activity with IRTs of less than one second. For IRTs of less than one minute (five minutes) up to 62 % (72 %) of the lines have mixed activity.

6.2.2 Application protocol mix

While our approach for analyzing the application protocol mix of MHDs is limited (see Section 6.1.3, 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 ISP-A2, 2.3–2.5 % in ISP-A3 and ISP-A4-a. However, it drops to 0.2 % in ISP-A4-b 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 ISP-A3, and 15 % RTMP streaming in ISP-A4-a, caused by only a handful of MHDs.

6.2.3 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 6.1.2.

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, see [60]. We then group these into a handful of categories. Application downloads from Apple's AppStore are compressed ZIP objects

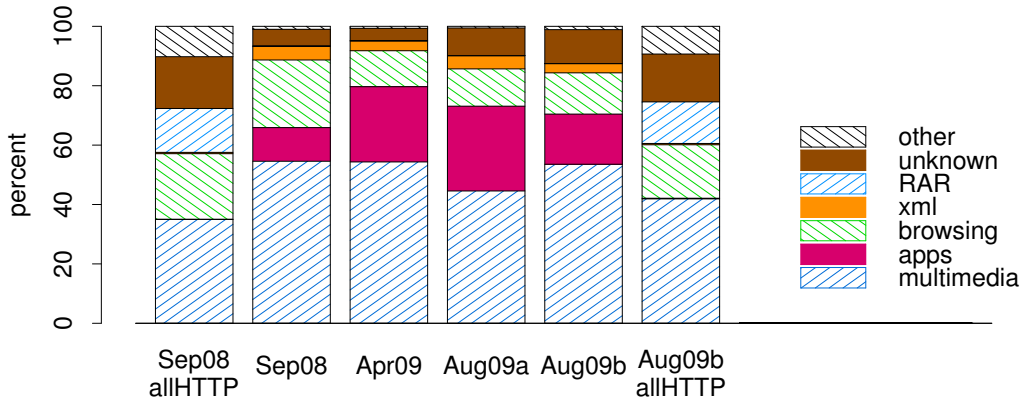


Figure 6.4: **Barplot of content-type popularity for MHD and all HTTP traffic grouped by category**

with the file extension .ipa. These are classified as application downloads. We classify video and audio content as multimedia and images as Web browsing since the latter are usually integral part of Webpages.

Figure 6.4 shows the HTTP content type categories for MHDs and compares them with all HTTP traffic. We find that multimedia (audio, video) content is the most voluminous MHD content-type across all traces followed by application downloads. Interestingly, XML objects are also common. They account for 2.5–4.6 % of the transfered 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 ISP-A2).

Comparing these results to all HTTP traffic [60] 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.

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 ISP-A4-a 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 HTTP traffic by MHD differs from overall HTTP traffic we compare the distribution of HTTP object sizes. See Figure 6.5 and Figure 6.6 for a plot of the Cumulative Complementary Distribution (CCDF) and Probability

Density Function (PDF) of the logarithm of the data on a logarithmic x-axis. for the ISP-A3. The results for the other traces are similar. We find that both distribution are consistent with a heavy-tailed distribution (see Figure 6.5). While the dominating mode of objects sizes downloaded by MHDs is larger (see support lines in Figure 6.6) the tail is heavier for all HTTP traffic.

6.2.4 Mobile applications

Figure 6.7 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, given our own usage, 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 e. g., 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 (along with the rest of the VZ family: MeinVZ and SchuelerVZ). 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.

6.2.5 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 6.2), e.g., more than 70 % of downloads for the 2009 traces. Surprisingly, we see that for ISP-A4-a and ISP-A4-b the 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 revealed 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

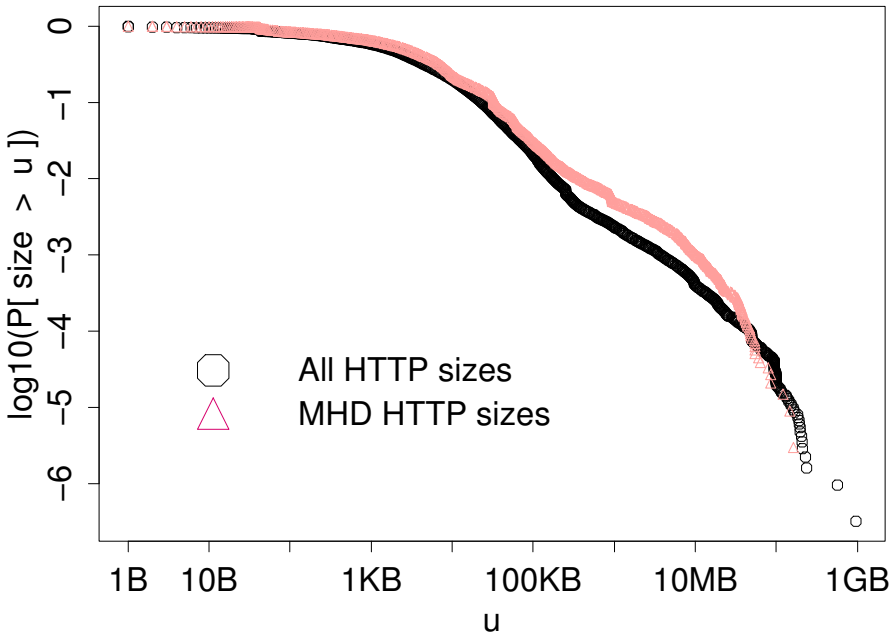


Figure 6.5: **CCDF of HTTP object sizes for all and MHD traffic in ISP-A3**

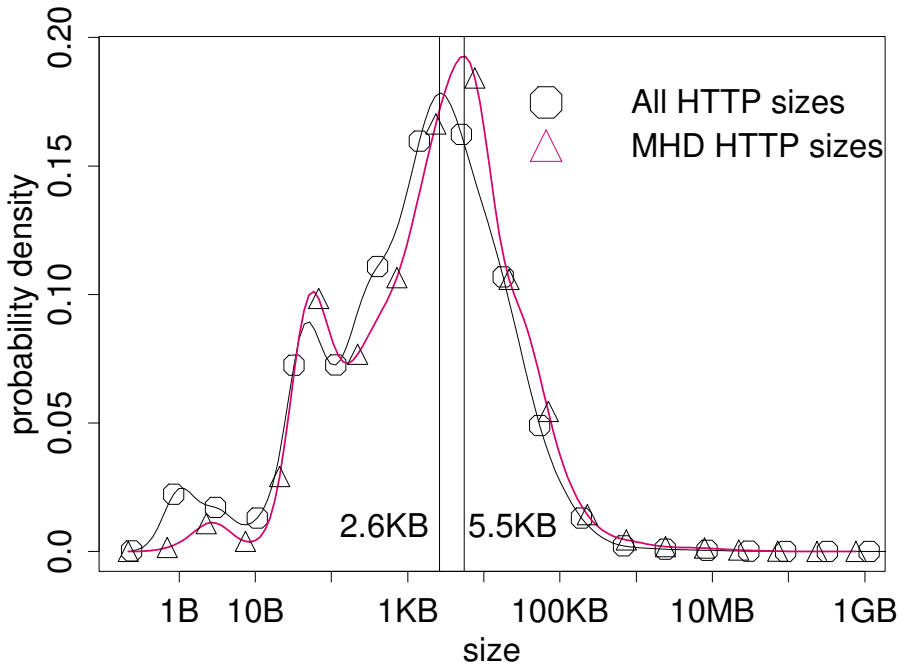


Figure 6.6: **PDF of HTTP object sizes for all and MHD traffic in ISP-A3**

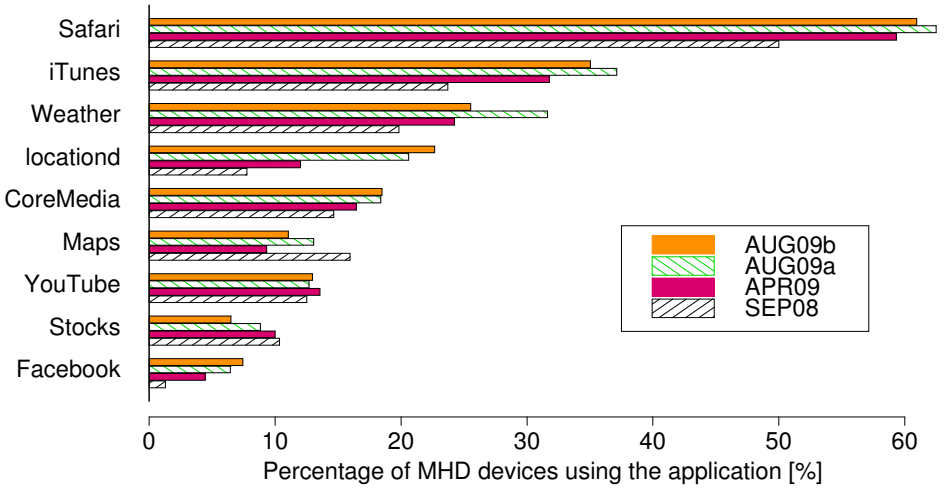


Figure 6.7: **Barplot of application popularity by number of MHD devices using this application**

Table 6.2: **Request and volume of downloads from AppStore**

	# Apps	by PC/Mac		by MHD	
Trace	available	Volume	# Req	Volume	# Req
ISP-A2	3,000	<1 GB	<100	<1 GB	<100
ISP-A3	7,500	<1 GB	>100	>2 GB	>250
ISP-A4-a	70,000	>2 GB	>150	>3 GB	>450
ISP-A4-b	70,000	>3 GB	>150	>3 GB	>400

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.

6.3 Related work

Only a small number of studies have focused on Internet traffic in 3G mobile or cellular networks. Moreover, so far, to the best of our knowledge, no study has focused on multimedia (video) enabled mobile hand-held devices.

Svoboda et al. [102] analyze various aspects of GPRS and UMTS traffic using anonymized header traces from 2004 and 2005. They study traffic volume per user and protocol mix and find that the traffic volume per user is 5.4 MB per week. In terms of protocol mix, they see that HTTP is the dominant protocol with 40–60 % of traffic. For GPRS WAP is second most prominent protocol with 19–37 %. With

UMTS no other protocol has a share of more than 8 %. Heikkinen et al. [38] 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, while email is responsible for 10–24 %. P2P is not noticeable. Williamson et al. [115] analyze packet/data call event traces from a CDMA2000 cellular 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 [13, 53, 118]. Other studies analyze the content requested or available for mobile devices. Using data from 2000, Adya et al. [3] 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. [104] use active measurements to crawl the Web for sites offering specialized content for mobile devices. In particular, they searched for popular mobile Web formats (e. g., WML, XHTML-MP). Verkasalo [108] 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. [106] 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 Wi-Fi. 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 asses who many *users* utilize a particular application, which is the approach we use to characterize application usage.

6.4 Summary

Our analysis of 20,000 residential broadband DSL lines of a large European ISP shows that there is a significant and increasing number of active MHDs. 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. In future work we plan a more detailed analysis of non-HTTP protocols and refine our methodology for protocol classification. In addition, we plan to extend our analysis to traces from cellular data networks.

Chapter 7

Today's Usenet Usage: Characterizing NNTP Traffic

The Usenet evolved from its UUCP architecture to a worldwide distributed Internet discussion system in which users read and post public messages to one or more categories, known as newsgroups, via NNTP. These messages are called articles or posts, and collectively are referred to as news. Usenet is similar to bulletin board systems and as such it is the precursor to various Internet forums. Today the traditional Usenet features are offered by forums and mailing lists and are integrated into blogs and Online Social Networks (OSNs).

The Usenet is realized across a constantly changing set of servers that store and forward messages among each other. Since Usenet servers do not have unlimited disk space to hold every article that was ever posted, the oldest articles are deleted as new articles arrive. Usenet distinguishes between binary groups and text based groups since the storage and bandwidth requirements for binary groups are substantial. Due to its size a single binary message may squeeze out several hundreds of text-only postings and its download may impose significant bandwidth cost for the server provider. Therefore, NNTP administrators started to exclude binary groups on their publicly accessible news servers. As a consequence, NNTP became less desirable and eventually its usage declined as alternatives became available.

However, NNTP is again gaining popularity and multimedia data such as video, audio, and pictures as well as RAR archives are among the dominant Internet content [60, 85, 93]. Given this revival it is important to understand its causes and examine its characteristics. Therefore, we developed an analyzer for the Bro [78] network intrusion detection system (NIDS). Our analyzer takes advantage of Bro's capabilities and utilizes dynamic protocol detection [25] and a specialized protocol semantic parser.

In this chapter, we present observations from passive packet-level monitoring of more than 20,000 residential DSL lines from a major European ISP. This unique vantage point coupled with our NNTP protocol analyzer provides a broad view of its usage. We used Bro's online analysis capabilities to collect anonymized NNTP summaries which lasted for more than two weeks. In addition, we applied our analyzer to the traces also studied by Maier et al. [60].

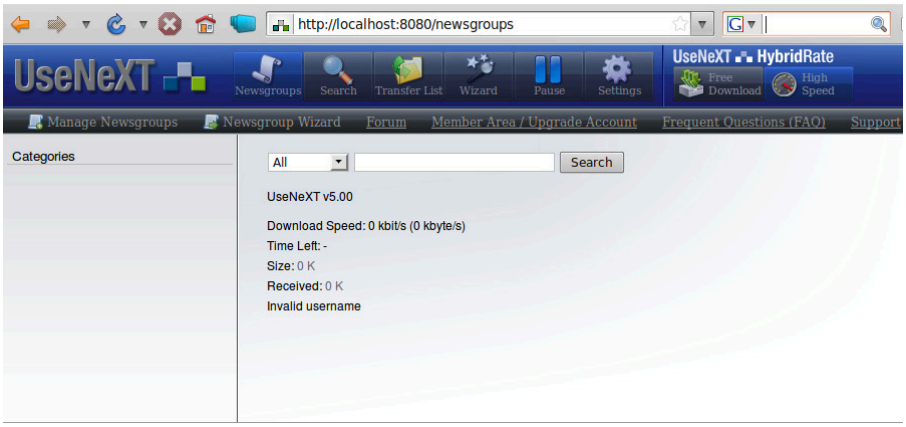


Figure 7.1: **Screen-shot of an NNTP client (Linux) of a fee-based offer**

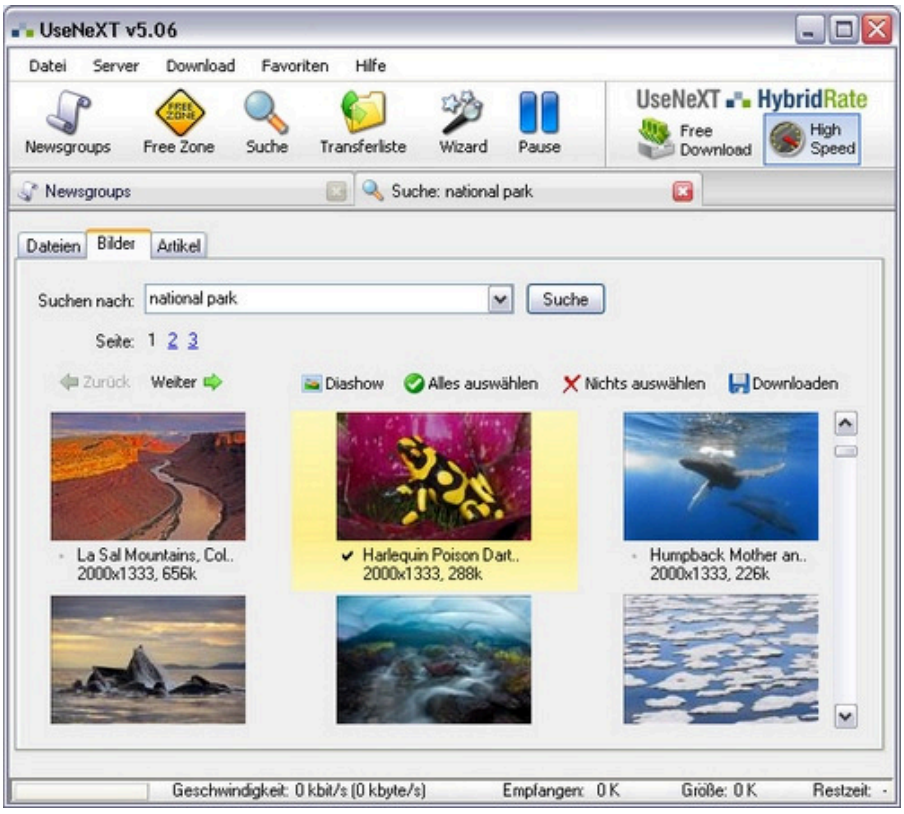


Figure 7.2: **Screen-shot of an NNTP client (Windows) of a fee-based offer**

To the best of our knowledge this is the first study of NNTP traffic since the advent of blogs and OSNs that provide NNTP like features in a shiny browser-based dress. Our initial expectation was that NNTP servers do not provide binary data. However, we find that most of the traffic ($> 99\,\%$) is binary content. We find that this is enabled by NNTP servers that require their users to pay a monthly fee and to authenticate themselves before using the server. Examples for such fee-based server providers are GigaNews or UseNext. Examining content-types of this binary traffic we find that archive formats as well as multimedia (AVI, MP3, and JPG) are dominating and are predominantly transfered with yEnc as binary-to-text encoding.

Furthermore, we note that such fee-based NNTP offers often provide customized NNTP clients (see for example Figure 7.1 or Figure 7.2). Some of these clients take advantage of opening multiple simultaneous TCP connections and use globally unique article IDs instead of per news-group indexing. Moreover, the per connection throughput for NNTP is significantly larger for NNTP clients than for traditional P2P clients. Therefore, we conclude that NNTP's revival is due to the possibility of using NNTP as a high performance client/server-based alternative to P2P file-sharing, even though users have to pay a monthly fee.

The remainder of this chapter is structured as follows: In Section 7.1 we give a short summary of NNTP. The design and implementation of the analyzer is discussed in Section 7.2. After giving a short overview of our datasets in Section 7.3 we present our results in Section 7.4. We summarize in Section 7.5.

7.1 A refresher on NNTP

While NNTP is a well-known and well-established protocol its technical details might have faded from the readers memory. Therefore, we briefly review the basics of the protocol underlying the Usenet—NNTP—in this section. The Usenet is one of the oldest parts of the Internet, in fact it predates the World Wide Web by more than 10 years. It was designed and is still being used as a system to exchange messages (called articles) between groups of users with similar interests—a function that today is also provided, e. g., by Web-forums.

The Usenet is structured into a (pseudo) hierarchy of groups, e. g., `comp.os.linux.networking` or `comp.os.minix`. Anyone with access to a news server can subscribe to any of the news groups and then read or post messages within the group. The news servers are usually hosted by ISPs or at universities and are connected to form an overlay network[1]. The overlay is used to replicate articles between servers so that everyone has access to all articles not just those posted to the local news server. There is one limitation however: It is the decision of the administrator of a

[1] Users whose ISP does not offer access to a news server can subscribe for a small fee to independent servers, e. g., `http://news.individual.net/`.

news server if a particular news group is hosted on his server, e. g., nowadays most news servers typically do not host the majority of the binary groups in order to avoid the risk of excessive bandwidth usage.

While NNTP is used for both client-server as well as server-server communication we in the remainder of this chapter focus on client-server communication since our focus is on NNTP usage by residential users. The Network News Transfer Protocol (RFC 3977) and its message formats (RFCs 5536, 5537, until Nov 2009: RFC 1036) are very similar to SMTP. In fact, many of the article header fields are identical to the email message header fields, e. g., `From` and `Subject`. In addition, there are NNTP specific headers, for example the `Message-ID` header, which contains a unique identifier for an article valid on all news servers.

The IANA assigned ports 119/tcp for NNTP and 563/tcp for SSL-encapsulated NNTP (NNTPS) are the default ports for communication between an NNTP client and server. The dialog starts when the client contacts the server. The server answers with a greeting message. Then the client can issue its commands. The server replies to each client command with a three digit status code, a status message, and an optional data block in "multi-line" encoding which contains, e. g., the requested article. Likewise a client can send a "multi-line" data block to the server, e. g., to post an article. At the beginning of the connection a news server may require authorization before the client can issue any commands. Overall, we identify 33 different NNTP commands that are either specified in one of the RFCs or offered by popular servers such as INN. Examples include selecting a group (`GROUP`), listing the articles within a group (`LISTGROUP` or (`X`)`OVER`), and fetching (`ARTICLE`, `BODY` and `HEAD`) or sending (`POST`) articles.

The article itself is composed of a newline-separated list of headers and an article body separated by a double newline[2]. Within NNTP everything is encoded as a "multi-line" data-block, including article bodies, or command results. This "multi-line" data-block format of NNTP imposes several restrictions on the content. For example it cannot contain NUL characters, it has a limited line length, and the period character (full stop) at the beginning of a line has to be escaped. Therefore, it is impossible to transfer binary data without encoding. Popular encodings are yEnc, UUencode, MIME, base64, BinHex, Quoted-Printable and XXEncode.

Note that MIME has just recently been standardized (RFCs 5536 and 5537) as transfer encoding for the Usenet while yEnc and UUencode are well established. UUencode has originated on System V and has traditionally been used to transfer email messages and Usenet articles before the corresponding TCP based protocols were in use. With Usenet in mind yEncode has been designed to minimize the overhead imposed by alternative encoding schemes. The idea is to only encode characters if it is absolutely required to adhere to the message format standard. The name yEncode is actually a wordplay: "Why encode?". While it is in principle

[2] A newline in this context is the two character string `\r\n`.

possible that other binary encodings may be in use and may not be identified their traffic volume in our data sets is minimal.

7.2 Methodology

For our analysis we use the Bro network intrusion detection system [78] as network protocol analysis tool. Bro features TCP stream re-assembly, a scripting language making it easy to manage state and to generate reports, and BinPAC, a protocol parser generator [76]. Moreover, Bro supports dynamic protocol detection (DPD) [25]. Our Bro analyzer for NNTP is also based on DPD. For each connection, the system first identifies potential protocols in use and then activates appropriate analyzers to verify the decision and extract higher-level semantics. As such the traffic classification is based on protocol analysis rather than port numbers and/or signature matching and we can detect NNTP usage on non-NNTP ports. However, we identified only a small fraction of NNTP traffic on non-NNTP ports. In addition, the protocol analysis enables us to understand which features/commands of NNTP are used. Moreover we can identify non-RFC conforming use of the NNTP protocol.

For detecting NNTP connections we use a DPD signature that checks if the server side of the connection contains an NNTP welcome banner and if the client side uses one of the 33 NNTP commands[3]. After a connection is identified as candidate for NNTP an analyzer written in BinPAC is attached to the connection and generates an event for every detected request command, reply message, and client and server side (multi-line) data sections.

Note, that because NNTP is stateful, as opposed to, e. g., HTTP, our analyzer also needs to keep state per instance. Examples for this necessity include client-side data transfers (e. g., via `POST`, `IHAVE` and `XREPLIC` commands) or replies with or without multi-line data blocks depending on the request. In Figure 7.3, we show that if the client wants to post a multi-line article it first sends a single-line query, waits for a positive response from the server, and only then sends a multi-line data block. In Figure 7.4 we illustrate different response-types for status code 211 depending on the command issued by the client. Therefore, the analyzer needs to remember the last command for every connection, producing state that needs to be managed.

NNTP uses different encoding types to transfer binary content. To determine the encoding type we use the BinPAC analyzer itself instead of the scripting layer to lower CPU load. Thus, we add an event if a text-encoded binary payload is located to inform the scripting layer about encoding type and method. We distinguish between

[3] To minimize the number of false positives we currently filter traffic on TCP port 25, mainly used by SMTP.

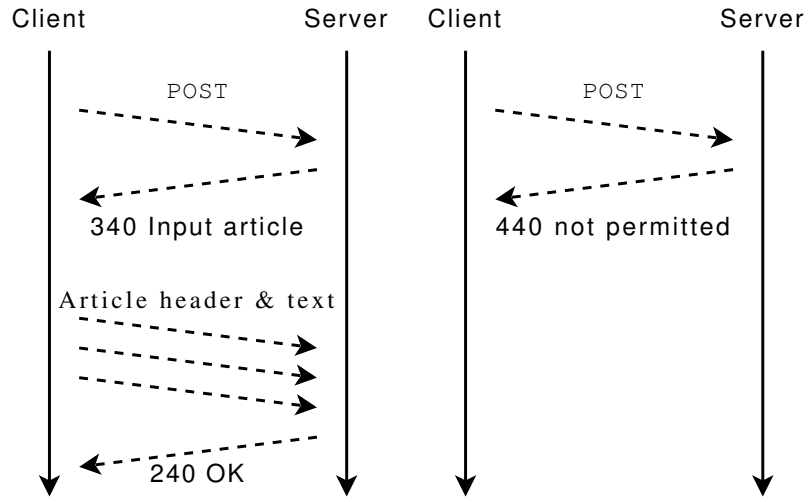


Figure 7.3: **Example scenarios of a `POST` transaction: multi-line data is transmitted only when `POST` request is granted by the server (left)**

(i) yEncode (yEnc) [39], *(ii)* UUencode (RFC 976), *(iii)* MIME (RFC 2045), and *(iv)* non-binary content.

Articles with MIME attachments are identified by looking for and inspecting the `Content-Type` header. Content types and filenames are then extracted by parsing the MIME formatted body. UUencoded article bodies are detected by looking for an UUencode header in the article body. This header starts with the string '`begin␣`' followed by a three digit number denoting the UNIX permissions in octal notation, and a file name. The binary block ends with the string '`end`' on a single line. yEnc works similar to UUencode: the binary block starts with a string '`=ybegin␣`' followed up by parameters describing the length of each block, the size of the resulting file, and the file name on the same line. Each block ends with the string '`=yend␣`' followed by additional parameters.

Our methodology for determining the content-type of a binary encoded file is straightforward and relies on either the filename extension or the content-type of the binary-to-text encoding. In future work we plan to extend the analyzer to decode the binary file and use libmagic to determine the file-type.

The analyzer produces an anonymized one line summary for every single or multi-line request or response including a time-stamp, a connection identifier, anonymized IP addresses of client and server, size of the action, and information about the observed action. This output is post-processed to generate one-line summaries of corresponding NNTP request and response pairs, called NNTP transactions in the remainder of the chapter. In these one-line summaries we also report the news group in which the request was issued.

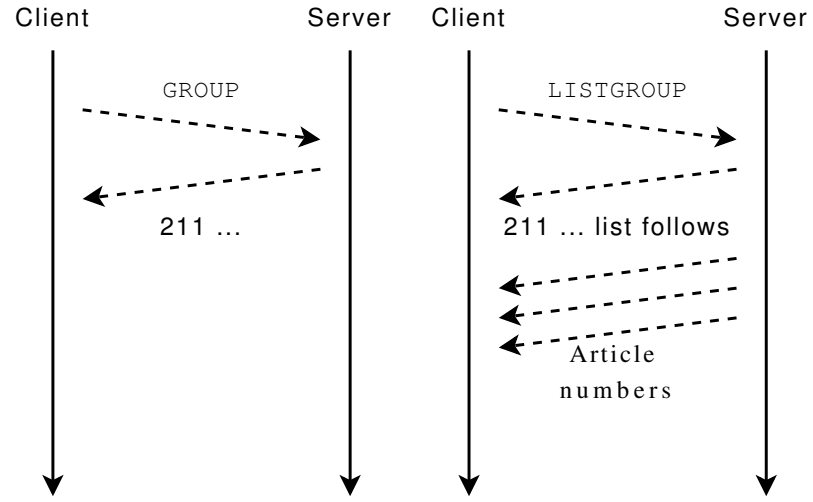


Figure 7.4: **Example scenarios of a 211 response code: multi-line data is transmitted only when a `LISTGROUP` request was send (right)**

Table 7.1: **Overview of anonymized packet traces and NNTP summaries.**

Name	Start date		Duration	Size	NNTP
NNTP-16d	Wed	05 Aug'09 3am	$15\frac{1}{4}$ d	> 2 GB	n/a
ISP-A2	Thu	18 Sep'08 4am	24 h	>4 TB	5 %
ISP-A3	Wed	01 Apr'09 2am	24 h	>4 TB	2 %
ISP-A4	Fri	21 Aug'09 2am	48 h	>11 TB	2 %

7.3 Data sets

We base our study on multiple sets of anonymized packet-level observations of residential DSL connections collected at aggregation points within a large European ISP. The monitor, using Endace monitoring cards, operates at the broadband access router connecting customers to the ISP's backbone. Our vantage point allows us to observe more than 20,000 DSL lines.

Table 6.1 summarizes characteristics of the data sets, including their start, duration, size, and fraction of NNTP. We used Bro's online analysis capabilities to collect an anonymized trace summary which covers more than two weeks of NNTP traffic (NNTP-16d). Moreover, we use several anonymized one/two day packet traces collected over a period of 11 months (ISP-A2, ISP-A3, ISP-A4). These are the same traces as studied by Maier et al. [60]. While we typically did not experience any packet loss, there are several multi-second periods (less than 5 minutes overall per packet trace) with no packets due to OS/file-system interactions.

All traces contain a noticeable fraction of NNTP traffic. Similar results have been observed at different times and at other locations of the same ISP. For the online trace summary we only capture NNTP traffic and therefore do not know the exact fraction. However, comparing the volume to other traces we presume that it corresponds to a similar fraction.

We omit analyzing NNTPS for two reasons: *(i)* due to the encrypted nature of NNTPS it is impossible to inspect the content and *(ii)* the amount of traffic on the respective port is small compared to the observed NNTP traffic. Indeed, we do not observe more than 0.3 % of total traffic on port 563, the standard SSL port of NNTP.

7.4 Results

To understand the revival of NNTP we have to study how NNTP is used today. Accordingly, in this section, we start by studying the overall characteristics of our traces. Then we examine the volume distribution of NNTP transactions, the popularity of commands, encoding methods, content types, news groups, and NNTP servers.

7.4.1 NNTP characteristics

In our traces of residential Internet traffic 2 % (5 % for ISP-A2) is identified as NNTP. We observe roughly 150 users in each of the packet level traces (ISP-A2, ISP-A3, ISP-A4) and roughly 300 users in NNTP-16d. We also observe that the typical NNTP client uses multiple TCP connections in parallel (1st quartile: 2–3, median: 4–6, 3rd quartile: 7–8) and that up to 12 different servers are contacted by a single client—maybe to balance the load across different servers. Note that less than 1 % of the users (150 out of 20000) causes more than 2 % of the traffic with NNTP only. In addition these users are also among the top users in terms of per-line traffic contribution. This is confirmed by Maier et al. [60]. They also report that P2P use and NNTP use is unlikely to be observed at the same line.

Furthermore, we find that roughly 99 % (only 96 % in ISP-A4) of the transmitted volume is due to binary transfers. Yet, we notice that the fraction of binary queries is rising. The fraction is 55 % for ISP-A2 whereas it is roughly 90 % for the other traces (ISP-A3, ISP-A4, NNTP-16d). Moreover, the average duration of observed NNTP connections[4] is 24 minutes for ISP-A2 while the average connection duration of all 2009 traces is 6 or 9 minutes. In order to understand the origin of these

[4]For this statistic we only consider complete NNTP connections, i. e., those that contain the command `QUIT`.

differences we investigate ISP-A2 in more detail. We find that there are three users in this trace that exhibit strange behavior:

- One line is responsible for roughly 30 % of the NNTP volume in ISP-A2 (mainly via `BODY` commands). In all other traces the top contributing line is responsible for at most 10 % of the volume.
- Another line is responsible for more than 90 % of the `GROUP` and `STAT` commands in ISP-A2.
- A third line interacts with a server such that after normal NNTP activities the server sends tons of single-line responses with an unknown status code (not specified in the RFCs).

These three lines are responsible for roughly 48 % of the NNTP transactions and thus we also report results for ISP-A2 without these 3 lines, labeled as "ISP-A2-w3l". For example, we find that 79 % instead of 55 % are binary downloads for ISP-A2-w3l. Since, ISP-A2 also contains some regular news reading activity the results for ISP-A2-w3l are closer to those of the other traces but still differ. For privacy reasons, we do not investigate these 3 lines in detail.

7.4.2 Distribution of transaction volume

Recall, that an NNTP transaction consists of an NNTP request and its corresponding NNTP response. Figure 7.5 depicts the cumulative distribution of the transaction volumes of each of the traces. This plot highlights several specifics of NNTP traffic. For instance, a single-lined message must not exceed 512 bytes. This results in the step around 100 bytes. The most frequent transaction sizes are 252 kB and the median transaction size 387 kB (see helper line in plot). Those seem to correspond to the sizes of typical binary data blocks, which are parts of multi-part archives or multimedia files. We assume that these sizes are due to either the software used to partition the files or server restrictions on article length. Although we tried to confirm any of the explanations, we did not find evidence.

7.4.3 Popularity of NNTP commands

Although we identified more than 30 possible NNTP commands and include all of them in our signature for detecting NNTP traffic, only 16 of them are observed in any of our traces. The overall frequencies of the top commands are listed in Table 7.2. As expected, the most frequently issued command is `ARTICLE`. Interestingly, the `BODY` request is the next most popular and it is significantly more popular than `HEAD`. This indicates that the users are not downloading the complete meta information about the bodies that they download. Note the effect of removing the 3 lines from ISP-A2.

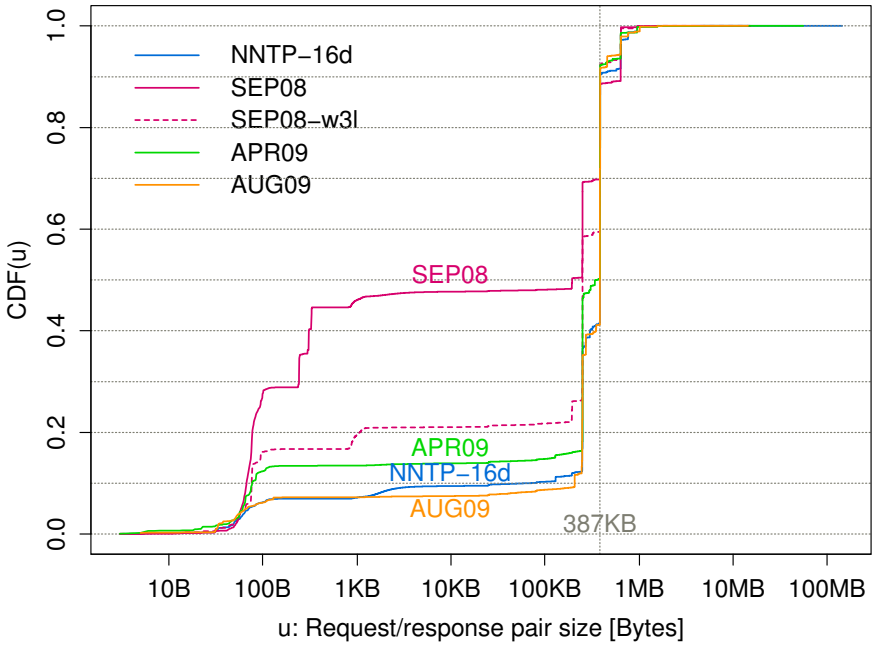


Figure 7.5: **CDF of transaction volumes**

In terms of volume only the commands `ARTICLE` and `BODY` are relevant and together contribute more than 99 %. NNTP offers two different identifiers for articles: *(i)* an article number relative to the newsgroup or *(ii)* a globally unique article identifier. The first kind of identifier requires the user to first select the corresponding newsgroup, e. g., via the `GROUP` command. The latter identifier can be used without entering a newsgroup. We find that the globally unique article identifiers dominate and may explain the small number of `GROUP` commands.

7.4.4 Popularity of binary-to-text encoding methods

Given that most requests and almost all of the bytes are due to binary content we next explore which binary-to-text encodings are used. In all traces yEnc dominates with more than 99 % of the bytes being transfered (Table not shown). The frequency of requests for each encoding method is shown in Table 7.3. Among the binary encodings yEnc is still dominant. We assume that yEnc's prevalence results from its significantly smaller encoding overhead.

7.4.5 Content-types of binary encoded files

Given that almost all NNTP traffic is binary encoded we now explore which files are contained within these binary transfers, see Table 7.4. The most frequently

Table 7.2: **Frequency of NNTP commands**

Commands	ISP-A2	ISP-A2-w3l	ISP-A3	ISP-A4	NNTP-16d
reply only	15,9 %	-	-	-	-
`ARTICLE`	43.6 %	82.6 %	83.2 %	66.5 %	76.5 %
`BODY`	15.7 %	8.1 %	10.3 %	27.1 %	18.1 %
`GROUP`	14.1 %	0.9 %	1.1 %	0.4 %	0.6 %
`STAT`	6.4 %	-	-	-	-
`HEAD`	2.1 %	4.1 %	<0.1 %	<0.1 %	0.3 %
`AUTHINFO`	1.0 %	1.8 %	2.3 %	2.0 %	1.8 %
`QUIT`	0.1 %	0.2 %	0.7 %	0.2 %	0.2 %
`MODE`	0.1 %	<0.1 %	0.2 %	1.2 %	0.6 %
`XOVER`	<0.1 %	<0.1 %	<0.1 %	<0.1 %	<0.1 %

Table 7.3: **Frequency of binary-to-text encoding methods**

	ISP-A2	ISP-A2-w3l	ISP-A3	ISP-A4	NNTP-16d
non-binary	47.7 %	21.1 %	14.7 %	9.8 %	9.8 %
yEnc	52.2 %	78.6 %	84.8 %	89.9 %	89.9 %
UUEncode	0.1 %	0.3 %	0.5 %	0.3 %	0.2 %
MIME	<0.1 %	<0.1 %	<0.1 %	<0.1 %	<0.1 %

transferred file format is archive/rar. RAR is a file compression software which is able to archive files and separate them into multiple smaller files. The file extension of separated files is three characters either 'rar', a 3-digit number or 'r' followed by a 2-digit number. Files for which the extension matches this condition have been assumed to be RAR files. We also observe parchive parity or index files (PAR2). These can be used to recover broken or unavailable data [77]. In addition, we see a significant fraction of multimedia formats. We note that in P2P file-sharing systems, multimedia is more popular than archives according to ipoque [93].

7.4.6 Popularity of news groups

Given our results so far we expect to see a large popularity of binary news groups. This is indeed the case. Most of the observed Usenet activities are in sub groups of `alt.binaries`. However, since most articles are requested via their globally unique ID we cannot identify which group they belong to. Moreover, for globally unique

Table 7.4: **Frequency of binary file types**

	ISP-A2	ISP-A3	ISP-A4	NNTP-16d
archive/rar	84.30 %	84.13 %	83.81 %	81.18 %
video/avi	5.19 %	7.08 %	3.16 %	6.73 %
archive/par2	2.57 %	1.73 %	2.01 %	2.24 %
audio/mp3	1.46 %	2.36 %	2.90 %	3.11 %
image/jpg	0.40 %	2.07 %	1.26 %	0.66 %
other file types	6.08 %	2.63 %	6.86 %	6.08 %

IDs it may be wrong to presume that an article belongs to the most recently selected group. Therefore, we do not present numbers in this paper.

7.4.7 Popularity of news servers

Since binary news groups are usually not available on public NNTP servers we now examine which servers are contacted and apparently provide the desired binary content. To identify the server operator we use two approaches: *(i)* we parsed the welcome message line of the servers, *(ii)* we cooperated with the ISP to perform reverse lookups of the servers IP addresses. Both methods yield the same results: Most of the Usenet servers offering binary groups are commercial servers that require a monthly fee, i. e., GigaNews, UseNeXT, Alphaload, and Firstload. Further investigation reveals that there is even a reselling business for commercial NNTP server access.

Our investigation reveals that more than 93 % of the queries are directed to and more than 99 % of the volume of all NNTP traffic is exchanged with commercial Usenet servers. Browsing the Webpages of these commercial Usenet server operators and taking advantage of some free trial offers we find that such offers come along with a special NNTP client. Clients for Unix-like OSes, e. g., MacOS or Linux, are often realized as background processes that are controlled via a Web-browser by using machine local communication. These clients also provide a search engine for NNTP articles. This is an additional feature that NNTP does not provide by default. It is possible that such search queries are not using the NNTP protocol.

7.4.8 Throughput of NNTP

Next, we pose the question why people are willing to pay a monthly fee for something that is likely to also be freely available, e. g., via P2P file-sharing services. We therefore investigate the achieved throughput of NNTP flows and compare these

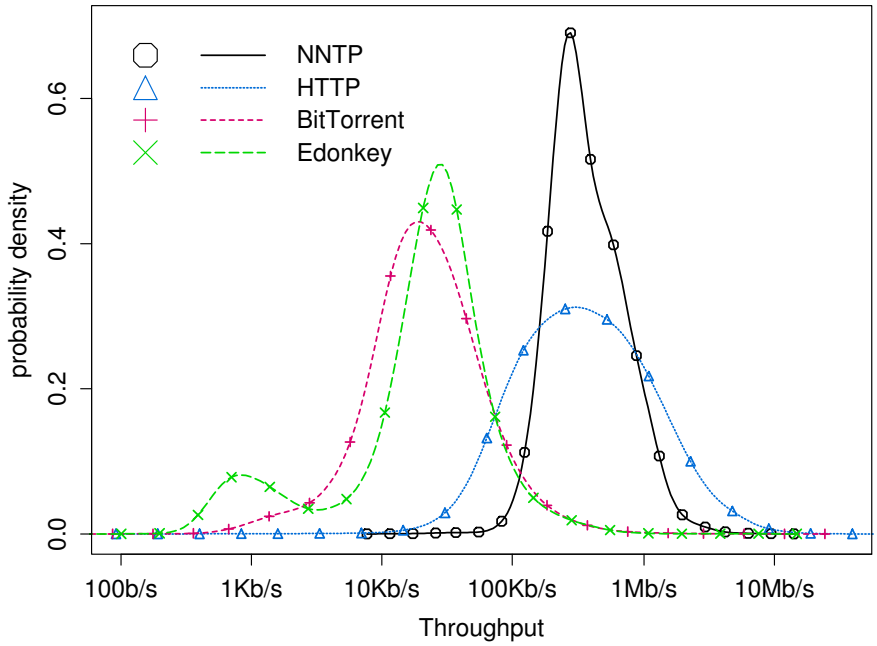


Figure 7.6: **PDF of achieved throughputs of flows > 50 kBytes for different protocols in ISP-A3.**

to BitTorrent, eDonkey, and HTTP flows. Figure 7.6 shows a probability density function of the logarithm[5] of the achieved throughput for ISP-A3. We can clearly see that NNTP outperforms the P2P-based systems. Given that both P2P-based systems and NNTP use multiple connections in parallel we assume that the overall time to download the same amount of data is an order of magnitude shorter for NNTP. This is basically due to the fact that NNTP servers usually have better Internet connectivity as compared to P2P users. For HTTP the achieved throughput is in the same order as for NNTP, although HTTP has a higher variability than NNTP. Thus, better performance may be the reason to either use fee-based NNTP servers or fee-based One-Click-Hosters (such as Rapidshare or MegaUpload) for file-sharing. One advantage for NNTP is the ability to search NNTP newsgroups for specific content. Moreover, NNTP features a protocol/operation inherent content replication and content distribution schemes.

[5]Coupled with a logarithmic scale on the x-axis, plotting the density of the logarithm of the data facilitates direct comparisons between different parts of the graphs based on the area under the curve.

7.5 Summary

In this paper we presented our NNTP analyzer for the Bro NIDS and the analysis of how NNTP is used today. Our analysis is based on a live measurement lasting more than two weeks and several packet level traces. The most important observation is that more than 99 % of the volume are binary data transmissions. The distribution of transaction volumes shows that around 80 % of all transaction sizes are at two distinct peaks (252 kB and 387 kB). Furthermore, most traffic is exchanged with commercial servers that require a monthly fee from their customers. The achieved throughput of NNTP connections is at least an order of magnitude higher than P2P-systems like BitTorrent and eDonkey. These insights lead us to the conclusion that the Usenet is "misused" as a file sharing network.

In future work we plan to extend the analysis by using libmagic for content-type determination and including meta information contained in article headers (e. g., news groups).

Chapter 8

Conclusion

The constant evolution of the Internet continuously brings forth new Web trends. In this thesis, we study the effects of several new trends on network traffic and characterize their usage. This chapter summarizes our results. In the outlook, we outline how to utilize our results and present interesting topics for future research.

8.1 Summary

In this thesis, we present a methodology for examining the usage of new trends and phenomena in the Web. We apply our methodology to packet level network observations gathered at several populations of tens of thousands of Internet users.

We start out by evaluating the performance of packet capturing systems. We find that not all operating system and CPU architecture combinations perform equally well. Indeed, the FreeBSD/AMD combination yields the best results. Although we ascertain the best hardware and OS combination for the task of packet capturing, we also learn that there are substantial limitations, especially for the task of capturing in 10 Gigabit environments. To overcome these limitations we propose several approaches for analysis load reduction by distributing the traffic over several analysis entities or by using specialized hard- or software.

Equipped with a high performance measurement system, we attempt to understand how some of the new trends and technologies influence the characteristics of network traffic. For this task we select four AJAX-enabled applications, i. e., Google Maps and Mail, the Web-mailer `gmx.de`, and the social network `lokalisten.de`. AJAX is a JavaScript framework that allows execution of Web applications inside the browser, enabling pre-fetching of content or local user interface rendering. We compare the traffic of the AJAX-based applications against ambient HTTP traffic and find that the traffic patterns of the new technology differs. In particular, our results show that sessions last longer and transfer more data. Moreover, HTTP requests are more and more automated causing two orders of magnitude shorter inter-request (arrival) times and as such, the traffic is more bursty.

Inspired by the huge number of users registered in Online Social Networks (OSNs) we study what it is that attracts so many people. We select popular – and in our

data – well-represented OSNs to assess their usage. From the traffic generated by `facebook.com`, `studivz.net`, `linkedin.com`, and `hi5.com` we reverse-engineer user actions and identify popular features. Our analysis reveals that the popular features are message exchanging, profile browsing, and photo management. Furthermore, we investigate how much time users spend on an OSN. Although the time between login and logout can reach days, considering only periods of user activity we find that users have few active periods of roughly five minutes each per day.

Subsequently, we examine the usage of mobile hand-held devices (MHDs, such as iPhones or Blackberrys). As we do not have access to traces of cellular networks, we take advantage of the Wi-Fi capabilities of MHDs. Investigating traces of residential broadband customers, we indeed observe activities by MHDs. Given the market shares of smart phones we are surprised that more than 70 % of the MHDs are iPhones or iPods. The traffic to these devices is dominated by multimedia content and mobile applications.

Last, we investigate another trend in the Internet, namely the unexpectedly high traffic share of NNTP, the protocol underlying the Usenet. Our initial expectation was that Usenet is no longer widely used, as standard servers do not allow binary content. But the opposite shows to be the case: NNTP traffic is dominated by binary transfers to providers that charge a monthly fee for accessing their servers. These servers provide retention times of several hundred days and even ship their own clients. Comparing the throughput of NNTP connections to P2P NNTP's advantage is striking: NNTP can achieve ten times higher throughputs than P2P.

While every analysis has its own approach and results, our experience shows that there are building blocks that can be reused to speed up subsequent analysis. For example, we use similar techniques in Chapter 4 and Chapter 5 to group requests into sessions. Another example is the tool for connection preserving trace splitting, in order to utilize all CPU cores in a system, which was used for the analyses in Chapter 5, 6, and 7.

It is intriguing to witness that every analysis reveals something that was not expected upfront. Examples include that FreeBSD performs better despite the additional copy operation, that OSN users log in and out with different accounts several times in a row, or that people are paying a monthly fee for faster downloads.

8.2 Future work

Due to the ever increasing importance of the Internet, new trends will continue to emerge. These trends will pose new requirements to the underlying protocols and the network. Therefore, network measurements will continue to be an important source of insights about the Internet. The increasing network speed will also require new approaches in Internet monitoring.

As mentioned in Chapter 5, we are currently building OSN usage models that enable us to evaluate new OSN designs. In the context of the PeerSoN project [15] (`www.peerson.net`) we plan to use our models to evaluate the availability of content in a distributed Online Social Network. Based on realistic user behavior we intend to compare different content replication strategies in terms of availability. Furthermore, we intend to compare the response times of PeerSoN against well established client/server based OSNs such as Facebook.

Market forces are a major contributor for the development of new systems and proposals to improve the Internet. For example, Internet service providers are interested in reducing the amount of traffic that they need to transport, since this reduces their costs. To address this issue we will continue to scrutinize how content that is downloaded multiple times can be leveraged in order to reduce network traffic. Our poster [4] at SIGCOMM'09 presented first results for P2P systems. In the P2P context the content duplication can be utilized by deploying network topology aware neighborhood selection strategies as discussed in the IETF ALTO working group [94]. So far the results for the HTTP protocol are not promising and a more detailed analysis is needed. For example, big content providers and CDNs artificially personalize their content to enable per request load-balancing. This is counter-productive for achieving high cache hit rates. But perhaps a compromise can be found once the extent of all contributing factors is known.

During our data analysis we noticed that many Internet users are using network address translation (NAT) to connect to the Internet. Quantifying the extent of deployed NAT gateways and understanding the bias that multiple devices behind one IP address imply might be crucial for upcoming Internet studies. This is especially true for techniques that rely on a close coupling of IP addresses to end hosts. Combined with the findings of Maier et al. [60] that DSL lines experience high address churn this can render IP addresses unusable for the task of end host identification.

Acknowledgements

This thesis would not have been possible without the help of a lot of people.

First and foremost, I am deeply indebted to my wife Britta. It was her continuous love, support, and indulgence of extraordinary working hours that enabled me to reach this point. She is also the secretary of our research group, which at least doubles my gratitude to her.

I am extremely grateful to my advisor Anja Feldmann. During my Diplomarbeit she introduced me to the world of research and kindly offered me to continue working with her as a doctoral student. Her motivation, guidance, and support during this work was always welcome and helpful. Furthermore, I want to thank her for providing me with the freedom to select my own topics, some of which are presented in this thesis. Over the course of time, she has become a good friend. Moreover, working and discussing with her was always fun and her feedback greatly improved the thesis.

Next, I want to thank my various collaborators. First I want to emphasize the support by Bala Krishnamurthy and Walter Willinger. They were my mentors during my Internship at AT&T Labs – Research, New Jersey. Although we had difficult discussions, I am happy to have learned from them. Moreover, both of them helped a lot in subduing the bureaucracy required to earn money in the US.

I also thank my colleagues and friends from our group at TU Munich and TU Berlin. Special thanks go to my collaborators Sachin Agarwal, Bernhard Ager, Tansu Alpcan, Gregor Maier and Jörg Wallerich, and to all the undergrad students working with me.

The following institutions generously supported this work by providing access to network data. AT&T Labs – Research, New Jersey, the Leibnitz-Rechenzentum, Munich, and Deutsche Telekom AG Laboratories, Berlin. I really appreciate their trust in me, something that is not a given.

Last, but not the least, I thank my parents and my sister who always believed in me.

List of Figures

We use abbrevations for the most common plots in this theses: *(i)* Probability density functions (PDFs), *(ii)* Cumulative distribution functions (CDFs), and *(iii)* Complementary cumulative distribution functions (CCDFs).

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 4th International World Wide Web Conference (WWW)* (Dec. 1995).

[2] Acquisti, A., and Gross, R. Imagined communities: Awareness, information sharing, and privacy on the Facebook. In *Proceedings of the 6th Workshop on Privacy in the Electronic Society (WPES)* (2006).

[3] Adya, A., Bahl, P., and Qiu, L. Characterizing alert and browse services of mobile clients. In *Proceedings of the USENIX Annual Technical Conference* (2002), pp. 343–356.

[4] Ager, B., Schneider, F., and Feldmann, A. Cacheability of bulk content for ISPs. In *Poster Session of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Aug. 2009).

[5] Ager, B., Schneider, F., Kim, J., and Feldmann, A. Revisiting cacheability in times of user generated content. In *Proceedings of the 13th IEEE Global Internet Symposium* (Mar. 2010).

[6] 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`, Sept. 2008.

[7] Arrington, M. Facebook now nearly twice the size of MySpace worldwide, 2009. `http://www.techcrunch.com/2009/01/22/facebook-now-nearly-twice\-the-size-of-myspace-worldwide/`.

[8] Atterer, R., Wnuk, M., and Schmidt, A. Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the 15th International World Wide Web Conference (WWW)* (2006), pp. 203–212.

[9] Backstrom, L., Huttenlocher, D., Kleinberg, J., and Lan, X. Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge discovery and data mining* (2006), pp. 44–54.

[10] Barford, P., Bestavros, A., Bradley, A., and Crovella, M. Changes in web client access patterns: Characteristics and caching implications. *World Wide Web 2*, 1-2 (1999), 15–28.

[11] Basher, N., Mahanti, A., Mahanti, A., Williamson, C., and Arlitt, M. A comparative analysis of web and peer-to-peer traffic. In *Proceedings of the 17th International World Wide Web Conference (WWW)* (New York, NY, USA, 2008), ACM, pp. 287–296.

[12] Benevenuto, F., Rodrigues, T., Cha, M., and Almeida, V. Characterizing user behavior in online social networks. In *Proceedings of the Internet Measurement Conference (IMC)* (2009), pp. 49–62.

[13] Benko, P., Malicsko, G., and Veres, A. A large-scale, passive analysis of end-to-end TCP performance over GPRS. In *Proceedings of the 23th Conference of the IEEE Computer and Communications Societies (INFOCOM)* (2004), vol. 3, pp. 1882–1892 vol.3.

[14] Bernaille, L., Teixeira, R., and Salamatian, K. Early application identification. In *Proceedings of the ACM Conference on Emerging Networking Experiments And Technologies (CoNEXT)* (New York, NY, USA, 2006), ACM, pp. 1–12.

[15] Buchegger, S., Schiöberg, D., Vu, L.-H., and Datta, A. PeerSoN: P2P social networking: Early experiences and insights. In *Proceedings of the 2nd ACM EuroSys Workshop on Social Network Systems (SNS)* (2009), pp. 46–52.

[16] 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), pp. 1–14.

[17] Cha, M., Mislove, A., Adams, B., and Gummadi, K. P. Characterizing social cascades in Flickr. In *Proceedings of the 1st ACM SIGCOMM Workshop on Online Social Networks (WOSN)* (2008), pp. 13–18.

[18] Challenger, J., Iyengar, A., and Danzig, P. A scalable system for consistently caching dynamic web data. In *Proceedings of the 18th Conference of the IEEE Computer and Communications Societies (INFOCOM)* (Mar. 1999), vol. 1, pp. 294–303 vol.1.

[19] Chun, H., Kwak, H., Eom, Y.-H., Ahn, Y.-Y., Moon, S., and Jeong, H. Comparison of online social relations in volume vs. interaction: A case study of Cyworld. In *Proceedings of the Internet Measurement Conference (IMC)* (2008), pp. 57–70.

[20] Croll, A. The impact of AJAX on web operations, Dec. 2005. `http://www.bitcurrent.com/105/`.

[21] Crovella, M., and Bestavros, A. Self-similarity in world wide web traffic: Evidence and possible causes. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)* (May 1996), pp. 160–169.

[22] Deri, L. Improving passive packet capture: Beyond device polling. In *Proceedings of the 4th International System Administration and Network Engineering Conference (SANE)* (Sept. 2004).

[23] Deri, L. nCap: Wire-speed packet capture and transmission. In *Proceedings of the IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services (IM 2005, E2EMON)* (May 2005).

[24] Document Object Model (DOM), 2007. `http://www.w3.org/DOM`.

[25] Dreger, H., Feldmann, A., Mai, M., Paxson, V., and Sommer, R. Dynamic application-layer protocol analysis for network intrusion detection. In *Proceedings of the 15th USENIX Security Symposium* (2006), pp. 257–272.

[26] Ellison, N., Steinfield, C., and Lampe, C. Spatially bounded online social networks and social capital: The role of Facebook. In *Proceedings of the Annual Conference of the International Communication Association* (2006).

[27] Endace Measurement Systems. `http://www.endace.com`, 2009.

[28] Erman, J., Gerber, A., Hajiaghayi, M. T., Pei, D., and Spatscheck, O. Network-aware forward caching. In *Proceedings of the 18th International World Wide Web Conference (WWW)* (2009), pp. 291–300.

[29] Facebook. Facebook blog on the release of `new.facebook.com`, 2008. `http://blog.new.facebook.com/blog.php?post=30074837130`.

[30] Facebook, 2009. `http://www.facebook.com`.

[31] Feldmann, A., Rexford, J., and Caceres, R. Efficient policies for carrying Web traffic over flow-switched networks. *IEEE/ACM Transactions on Networking (ToN) 6*, 6 (1998), 673–685.

[32] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. Hypertext transfer protocol – HTTP/1.1. RFC 2616, 1999.

[33] Fiveg, A. Improving the freebsd packet capturing stack. Diplomarbeit, Technische Universität Berlin, Jan. 2010. Not yet published.

[34] 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), pp. 15–28.

[35] Gjoka, M., Sirivianos, M., Markopoulou, A., and Yang, X. Poking Facebook: Characterization of OSN applications. In *Proceedings of the 1st ACM SIGCOMM Workshop on Online Social Networks (WOSN)* (2008), pp. 31–36.

[36] Global Internet Geography. TeleGeography research. `http://www.telegeography.com/product-info/gb/download/executive-summary.pdf`, 2009.

[37] Golder, S., Wilkinson, D., and Huberman, B. A. Rhythms of social interaction: Messaging within a massive online network. In *Proceedings of the 3rd International Conference on Communities and Technologies (CT2007)* (June 2007).

[38] Heikkinen, M., Kivi, A., and Verkasalo, H. Measuring mobile peer-to-peer usage: Case Finland 2007. In *Proceedings of the 10th International Conference on Passive and Active Measurement (PAM)* (2009), vol. 5448, pp. 165–174.

[39] Helbing, J. yEnc - Efficient encoding for Usenet and eMail. Project home page `http://www.yenc.org/`, 2003.

[40] Hi5, 2009. `http://www.hi5.com`.

[41] Internet World Stats. World internet users and population stats. `http://www.internetworldstats.com/stats.htm`, Sept. 2009.

[42] Jacobson, V., Leres, C., and McCanne, S. libpcap and tcpdump, 2009. `http://www.tcpdump.org`.

[43] Joinson, A. N. Looking at, looking up or keeping up with people? Motives and uses of Facebook. In *Proceedings of the 26th SIGCHI Conference on Human Factors in Computing Systems* (2008), pp. 1027–1036.

[44] Kammenhuber, N., Luxenburger, J., Feldmann, A., and Weikum, G. Web search clickstreams. In *Proceedings of the Internet Measurement Conference (IMC)* (2006), pp. 245–250.

[45] Kim, J., Schneider, F., Ager, B., and Feldmann, A. Today's usenet usage: Characterizing NNTP traffic. In *Proceedings of the 13th IEEE Global Internet Symposium* (Mar. 2010).

[46] Krishnamurthy, B. A measure of online social networks. In *Conference on Communication Systems and Networks (COMSNETS)* (2009).

[47] Krishnamurthy, B., and Rexford, J. *Web protocols and practice: HTTP/1.1, Networking protocols, caching, and traffic measurement*. Addison-Wesley, 2001.

[48] Krishnamurthy, B., and Wills, C. E. Characterizing privacy in online social networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Online Social Networks (WOSN)* (2008), pp. 37–42.

[49] Kumar, R., Novak, J., and Tomkins, A. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge discovery and data mining* (2006), pp. 611–617.

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

[51] Labovitz, C., McPherson, D., and Iekel-Johnson, S. NANOG 47: 2009 internet observatory report. `http://www.nanog.org/meetings/nanog47/abstracts.php?pt=MTQ1MyZuYW5vZzQ3&nm=nanog47`, 2009.

[52] Lampe, C. A., Ellison, N., and Steinfield, C. A familiar face(book): Profile elements as signals in an online social network. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2007), pp. 435–444.

[53] Lee, Y. Measured TCP performance in CDMA 1x EV-DO network. In *Proceedings of the 7th International Conference on Passive and Active Measurement (PAM)* (Mar. 2006).

[54] Leibniz Rechenzentrum LRZ. Münchner Wissenschaftsnetz MWN, 2007.

[55] Leland, W. E., Taqqu, M. S., Willinger, W., and Wilson, D. V. On the self-similar nature of Ethernet traffic. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Mar. 1993), pp. 183–193.

[56] Liben-Nowell, D., Novak, J., Kumar, R., Raghavan, P., and Tomkins, A. Geographic routing in social networks. *Proceedings of the Proceedings of the National Academy of Sciences of the United States of America 102*, 33 (Aug. 2005), 11623–11628.

[57] LinkedIn, 2009. `http://www.linkedin.com`.

[58] LSI corporation. 3ware RAID controllers. `http://www.3ware.org`, 2009.

[59] M. Crovella, P. B. The network effects of prefetching. In *Proceedings of the 17th Conference of the IEEE Computer and Communications Societies (INFOCOM)* (apr 1998), vol. 3, pp. 1232–1239.

[60] 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)* (Nov. 2009), pp. 90–102.

[61] Maier, G., Schneider, F., and Feldmann, A. A first look at mobile handheld device traffic. In *Proceedings of the 11th International Conference on Passive and Active Measurement (PAM)* (Apr. 2010), vol. 6032 of *Lecture Notes in Computer Science*, Springer, pp. 161–170.

[62] 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* (Aug. 2008), ACM Press, pp. 183–194.

[63] McCanne, S., Jacobson, V., and Watson, R. BPF – berkeley packet filter (FreeBSD manpage). `http://www.freebsd.org/cgi/man.cgi?query=bpf&apropos=0&sektion=0&manpath=FreeBSD+8-current&format=html`, 2009.

[64] Mislove, A., Koppula, H. S., Gummadi, K. P., Druschel, P., and Bhattacharjee, B. Growth of the Flickr social network. In *Proceedings of the 1st ACM SIGCOMM Workshop on Online Social Networks (WOSN)* (2008), pp. 25–30.

[65] Mislove, A., Marcon, M., Gummadi, K. P., Druschel, P., and Bhattacharjee, B. Measurement and analysis of online social networks. In *Proceedings of the Internet Measurement Conference (IMC)* (2007), pp. 29–42.

[66] Mogul, J. C., and Ramakrishnan, K. Eliminating receive livelock in an interrupt-driven kernel. *ACM Transactions on Computer Systems 15*, 3 (Aug. 1997), 217–252.

[67] Moore, A., and Papagiannaki, K. Toward the accurate identification of network applications. In *Proceedings of the 6th Passive and Active Measurement Workshop (PAM)* (Apr. 2005), vol. 3431 of *Lecture Notes in Computer Science*, Springer, pp. 41–54.

[68] Napatech. `http://www.napatech.com`, 2009.

[69] 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), pp. 43–56.

[70] Nazir, A., Raza, S., Gupta, D., Chuah, C.-N., and Krishnamurthy, B. Network level footprints of Facebook applications. In *Proceedings of the Internet Measurement Conference (IMC)* (New York, NY, USA, 2009), ACM, pp. 63–75.

[71] NetFPGA. `http://www.netfpga.org`, 2009.

[72] Odlyzko, A. M. Internet traffic growth: Sources and implications. `http://www.dtc.umn.edu/mints/home.php`, 2003.

[73] Olsson, R. pktgen the Linux packet generator. In *Proceedings of the Linux Symposium – Volume Two* (Ottawa, Canada, July 2005), pp. 11–24.

[74] Olsson, R. Linux kernel packet generator. Linux Kernel sources, v2.6.8, documentation in kernel tree: `Documentation/networking/pktgen.txt`, last changes 2002. Uppsala University, Sweden.

[75] OpenFlow Consortium. `http://www.openflowswitch.org`, 2009.

[76] Pang, R., Paxson, V., Sommer, R., and Peterson, L. binpac: A yacc for writing application protocol parsers. In *Proceedings of the Internet Measurement Conference (IMC)* (2006), pp. 289–300.

[77] Parchive. `http://parchive.sourceforge.net/`.

[78] Paxson, V. Bro: a system for detecting network intruders in real-time. *Computer Networks 31*, 23-24 (1999), 2435–2463.

[79] Paxson, V. Bro intrusion detection system, 2007. `http://www.bro-ids.org`.

[80] Paxson, V., and Floyd, S. Wide area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking (ToN) 3*, 3 (June 1995), 226–244.

[81] Plissonneau, L., Costeux, J.-L., and Brown, P. Analysis of peer-to-peer traffic on adsl. In *Proceedings of the 6th Passive and Active Measurement Workshop (PAM)* (2005), vol. 3431, pp. 69–82.

[82] Pronchev, I. Packet capturing using the linux netfilter framework. Undergrad project, Technische Universität München, Oct. 2006. `http://www.net.t-labs.tu-berlin.de/~fabian/papers/pronchev_sep_okt2006.pdf`.

[83] Rizzo, L. Device polling support for FreeBSD. In *Proceedings of the Main European BSD Conference (EuroBSDCon)* (Brighton, UK, 2001).

[84] Salim, J. H., Olsson, R., and Kuznetsov, A. Beyond softnet. In *Proceedings of the Fifth Annual Linux Showcase & Conference* (Nov. 2001).

[85] Sandvine Inc. 2009 global broadband phenomena. Research Report `http://www.sandvine.com/news/global_broadband_trends.asp`, 2009.

[86] Schneider, F. Performance evaluation of packet capturing systems for high-speed networks. Diplomarbeit, Technische Universität München, Munich, Germany, Nov. 2005.

[87] Schneider, F., Agarwal, S., Alpcan, T., and Feldmann, A. The new Web: Characterizing AJAX traffic. In *Proceedings of the 9th International Conference on Passive and Active Measurement (PAM)* (Apr. 2008), vol. 4979 of *Lecture Notes in Computer Science*, Springer, pp. 31–40.

[88] 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)* (Nov. 2009), ACM Press, pp. 35–48.

[89] Schneider, F., and Wallerich, J. Performance evaluation of packet capturing systems for high-speed networks. In *Proceedings of the ACM Conference on Emerging Networking Experiments And Technologies (CoNEXT) Student Workshop* (Oct. 2005), pp. 284–285.

[90] Schneider, F., Wallerich, J., and Feldmann, A. Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware. In *Proceedings of the 8th International Conference on Passive and Active Measurement (PAM)* (Apr. 2007), vol. 4427 of *Lecture Notes in Computer Science*, Springer, pp. 207–217.

[91] Schneider, F., Wallerich, J., Feldmann, A., and Sommer, R. High performance packet capture. `http://www.net.t-labs.tu-berlin.de/research/hppc`.

[92] Schulze, H., and Mochalski, K. Internet study 2007. `http://www.ipoque.com/resources/internet-studies/` (need to register), 2007.

[93] Schulze, H., and Mochalski, K. Internet study 2008/2009. `http://www.ipoque.com/resources/internet-studies/` (need to register), 2009.

[94] Seedorf, J., and Burger, E. W. Application-layer traffic optimization (ALTO) problem statement. RFC 5693, Oct 2009.

[95] Silverstein, C., Marais, H., Henzinger, M., and Moricz, M. Analysis of a very large web search engine query log. *ACM SIGIR Forum 33*, 1 (1999), 6–12.

[96] Snort, 2009. `http://www.snort.org`.

[97] Solera Networks. `http://www.soleranetworks.com`, 2009.

[98] Spink, A., Koshman, S., Park, M., Field, C., and Jansen, B. J. Multitasking web search on vivisimo.com. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC) - Volume II* (2005), pp. 486–490.

[99] Spink, A., Wolfram, D., Jansen, M. B. J., and Saracevic, T. Searching the web: The public and their queries. *Journal of American Society for Information Science and Technology 52*, 3 (2001), 226–234.

[100] StudiVZ, 2009. `http://www.studivz.net`.

[101] Stutzman, F. An evaluation of identity-sharing behavior in social network communities. *Journal of the International Digital Media and Arts Association 3*, 1 (2006).

[102] 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, pp. 40–51.

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

[104] Timmins, P., McCormick, S., Agu, E., and Wills, C. Characteristics of mobile web content. *Hot Topics in Web Systems and Technologies* (2006), 1–10.

[105] Torkjazi, M., Rejaie, R., and Willinger, W. Hot today, gone tomorrow: On the migration of MySpace users. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Online Social Networks (WOSN)* (2009), pp. 43–48.

[106] 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), pp. 267–279.

[107] Valafar, M., Rejaie, R., and Willinger, W. Beyond friendship graphs: A study of user interactions in Flickr. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Online Social Networks (WOSN)* (2009), pp. 25–30.

[108] Verkasalo, H. Empirical observations on the emergence of mobile multimedia services and applications in the U.S. and Europe. In *Proceedings of the 5th international conference on Mobile and ubiquitous multimedia* (2006).

[109] Viswanath, B., Mislove, A., Cha, M., and Gummadi, K. P. On the evolution of user interaction in Facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Online Social Networks (WOSN)* (2009), pp. 37–42.

[110] Wasserman, S., and Faust, K. *Social network analysis*. Cambridge University Press, 1994.

[111] Watson, R., and Peron, C. FreeBSD developer summit presentation describing zero-copy BPF. `http://www.watson.org/~robert/freebsd/2007asiabsdcon/20070309-devsummit-zerocopybpf.pdf`, 2007.

[112] Weinreich, H., Obendorf, H., Herder, E., and Mayer, M. Off the beaten tracks: Exploring three aspects of web navigation. In *Proceedings of the 15th International World Wide Web Conference (WWW)* (2006), pp. 133–142.

[113] Wikipedia. Smartphone: Operating systems. `http://en.wikipedia.org/w/index.php?title=Smartphone&oldid=315621107%23Operating_systems`, Sept. 2009.

[114] WildPackets. `http://www.wildpackets.com`, 2009.

[115] Williamson, C., Halepovic, E., Sun, H., and Wu, Y. Characterization of CDMA2000 cellular data network traffic. In *Proceedings of the The IEEE Conference on Local Computer Networks (LCN)* (2005), pp. 712–719.

[116] Willinger, W., Taqqu, M. S., Sherman, R., and Wilson, D. V. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking (ToN) 5*, 1 (Apr. 1997), 71–86.

[117] Wireshark Foundation. `http://www.wireshark.org`, 2009.

[118] 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 8th International Conference on Passive and Active Measurement (PAM)* (2007), pp. 223–227.

[119] 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`, Sept. 2009.

[120] Wood, P. libpcap mmap mode on linux. `http://public.lanl.gov/cpw/` see 2. (accessed on 30 Sep 2005).

[121] Zakas, N., McPeak, J., and Fawcett, J. *Professional AJAX*. Wiley, 2006.

[122] Zink, M., Suh, K., Gu, Y., and Kurose, J. Watch global, cache local: YouTube network traces at a campus network – Measurements and implications. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (Jan. 2008), vol. 6818.