Pitfalls in HTTP Traffic Measurements and Analysis

Fabian Schneider[1,2], Bernhard Ager[2], Gregor Maier[2,3],
Anja Feldmann[2], and Steve Uhlig[4]

[1] NEC Laboratories Europe, Heidelberg, Germany
(work done at UPMC Sorbonne Universités, LIP6, Paris, France)
[2] TU Berlin / Telekom Innovation Laboratories, Berlin, Germany
[3] International Computer Science Institute, Berkeley, CA, USA
[4] Queen Mary, University of London, London, UK

Abstract. Being responsible for more than half of the total traffic volume in the Internet, HTTP is a popular subject for traffic analysis. From our experiences with HTTP traffic analysis we identified a number of pitfalls which can render a carefully executed study flawed. Often these pitfalls can be avoided easily. Based on passive traffic measurements of 20.000 European residential broadband customers, we quantify the potential error of three issues: Non-consideration of persistent or pipelined HTTP requests, mismatches between the Content-Type header field and the actual content, and mismatches between the Content-Length header and the actual transmitted volume. We find that 60 % (30 %) of all HTTP requests (bytes) are persistent (i. e., not the first in a TCP connection) and 4 % are pipelined. Moreover, we observe a Content-Type mismatch for 35 % of the total HTTP volume. In terms of Content-Length accuracy our data shows a factor of at least 3.2 more bytes reported in the HTTP header than actually transferred.

1 Introduction

HTTP has become the preferred protocol for many Internet services. Internet users exchange most of their content via HTTP. They download videos from YouTube, share photos and status updates via Facebook and Twitter, send and read emails with Gmail or Yahoo, or get the latest news and shop online. Thus analyzing HTTP traffic has been the focus of many recent studies [1–6, 9, 10]. Often these studies are based on passive packet-level traffic measurements. These measurements often imply huge amounts of data to be analyzed, sometimes requiring simplifications to scale the analysis. Some of these simplifications will lead to potential biases in the results or even render a study flawed, as we show in this paper.

In prior analysis of large HTTP traces [1, 6, 9, 10], we encountered such inconsistencies and identified and solved the problem. In this paper we quantify the potential errors of those issues. We study the prevalence of pipelined and persistent HTTP connections, and the accuracy of the Content-Length and Content-Type HTTP header fields. Our results are drawn from anonymized HTTP traffic from roughly 20,000 European residential DSL customers (Section 2). Our contributions are the following:

– **Persistence and pipelining:** By not considering pipelined and persistent HTTP connections, e. g., by only considering the first request in a HTTP connection, the

Table 1. Overview of anonymized HTTP data sets.

Name	Start date	Duration	HTTP volume
SEP08	18 Sep 2008	24 h	≈ 2.5 TB
APR09	01 Apr 2009	24 h	≈ 2.5 TB
AUG09	21 Aug 2009	48 h	≈ 5.9 TB
MAR10	04 Mar 2010	24 h	≈ 3.3 TB
JUN10	23 Jun 2010	24 h	≈ 3.2 TB
HTTP14d	09 Sep 2009	14 d	≈ 42 TB
HTTP12d	07 May 2010	12 d	≈ 38 TB

number of HTTP requests is highly underestimated. Indeed, each connection contains more than 2 requests on average. We find (Section 3) that 60 % of all HTTP requests are persistent, i.e., they are not the first in a TCP connection. These persistent requests are responsible for 30 % of the HTTP volume, and will be missed if not taken into account in the analysis. Contrary to our expectations, we find that the most influential aspect on the use of persistent or pipelined requests is the contacted web service, not the user's browser.

– **HTTP content type:** HTTP servers can specify the mime-type of the transferred object in the Content-Type header. However, there is no guarantee that this information is correct. Indeed, we find (Section 4) that for 35 % of the HTTP volume, the Content-Type header is different from what is found by libmagic.
– **HTTP content size:** The Content-Length field in HTTP headers can state incorrect values for canceled transfers or erroneous Web servers. In our dataset we record at least 3.2 times more bytes reported in Content-Length headers than is actually transferred (Section 5).

2 Data & Methodology

In this section we present our data sets before describing our methodology. We conclude this section by introducing the user population with respect to Browser and OS usage.

Anonymized HTTP data Our study is based on multiple sets of anonymized packet-level observations of residential DSL connections collected at a large European ISP over the course of more than 2 years. Data anonymization and analysis is performed immediately on the secured measurement infrastructure. We use Bro [8] with a customized HTTP analysis script. The monitor operates at the broadband access router connecting customers to the ISP's backbone and observes the traffic of more than 20,000 DSL lines. The monitor uses Endace monitoring cards. While we typically do not experience any packet loss, there are several multi-second periods with no packets (less than 5 minutes overall per trace) due to OS/file-system interactions. We summarize the characteristics, including start time, duration and size, of the traces in Table 1. HTTP14d and HTTP12d do not include request headers. Please refer to Maier et al. [6] for detailed characteristics of this user population (e.g. DSL session lenght, application usage or network performance).

Persistence and Pipelining We investigate the prevalence of persistence and pipelining in HTTP connections. We define the first request/response of a connection as *initial*,

and mark all follow-up requests/replies as *persistent*. Hence, *persistent connections* are connections with more than one request.[5] This allows us to derive the following metrics: *(i)* The fraction of connections that are persistent, *(ii)* the fraction of requests that are persistent, and *(iii)* the fraction of bytes transported in persistent request/response pairs.

Next, if a request is issued before the response of an earlier request in the same connection is finished, we mark this request as *pipelined*. Moreover, if pipelined requests are sent in the *same IP packet*[6], we mark them as such. Similarly to the persistent requests, we derive the same fractions as metrics for the pipelined/same packet marking method. Because HTTP14d and HTTP12d do not include request timestamps, we cannot use those to determine pipelined/same packet requests.

Note that we never mark the first request of a connection as persistent nor do we mark the first request of a pipeline (or same-packet) as such. The reasoning behind this is that we want to focus on the differences to a one request per connection model (HTTP/1.0). Also note that each request marked as pipelined will always be marked as persistent as well, and thereby the set of pipelined requests is a subset of persistent requests. Similarly, the set of same-packet requests is a subset of pipelined requests.

We also investigate whether different operating systems or browsers influence our results. Therefore, we annotate our data sets accordingly and perform our analysis for each subset. We extract browser type and operating system information from HTTP user-agent strings using an open source parser [7].

Content Type Another potential error can be made by relying on the Content-Type header to identify the mime type of transferred files. To assess this error, we extract the Content-Type header and analyze the initial portion of the HTTP body with libmagic. We then compare the type reported by HTTP headers and libmagic and analyze cases in which they disagree.

Content Length and Download Volume HTTP servers can set a Content-Length header in the response indicating the size of the body. For persistent HTTP connections, the existence of either a Content-Length or a Content-Encoding header is mandatory. When measuring the volume downloaded via HTTP, one can choose to *(i)* measure the bytes transported on the wire, or *(ii)* use the Content-Length header of the HTTP response. The second option may introduce errors, caused by user interaction (e. g., interrupted downloads), software errors, or the lack of Content-Length headers. While canceling a download leads to larger values reported than actually downloaded, the lack of Content-Length headers will typically cause to ignore the request and therefore underestimate the volume. We define the estimation error as the ratio of the size announced by the Content-Length header and the actually downloaded volume. If an HTTP response does not include a Content-Length header, we define the ratio as 1.

[5] Bro terminates connections at control packets (SYN, FIN/RST) or an inactivity timeout.

[6] Packet capture is often configured with a snap-length so that the first lines of the HTTP header are captured. Often, such traces would include persistent requests but not second (pipelined) ones in the same packet.

[7] http://user-agent-string.info/download/UASparser

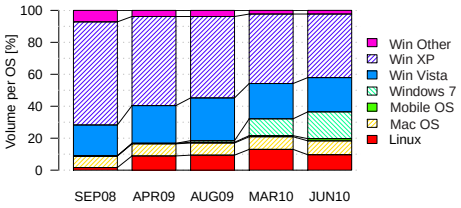

Fig. 1. Operating system popularity.

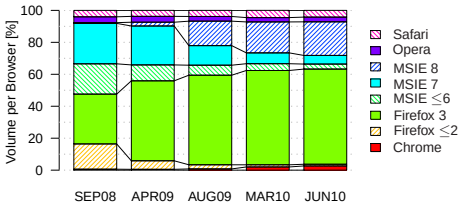

Fig. 2. Browser popularity.

Browser and OS Popularity In order to better understand the data sets and the monitored user population we present the popularity and distribution of operating systems and browser families. In Figure 1 we plot the traffic volume per operating system and in Figure 2 the traffic volume per browsers family. We only include popular, regular browsers (Internet Explorer, Firefox, Safari, Opera), and exclude other user-agents (e. g., software updates, media players).

In terms, of operating system popularity we see that Windows clearly dominates, with more than 80 % traffic volume. With time users are switching from older Windows versions to newer ones. Mac OS consistently accounts for 8 % while Linux increases its contribution from 2 % to 13 % in 2010.

When we look into browser popularity, we see that the majority of bytes are requested by Firefox users. Microsoft's Internet Explorer (MSIE) is the close second. As for OSes we see the adaption of new browser versions by users (e.g., Firefox 2 $\rightarrow$ Firefox 3), which significantly changes the contribution by browser *version*. In the overall share of browser families, Firefox gains 13 % and Google's Chrome rises to 2.5 %. While Opera and Safari remain at 3 % and 4 %, Internet Explorer declines.

3 Persistent and Pipelined Requests

3.1 Persistent Requests

We first determine the amount of persistent requests. In Figure 3 we plot the complementary cumulative distribution function (CCDF) of the number of requests per connection for JUN10, HTTP12d, and HTTP14d. We observe that 30 % of the connections (y-axis at $3 \cdot 10^{-1}$ in log-scale) have two or more requests. One connection in one thousand has more than 100 requests. We even observe connections with several tens of thousands of requests. The CCDFs for the other traces look similar. On average, each connection has 2.2 to 2.4 requests. This is consistent with Callahan et al. [2].

It also indicates a potentially high error for analyses that only consider the first request in a connection. The TimeMachine [7] suggests that "the most interesting information is in the beginning of a connection" and showed significant recording and analysis savings when only considering the first few bytes (the "cutoff") of a connection. If this cutoff is too low, successive persistent requests will be skipped from the analysis.

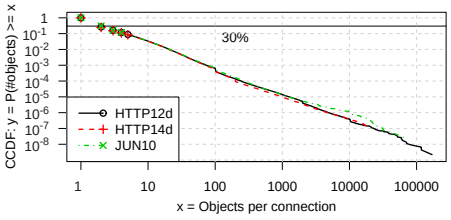

Fig. 3. CCDF of requests per connection for HTTP12d, HTTP14d, and JUN10.

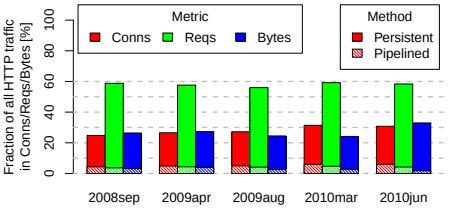

Fig. 4. Persistent requests/bytes/connections.

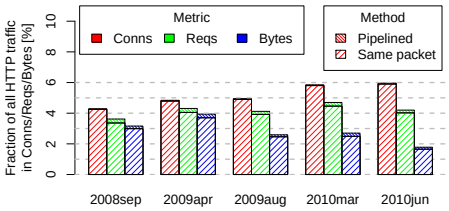

Fig. 5. Pipelined requests/bytes/connections.

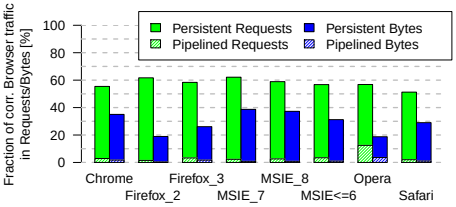

Fig. 6. Results per browser family for JUN10.

In Figures 4 and 5 we present the results of our three metrics—connections with (Conns) marked request, number (Reqs), and volume (Bytes) of marked requests—for the persistent, pipelined, and same packet requests for all 24h traces. Recall that pipelined is a subset of persistent and same packet is a subset of pipelined. Thus, the bars are not stacked, but overlay each other. Note the different scale on the y-axis. Again, we find the 30 % from Figure 3 for the combination of the persistent method with the Conns metric for JUN10.

Looking at persistent requests (solid bars in Figure 4), we do not observe significant changes over time/traces. The fraction of connections with persistent requests increases slightly from 25 % to 30 %. While the fraction of persistent requests is fairly constant around 60 %, the volume transferred in persistent requests is around 25 % with the exception of JUN10 where it is over 30 %.

Summary When only considering the first request in a HTTP connection, one would miss around two thirds of the request, and one third of the volume transferred.

3.2 Pipelined Requests

Pipelined requests (shaded bars in Figures 4 and 5) exhibit considerably lower values for all metrics. While the fraction of connections with pipelined requests increases from 4 % to 6 % from SEP08 to JUN10, the contribution in terms of volume of pipelined requests decreases from 4 % to below 2 %.

If a HTTP traffic analysis does look at multiple request in HTTP connections, but only includes those that are at the beginning of a packet, it would miss around 4 % of the requests and up to 4 % of the volume. Similarly, if the analysis assumes only one

request per packet and just greps for the last string that matches a certain header field in the packet, 4 % of the requests would be affected.

3.3 Impact of browser

Common wisdom[8] suggests that pipelining is disabled by default in the most popular browsers. Our traces on the other hand show a non-negligible amount of pipelined requests. We therefore drill down into the data and analyze it across browser families, operating systems, and content-types. In addition we select a number of—in our dataset—popular and high-volume web services for comparison.

We expect that browsers have a significant impact on the number of persistent and pipelined requests. While the server needs to support HTTP/1.1 for persistent and pipelined request, the browser ultimately has to issue the request. Callahan et al. [2] observed a drastic change in the number of requests per connection and attributes the change to a different default browser version in the environment they monitor.

In Figure 6, we show per browser results for the persistent and pipelined request marking methods for JUN10, as in Figure 4. We observe that the variance among the different browser categories is limited. Microsoft's Internet Explorer (MSIE), has around 10 % more persistent bytes than Firefox. Together, they account for 70 % of the volume and 80 % of the requests. Opera does stand out with an unusually high fraction of pipelined requests, which is expected as it is the only browser that has HTTP pipelining enabled by default. Yet, it comprises only about 2.5 % of the total requests and volume. We do find similar results for the per OS analysis. Linux and MacOS X have lower fractions of persistent bytes than Windows, although the fraction of requests is similar.

3.4 Impact of Web Service

We now select the 30 most requested and/or highest volume second level domains from our dataset and calculate our metrics as for the browser categories. We then group together domains from *(i)* the same web service (e. g., `facebook.com` and `fbcdn.com`) and *(ii)* similar types of pages when the results are similar: "OSN" consists of three locally popular online social networks (Jappy, Mein/StudiVZ, and Schueller.cc) with similar results and "Adult" consists of three video portals offering adult content. The fraction of persistent and pipelined requests and bytes of the resulting 18 web services are shown in Figure 7 for JUN10.

The plot shows more variations across web services compared to across browsers versions in Figure 6. Web services have a stronger influence on persistence and pipelining. The fraction of persistent requests ranges from 11 % (Uploaded.to) up to 88 % (WindowsUpdate), and the fraction of bytes in persistent requests even ranges from $<$1 % (MegaVideo and MegaUpload) to 95 % (again WindowsUpdate). In terms of pipelined requests we see maxima at 33 % for Microsoft and 11 % for RapidShare for requests and volume, respectively. These fractions are significantly higher than those we observe for Opera, which achieved the highest fraction so far. We do not observe a strong relation between the type of web service and persistence/pipelining. Consider

[8] Wikipedia (version 11 August 2011): `http://en.wikipedia.org/wiki/Pipelined_HTTP`.

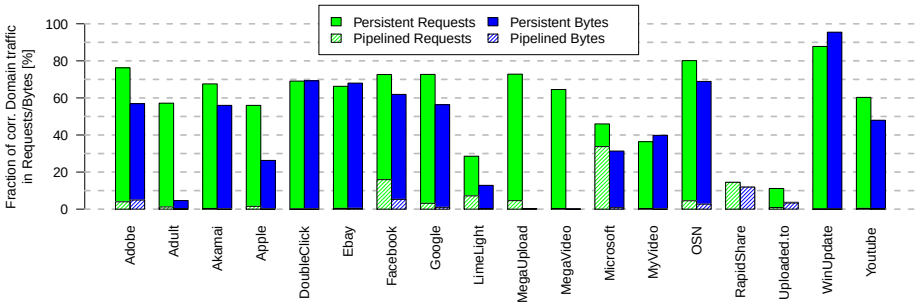

Fig. 7. Percentage of persistent/pipelined request/bytes for selected web services for JUN10.

for example the One-Click Hosters MegaUpload, RapidShare, and Uploaded.to. While MegaUpload has more than 70 % of persistent requests, the other two have barely more the 10 %. Moreover, all request that are persistent are also pipelined for RapidShare, yet the other two have almost no pipelined requests at all. The two CDNs (Akamai and LimeLight) also exhibit largely different fractions for persistent requests and volume.

The effects of different web services also translates into differences when examining the results by Content-Type. Similarly to YouTube, MegaVideo, and MyVideo, flash-video (`video/flv`) has less than 1 % pipelined requests and bytes. On the other hand, `application/rar` has the largest amounts of both pipelined requests and bytes, being mostly delivered by RapidShare.

The limited impact of browsers on persistence and pipelining can be explained by two trends in web content delivery. First, distributed content delivery infrastructures can prevent persistent requests. Second, more and more web services rely on service-supplied code to be executed in the browsers, such as AJAX-based clients [9], which issue the HTTP requests instead of the browser. For example, in all traces except JUN10 the fraction of persistent bytes for YouTube never exceeded 4 % and but is at 47 % for JUN10. A likely explanation for this change is prolonged server restructuring after Google's acquisition of YouTube in 2006: Over all the traces we observe the `googlevideo.com` domain emerge and vanish again. Evidence for the second explanation can be seen from the high fraction (15 %) of pipelined request in Facebook, which heavily uses AJAX, as it is unlikely that all these requests are issued only from Opera browsers.

4 Content Type

We next turn our focus to HTTP's Content-Type header. In Figure 8 we plot the fraction of HTTP bytes for which the Content-Type header and an analysis by libmagic disagree. We normalize the mime type strings by removing leading `x-` but otherwise perform a string comparison between the header and libmagic's results. We find that up to 36 % of HTTP volume exhibits mismatches. The most common case with up to 27 % is when the Content-Type header uses a generic type (e.g., application/octet-stream or text/plain[9])

[9] We only count text/plain as generic if the Content-Type header specifies text/plain and libmagic yields a non-text type.

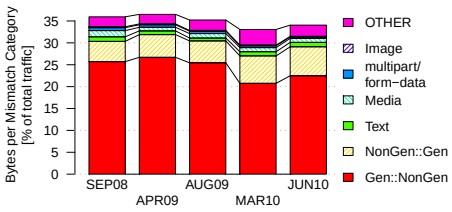

Fig. 8. Mismatches between Content-Type and libmagic as fraction of HTTP volume.

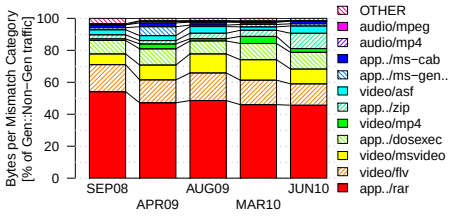

Fig. 9. Generic Content-Type but specific libmagic type: breakdown by libmagic class

but libmagic yields a known type. We label this class of mismatches Gen::NonGen. The opposite case (NonGen::Gen, in which the Content-Type specifies a type but libmagic fails to detect one is the second most common one. We observe that 4.6–6.6 % of HTTP bytes fall into this class. While some of the types are not supported by libmagic, e. g., Google safe browsing chunks, others include audio or video formats that should be supported by libmagic. It is hard to exactly assess whether libmagic or the Content-Type header are "correct" in such cases. In the Text, Media, and Image classes the Content-Type header agreed on the general category of the type (e.g., image) but disagreed on the actual file format (JPEG vs. PNG). While libmagic should be accurate for image data, other media (audio, video) is harder to assess, given that there is a plethora of different container-types and codecs often with similar names.

When we investigate the largest class, Gen::NonGen, in more detail, we find that (in terms of bytes) RAR-archives are responsible for around 50 % of these mismatches, see Figure 9. In addition, we find that 50–74 % of all bytes with Content-Type header of text/plain are used as a generic type, i.e., libmagic indicates a non-text type (not plotted). In such cases, one would incorrectly infer a significantly higher fraction of text/plain than is actually the case.

We note that we excluded cases of mismatches for which the Content-Type header specified Javascript or CSS and libmagic yielded another text type (e.g., C-code). Such types are inherently hard for libmagic to classify correctly since the syntax of these languages is similar to C. Up to 4.1 % of bytes fall into this category.

An analysis relying on the Content-Type header alone would thus be unable to classify up to 27 % of HTTP bytes, since the Content-Type header is generic and the analysis would also over estimate the amount of text/plain content due to the frequent use of this type as a generic type. Up to an additional 10 % of HTTP bytes further show other disagreements. While it might be possible to roughly classify some of these (e.g., as video content), a more detailed breakdown (e.g., what kind of video) appears challenging.

5 Content Length

The Content-Length header is commonly used to analyze the size of HTTP transfers [4]. Unfortunately, this can lead to errors if the header size does not accurately reflect the downloaded volume, e. g., due to software bugs or interrupted downloads. In this section, we quantify the extent of the overall error and characterize its variance over

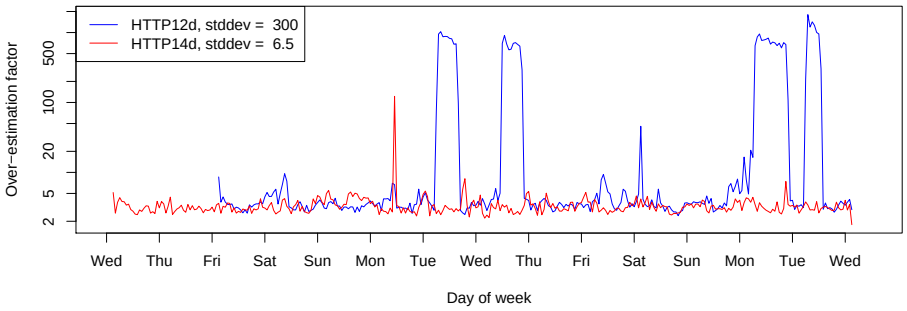

Fig. 10. Over-estimation factor of the Content-Length header (60 min bins, logarithmic y-axis).

time. We choose the HTTP14d and HTTP12d traces for this analysis to show the time-dependent behavior.

We find that the Content-Length headers over-estimate the actually downloaded volume by a factor of 3.65 for HTTP14d and a factor of 127 for HTTP12d. A closer examination of HTTP12d shows that a single user downloading two large files from a single host with a badly (mis-)configured download manager is the culprit. This download manager opened over 400,000 connections for each of these files to parallelize the download and requested large, overlapping byte ranges. However, the download manager aborts each download after receiving enough data to cover the whole file. Overall, the requested download volume from Content-Length headers sums up to over 4 PB, an over-estimation by a factor of more than 60,000. After removing these two files from the data set the over-estimation factor for HTTP12d drops to 3.82. We observe a similar case in HTTP14d, though to a far lesser extent and involving only a single file. After removing the corresponding file, the over-estimation factor reduces from 3.65 to 3.28 for HTTP14d.

This highlights how much the over-estimation factor depends on events of limited duration. This leads us to further investigate this volatility over time. Figure 10 plots the over-estimation factor for 60 minute bins. We align our traces by day of week. In general, the over-estimation factor is between 2.2 (2.4 for HTTP12d) and 5 for each 60 minute bin. However, we observe spikes exceeding these baselines by several orders of magnitude. Furthermore, we see the mis-configured download manager in HTTP12d that causes the over-estimation factor to rise to 500 to 2000 for several hours.

We note that based on these 60 minute time bins, the standard deviation for both traces exceeds the average by far. Moreover, there is no apparent weekly or daily pattern. Accurate HTTP object size measurement therefore requires to parse the whole HTTP stream.

6 Conclusion

In this paper we identify and investigate three potential pitfalls in HTTP traffic analysis. We study the accuracy of information in Content-Length and Content-Type headers, as well as the amount of persistent and pipelined traffic.

Our results indicate a significant over-estimation, at least 3.2 times, when relying on the HTTP Content-Length header for volume inference. For accurate volume accounting, complete processing of the data after the HTTP response header is required to detect transfer abortions, erroneous HTTP servers, and misconfigured download managers. The mismatch between Content-Type header and libmagic content types amounts to 35 % of the HTTP volume. Relying on the Content-Type header for content classification can lead to a significant amount of unclassified content due to a generic Content-Type, and to a lesser degree in misclassification of the content. Finally, only analyzing the first packet of a connection discards 60 % of the total HTTP requests and 30 % of the HTTP volume. Simplifying the analysis by capturing just enough bytes per packet to include HTTP headers leads to another risk: We find 4 % of the requests to be pipelined and transmitted together with the previous request in a single packet.

As future work, we plan to further analyze the use of pipelining and persistence by different web services and applications, especially with respect to application design and content delivery.

7 Acknoledgements

This work was partly supported by a fellowship within the postdoctoral program of the German Academic Exchange Service (DAAD).

References

1. AGER, B., SCHNEIDER, F., KIM, J., AND FELDMANN, A. Revisiting cacheability in times of user generated content. In *Proc. of IEEE Global Internet Symposium* (2010).
2. CALLAHAN, T., ALLMAN, M., AND PAXSON, V. A longitudinal view of http traffic. In *Proc. Conference on Passive and Active Measurement (PAM)* (2010).
3. DOVERSPIKE, R., AND GERBER, A. Traffic Types and Growth in Backbone Networks. Tech. rep., In Proc. of OFC/NFOEC (invited paper), March 2011.
4. ERMAN, J., GERBER, A., HAJIAGHAYI, M. T., PEI, D., AND SPATSCHECK, O. Network-aware forward caching. In *Proc. International World Wide Web Conference (WWW)* (2009).
5. LABOVITZ, C., IEKEL-JOHNSON, S., MCPHERSON, D., OBERHEIDE, J., AND JAHANIAN, F. Internet inter-domain traffic. In *Proc. ACM SIGCOMM Conference* (2010).
6. MAIER, G., FELDMANN, A., PAXSON, V., AND ALLMAN, M. On dominant characteristics of residential broadband internet traffic. In *Proc. Internet Measurement Conf. (IMC)* (2009).
7. MAIER, G., SOMMER, R., DREGER, H., FELDMANN, A., PAXSON, V., AND SCHNEIDER, F. Enriching network security analysis with time travel. In *Proc. ACM SIGCOMM Conference* (2008).
8. PAXSON, V. Bro: A system for detecting network intruders in real-time. *Computer Networks Journal 31*, 23–24 (Dec. 1999), 2435–2463. Bro homepage: `http://www.bro-ids.org`.
9. SCHNEIDER, F., AGARWAL, S., ALPCAN, T., AND FELDMANN, A. The new web: characterizing ajax traffic. In *Proc. Conference on Passive and Active Measurement (PAM)* (2008).
10. SCHNEIDER, F., FELDMANN, A., KRISHNAMURTHY, B., AND WILLINGER, W. Understanding online social network usage from a network perspective. In *Proc. Internet Measurement Conf. (IMC)* (2009).