

The New Web: Characterizing AJAX Traffic

Fabian Schneider, Sachin Agarwal, Tansu Alpcan, and Anja Feldmann

Deutsche Telekom Laboratories / Technische Universität Berlin
10587 Berlin, Germany
{fabian,anja}@net.t-labs.tu-berlin.de,
{sachin.agarwal,tansu.alpcan}@telekom.de

Abstract. The rapid advent of “Web 2.0” applications has unleashed 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. We present a traffic study of several Web 2.0 applications including Google Maps, modern web-email, and social networking web sites, and compare their traffic characteristics with the ambient HTTP traffic. We highlight the key differences between Web 2.0 traffic and all HTTP traffic through statistical analysis. As such our work elucidates the changing face of one of the most popular application on the Internet: The World Wide Web.

1 Introduction

The World Wide Web [1] is one of the most popular applications of the Internet and runs primarily over the HTTP protocol. While HTTP (Hyper Text Transfer Protocol) [2] 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 web page in response to the respective request.

The recent popularity of asynchronous communication enabled web sites 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) [3], 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 paper we use AJAX and “Web 2.0” interchangeably to refer to web applications that use this new paradigm on the Internet.

Contemporary web pages 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

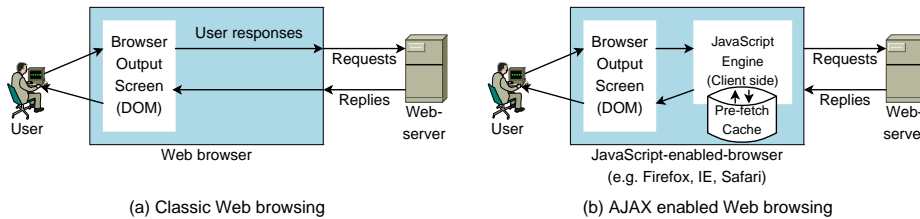


Fig. 1. Comparison of classical with AJAX enabled web applications.

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 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 the web server. The prediction algorithms of any automated pre-fetching scheme usually results in significantly larger downloads as compared to user-initiated web browsing due to inaccurate guesses on part of the prediction algorithms about which data to pre-fetch. Even when the prediction is accurate, HTTP traffic inter-request-times 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 adopter 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 web sites use AJAX technologies to offer rich and interactive user experiences. In this paper 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.

1.1 Related Work

A good overview of traditional Web protocols is given in the book by Krishnamurthy and Rexford [1]. One of the early works on characterizing the effect of HTTP traffic and HTTP pre-fetching is by Crovella [4]. 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., [5–7]. However, the underlying motivation for using caching in all these studies has been on reducing the overall download latency of popular web sites 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 web sites about the end-user perceived latency of AJAX based applications (e.g., [8]). The novel aspect of our work is that

we focus on the behavior of two large user populations and investigate multiple AJAX enabled applications.

1.2 Contributions

In this paper, 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 request (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 paper 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 2. In Section 3 we present the results of our statistical analysis comparing AJAX traffic with the HTTP traffic. Finally, we conclude in Section 4.

2 Methodology

In order to determine which Web 2.0 applications to study we first examine the popularity of different applications (Section 2.1). Google Maps is among the most popular web applications and a nice example of a AJAX-enabled application. Therefore, we provide a high level overview of its communication patterns in Section 2.2. Finally, we detail how we extract application characteristics from our data sets in Section 2.3. Similar extraction methodologies (skipped for brevity) are used for the other AJAX applications.

2.1 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, MWN-05 and MWN-07, are from MWN while

Table 1. Characteristics of the data sets

Trace	Start Date	Duration	Size	#Req Total	#Req GMaps
MWN-07	Feb 24th 2007	32h+	2.4 TB	30,0 M	222 K
LBNL-07	Mar 3rd 2007	~9h	214 GB	2,0 M	82 K
MWN-05	Oct 11th 2005	24h	2.5 TB	119 M	43 K

one trace, LBNL-07, is from LBNL. See Table 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 [1], we utilize the HTTP analyzer of Bro [10], 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 timestamps (request issued, cookie seen, request finished), requested hostname (servername¹), 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 servers² in the MWN-07 data set. We then 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 web sites (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 Web sites, e.g., Ebay and MSN. Figure 2 shows a pie chart of the number of requests per category for the MWN-07 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 web site is lokalisten.de with 1.4%. Although Google Mail only accounts for 0.65% of the requests we include it as our fourth applications since this gives

¹ We use server and host interchangeably in this discussion.

² Web server as specified by the hostname in the HTTP request.

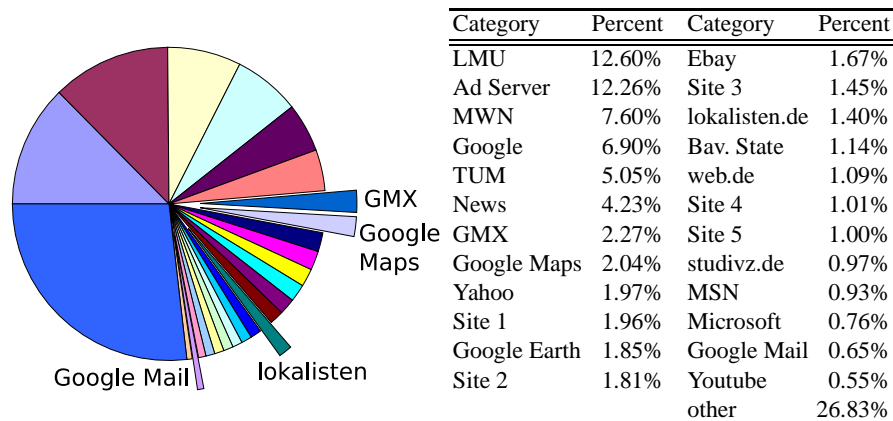


Fig. 2. Pie chart of the percentages of requests for the top 500 hostnames by categories. (Top 500 hostnames account for 53% of the total requests. Percentages are relative to the top 500.)

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 paper are among the top 500. We refer to these both most popular and AJAX-enabled applications as “Selected-4” in subsequent discussions.

2.2 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) [9] of the web browser such that it is no longer necessary to reload the entire web page 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 [2] 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.

2.3 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 on the same back-end server infrastructure (e.g., Google Maps, Google Search,

Google Video, etc.) 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. Separating Google Maps traffic from other Google services thus 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 ID³. 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 timeout⁴ 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 (IRT’s) for the Selected-4 applications as well as ALL-HTTP traffic.

3 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

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.

Table 2. Mean/Median Table for ALL-HTTP and Selected-4 applications in the MWN-07 data set. IRT's are Inter-Request(-arrival)-Times.

Application	#Requests	#Sessions		Bytes per Connection	Bytes per Session	#Req per Connection	#Req per Session	IRT's in a Connection	IRT's in a Session
ALL-HTTP	30 M	1.4 M		57890	278K	4	13	2.34	17.23
Google Maps	221 K	1127	mean	204476	2288K	18	197	1.39	1.54
lokalisten.de	128 K	3822		31856	129K	8	34	0.38	4.52
Google Mail	140 K	1020		9742	371K	4	138	23.02	31.84
GMX	288 K	6101		14163	95K	7	47	0.53	4.29
ALL-HTTP			median	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

and compute the PDFs of the logarithm of the data in order to be able to use a logarithmic X-axis. In addition, Table 2 presents mean and median values.

In our analysis we concentrate on the MWN-07 data set and only use the MWN-05 and LBNL-07 data sets to highlight some of the noticeable differences. Note that the 2005 data set was collected during Google Maps beta testing phase.

Figure 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 MWN-07 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 PDF for Google Maps and Mail as well as ALL-HTTP traffic in Figure 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 MWN-07 Google Maps connections comprise of connections that transfer less than 10 KB, whereas 81.8% of the ALL-HTTP connections from MWN-07 transfer less than 10 KB. Similar observations hold for the LBNL-07 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 [11].

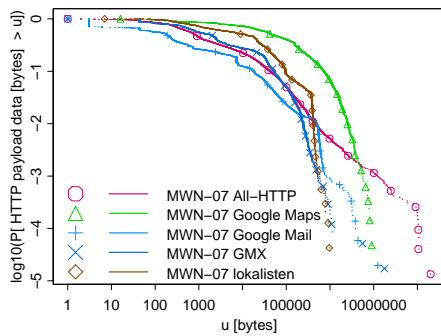


Fig. 3. HTTP payload bytes per connection.

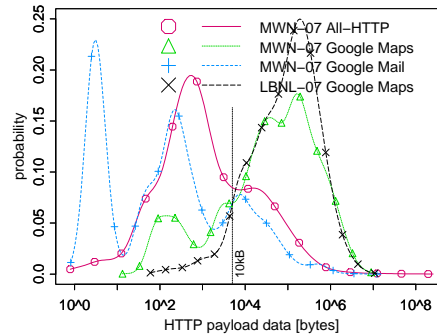


Fig. 4. HTTP payload bytes per connection.

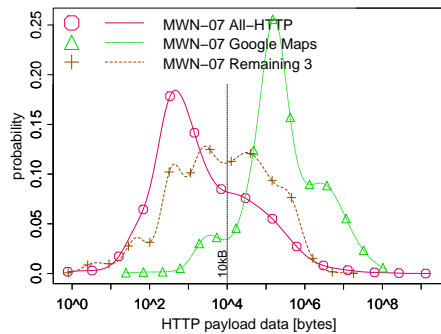


Fig. 5. HTTP payload bytes per Session

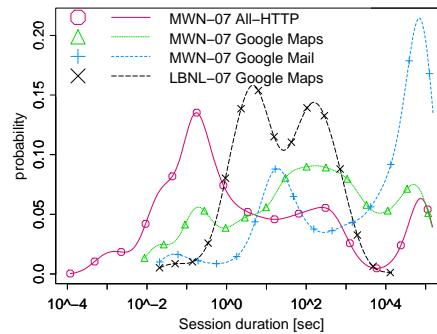


Fig. 6. Session durations

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 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 7 and 8 show the CCDF and PDF for ALL-HTTP and Selected-4 sessions in the MWN-07 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 2 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 6), is shorter than for AJAX enabled applications. Half of the ALL-HTTP sessions last between 0.008 and 2.13 sec-

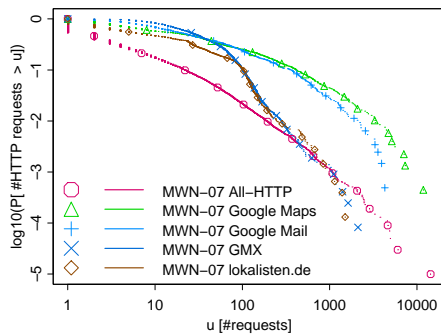


Fig. 7. Number of requests per session.

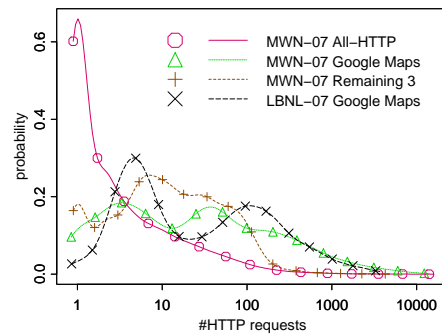


Fig. 8. Number of requests per session.

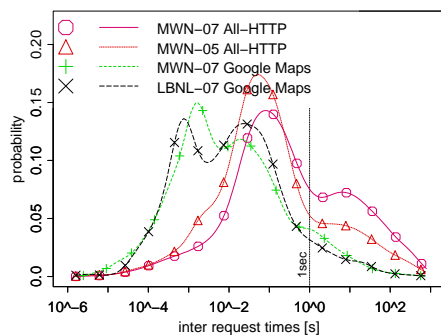


Fig. 9. PDF of inter-request-times within each session: ALL-HTTP and Google Maps.

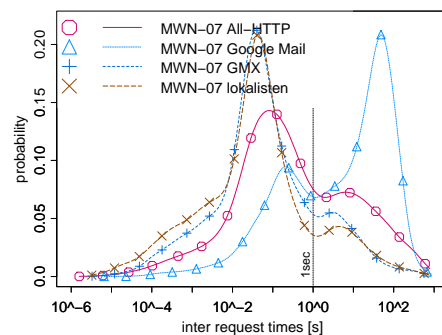


Fig. 10. PDF of inter-request-times within each session: Google Mail, GMX and lokalisten.de.

onds (5% – 55% quantile across all sessions) while 50% of Google Maps sessions in the MWN-07 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 web site. Overall these characteristics indicate that AJAX enabled applications last longer and are more active than ALL-HTTP sessions.

Finally, Figures 9 and 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 MWN-07 and LBNL-07. 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 are 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 Google Mail uses a polling interval of roughly 120 seconds (those 3 Bytes requests from Figure 4). Once these are removed the densities are quite similar again.

4 Conclusions

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 paper 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 paper may help web application developers in understanding how their applications affect Internet traffic, and how their applications can be designed for more efficient operation.

References

1. Krishnamurthy, B., Rexford, J.: Web protocols and practice: HTTP/1.1, Networking protocols, caching, and traffic measurement. Addison-Wesley (2001)
2. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Rfc 2616, hypertext transfer protocol – http/1.1 (1999)
3. Zakas, N., McPeak, J., Fawcett, J.: Professional AJAX. Wiley (2006)
4. M. Crovella, P.B.: The network effects of prefetching. In: INFOCOM. (1998)
5. Abrams, M., Standridge, C.R., Abdulla, G., Williams, S., Fox, E.A.: Caching proxies: limitations and potentials. In: WWW Conference. (1995)
6. Barford, P., Bestavros, A., Bradley, A., Crovella, M.E.: Changes in Web client access patterns: Characteristics and caching implications. World Wide Web (1999)
7. Challenger, J., Iyengar, A., Danzig, P.: A scalable system for consistently caching dynamic Web data. In: INFOCOM. (1999)
8. The impact of AJAX on web operations (2005) <http://www.bitcurrent.com/?p=105>.
9. Document Object Model (DOM) (2007) <http://www.w3.org/DOM>.
10. Paxson, V.: Bro intrusion detection system (2007) <http://www.bro-ids.org>.
11. Feldmann, A., Rexford, J., Caceres, R.: Efficient policies for carrying Web traffic over flow-switched networks. IEEE/ACM Trans. Networking 6(6) (1998)