A Longitudinal View of HTTP Traffic*

Tom Callahan†, Mark Allman‡, Vern Paxson‡,¶

†*Case Western Reserve University*, ‡*International Computer Science Institute*
¶*University of California, Berkeley*

Abstract. In this paper we analyze three and a half years of HTTP traffic observed at a small research institute to characterize the evolution of various facets of web operation. While our dataset is modest in terms of user population, it is unique in its temporal breadth. We leverage the longitudinal data to study various characteristics of the traffic, from client and server behavior to object and connection characteristics. In addition, we assess how the delivery of content is structured across our datasets, including the use of browser caches, the efficacy of network-based proxy caches, and the use of content delivery networks. While each of the aspects we study has been investigated to some extent in prior work, our contribution is a unique long-term characterization.

1 Introduction

In this paper we study logs of web traffic collected at the border of a small research institute over a three and a half year period (2006–mid-2009). There are an average of 160 active users per month in our dataset. While this is a relatively small population, we gain insight into the evolution of web traffic by taking a longitudinal view of the traffic. This investigation serves to re-appraise and update previous results. We believe our contribution has utility in informing the community's mental models about myriad aspects of how the modern web works—including things like transaction types and sizes, as well as how web content delivery is accomplished through content delivery networks, browser caches and the like. In addition, a multi-faceted view of web content delivery is useful in setting up realistic testbeds and simulations to accurately reflect the make-up and structure of today's web.

Our methodology employs web traffic logs from our intrusion detection system collected over three and a half years to study various aspects of the web. We describe our data collection and analysis methodology in § 2. We then characterize a number of facets of the traffic at the transaction-level in § 3. We next consider various aspects of user-driven behavior, such as object popularity and the impact of caching in § 4. Finally, we consider the structure of the web page delivery process, including the use of CDNs in § 5. We briefly touch on related work in § 6 and summarize in § 7.

2 Data and Methodology

For this work we use logs of web traffic taken at the border connecting the International Computer Science Institute (ICSI) with its ISP. We use the Bro intrusion detection system [12] to reconstruct HTTP [7] sessions from the observed packet stream.

* This work is supported in part by NSF grants CNS-0831535 and CNS-0831780.

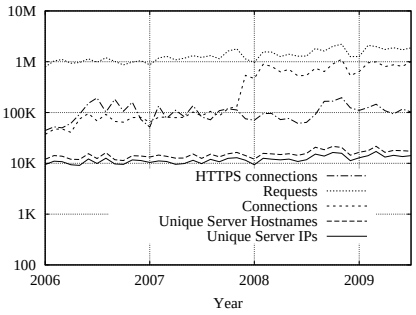

Fig. 1. Dataset Summary

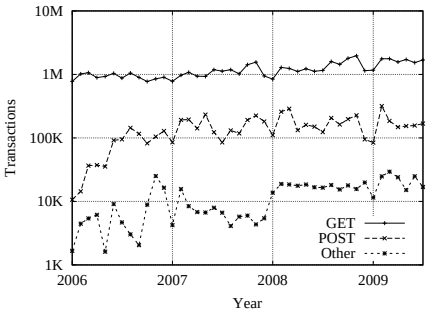

Fig. 2. HTTP Transaction Types

These sessions are then logged using Bro's standard HTTP logging policy (found in `http.bro` in the Bro package). The logs include timestamps, involved IP addresses, URLs, HTTP transaction types and sizes, hostnames and HTTP response codes. The dataset used in this paper runs from January 2006 through July 2009. Due to the size of the dataset we analyze only the first seven days of each month for logistical reasons. We do not believe this biases our results. The original logs include all incoming and outgoing HTTP traffic. However, we winnowed to only the outgoing connections (i.e., ICSI clients) as we do not wish to bias our results by the particular characteristics of the few server instances at ICSI. Of the 28.8 million total connections from the first seven days of each month in our dataset we retain 16.9 million as initiated by ICSI clients. In figure 1 we show the high-order characteristics of the dataset. The overall number of web object requests, HTTP connections, HTTPS connections, server hostnames and server IP addresses show general stability over time.[1] Since the HTTPS connections are encrypted we cannot further analyze them in this work. We identify "web servers" in two different ways: by IP address and by hostname. Due to the use of content delivery networks (CDNs) a particular IP address may host content for multiple distinct hostnames. In fact, in the figure we see this effect as there are more server hostnames than server IP addresses (this is studied in more detail in § 5). However, note that the opposite is also true: that a given hostname could have multiple IP addresses (e.g., to serve content from a close source or for load balancing). Finally, we note that the number of users is modest—an average of 160 per month with a standard deviation of 13—our contribution is the longitudinal tracking of this user population.

Finally, we note that there are two versions of Bro HTTP policy scripts used in gathering the data we employ in this study with one crucial difference. For the first ten months of 2006 the scripts gathered logical web sessions together as one logical entity under one identifier regardless of the number of underlying TCP connections used to obtain the components of the web pages. In these log files this process obscures the number of TCP connections used to transfer the data. Due to the onerous amount of state required to stitch together a web session from disparate TCP connections, starting

[1] The number of connections has a dramatic increase in December 2007. We delve into this in detail in § 5.

in mid-October 2006 the scripts were changed to simply gather together all activity on a per-TCP connection basis. For most of our analysis the difference in logging is not important, but for analysis that requires an understanding of the number of underlying TCP connections we start our analysis in November 2006 instead of January 2006. In figure 1 the reported number of connections for the first 10 months of the dataset is actually the number of web sessions (which is reported to give the reader context even though the precise number of connections is unknown).

3 HTTP Transaction Characterization

We first focus on characterizing client HTTP transactions. First, figure 2 shows the transaction type breakdown over time. Over the course of our dataset the majority of observed transactions—approaching 90% in most months—are requests for data (`GET` transactions). Most of the remainder of the transactions—around 10%—involve the user uploading data to the web server (`POST` transactions). A small number of additional transaction types are also observed (`HEAD`, `PROPFIND`, etc.). Together these additional types account for less than 1% of the transactions in most months. We note that in absolute terms the number of `GET`s and `POST`s have a slight increasing trend over our observation period (note, the figure is plotted on a log scale and therefore the increase is less readily apparent). Further, the number of `POST`s observed increases quickly at the beginning of our dataset. This is caused by a dramatic uptick in the use of GMail during early 2006. Non-GMail `POST` requests are more steady and only slowly increasing during this period.

Figures 3 and 4 show the average and median size of `GET` and `POST` transactions over time. Both transaction types show a generally increasing average transaction size over time which is likely explained by users both increasingly downloading richer content and participating in so-called web 2.0 sites that host user-provided content. The median results for `POST` transactions are interesting as they remain small and fairly constant over the study period. This indicates that simple form input that only results in the transmission of a small amount of data is prevalent throughout. For the `GET` requests we find the medians are generally an order of magnitude less than the averages. This is expected due to many previous studies that show most responses are short and a few responses carry most of the bytes—i.e., web traffic is heavy-tailed [5]. Figure 5 shows the distribution of `GET` response sizes for July 2 2007 as a typical example of the per-day distribution (this date was chosen arbitrarily as a weekday roughly in the middle of the study period). Finally, we note that the average `GET` response size in December 2006 is four times the size of the surrounding months. This anomaly is caused by a single client fetching a large series of big files. We removed this client from our analysis and plot a point on the graph to show the average size of `GET` responses without this particular client. Without the energetic client the average is similar to the surrounding months.

Figure 6 shows the median duration of HTTP connections, as well as the median time between establishing a connection and the client issuing an HTTP request. We note that the median connection duration is reduced between November and December 2007 which is explained by a reduction in the use of persistent HTTP connections (see § 5). As connections are used for fewer objects their duration drops. Before December 2007

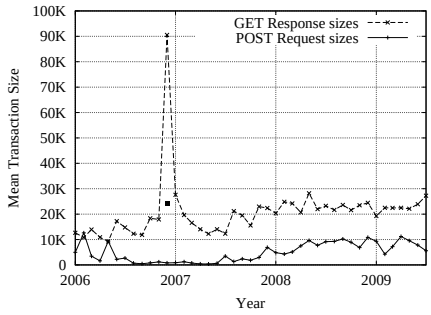

Fig. 3. Average transaction sizes

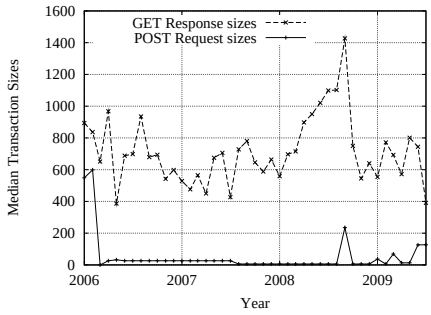

Fig. 4. Median transaction sizes

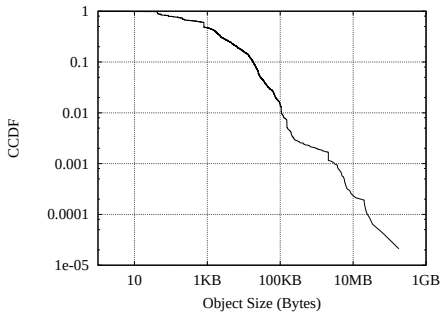

Fig. 5. CCDF of `GET` sizes for July 2 2007.

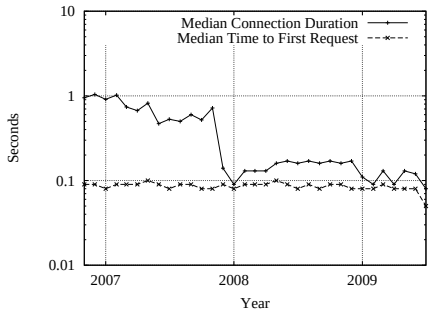

Fig. 6. Connection duration and request time.

the median connection duration was around 1 second and after this point the median duration falls to 100–200 msec. The short duration of connections suggests that seemingly small changes to the delivery process that save modest amounts of wall-clock time may ultimately benefit the user experience more than one might think at first blush—e.g., Early Retransmit [2] and reducing TCP's traditional exponential backoff between retransmissions [10].

Figure 6 also illustrates the time between establishing a connection and sending an HTTP request. In related work [1] we study claim-and-hold attacks on web servers whereby a malicious client opens a connection and does not send an HTTP request to force the server to allocate resources that can then not be used for legitimate traffic. In figure 6 we show that the median time before an HTTP request is issued is roughly constant—at just under 100 msec—in our dataset, which agrees with the results in [1]. However, we also note that we find successful transactions whereby the time between connection establishment and transmission of the HTTP request is quite a bit longer. In particular, we find this with GMail. The 99^{th} percentile interval is roughly 246 seconds for GMail in each year, while the interval ranges from 14 seconds in 2006 to 55 seconds in 2009 for non-GMail traffic. This indicates that expecting short intervals may not be the right model for newer web application-driven web pages.

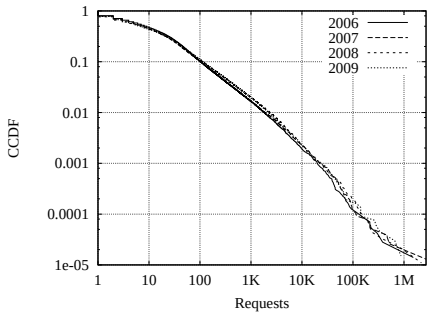

Fig. 7. Requests per hostname

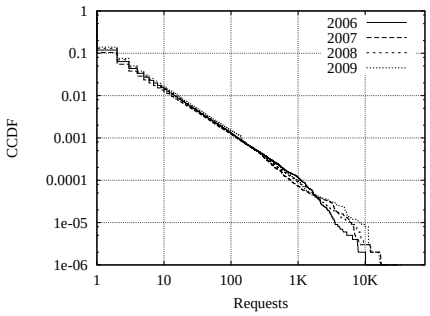

Fig. 8. Requests per object

4 User Behavior

Figure 7 shows the distribution of requests per server hostname. As discussed in § 5 all requests to a server name such as "www.cnn.com" are grouped together in this plot no matter what IP address handles the request. We see that across the years in our study the distribution of requests per hostname is similar. Many server hostnames are accessed infrequently—with the median being less than 10 requests per year. However, a small number of server hostnames are quite popular—with the maximum number of requests per hostname exceeding one million per year. We also observe that there is change in the top hostnames over time. We determined the top ten hostnames per year and find that four hostnames are within the top ten for all four years of our survey, one hostname is in the top ten in three of the years, two hosts appear in the list for two years and 17 hostnames appear only once across the four years.

Next we turn to object popularity. Figure 8 shows the distribution of the number of requests per object. The distributions are similar across the years. As seen in the plot, request popularity fits exceedingly well to a Zipf-like distribution. Across more than 3 orders of magnitude, the fall-off in the distribution matches closely to a Pareto distribution with $\alpha = 1$. Around 90% of the objects are accessed only one time.[2] Further, only a small number of objects are fetched more than 10 times. That said, there are some popular objects that are requested thousands of times over the course of a year.

Next, figure 9 shows the distribution of the unique number of objects (determined by URL) per hostname over time. We again observe stability across the years of our study. This plot shows that one-third of the hostnames we encounter (regardless of year) serve only a single unique object. Further, two-thirds of the hostnames serve ten or fewer objects. Similar to many other aspects of web traffic a small number of hostnames serve many web objects. For instance, roughly 5% of the hostnames provide more than 100 objects and hostnames top out at providing over one million objects.

Object popularity has a direct bearing on the usefulness of caching. We use our data to investigate both visible end-host caching and also potential savings from a network-based cache with our results illustrated in figure 10. First, we look at the number of

[2] As unique sets of parameters at the end of a URL generally yield distinct outputs, we consider the entire URL, including parameters, as a distinct object.

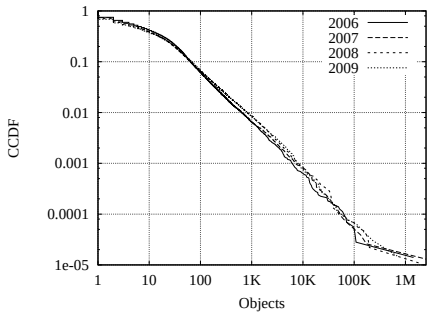

Fig. 9. Unique objects per hostname

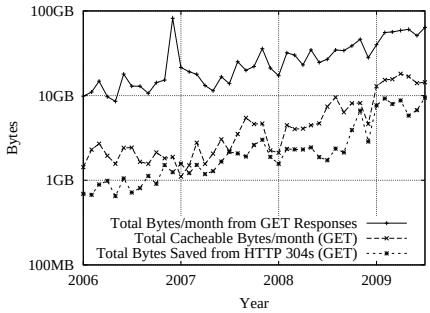

Fig. 10. Cond. GET and caching savings

bytes visibly saved by end-host caches using conditional GET requests. These requests call for a client re-requesting an object in the browser cache to include the timestamp of this cached object in the GET request. If the object has been updated the server will re-send it and otherwise will send a 304 ("not modified") response to the client, signaling that the version the client has is correct. As shown in the figure the number of bytes fetched would increase by roughly 10% without these conditional GET requests. The use of conditional GETs across the dataset increases in roughly the same manner as the overall number of bytes downloaded.

We next investigate the number of additional bytes that could benefit from caching at the institute-level—i.e., with some shared proxy cache at the border shared by all users. We first identify unique objects using the URL (including hostname) and object size. Ideally we would include a stronger notion such as a hash of the content itself, but our logs do not contain such information. We crunched 5.75 days worth of full-payload packet traces from January 2010 to assess the accuracy of our heuristic. We form tuples of URLs and sizes, (u, s), from the packet traces. In addition, for each we compute an MD5 hash of the object. For each (u, s) we record the number of MD5s observed. We find that 1.4% of the 846K (u, s) pairs in the trace have multiple MD5s. Further, we find this represents 6.7% of the bytes fetched over the course of the trace. Therefore, while we assume that if the object size stays the same the object has not changed this is wrong in a small number of cases. In addition, we calculate the amount of cachable data for each month in isolation and without any notion of timing out the objects or imposing a limit on the number or size of objects held in the cache. Therefore, our results are an upper bound on how an actual institute-wide cache would work with real-world constraints and a better understanding of the uniqueness of objects. As shown in figure 10 the percentage of bytes that could be cached by a network-wide proxy is 10–20% across most of our study. The number of cachable bytes does increase in 2009—when generally more than 25% of the GET requests could plausibly be handled by a cache. In 2009 to get the caching benefit suggested in the plot would require over 10 GB of storage space—which is quite modest in modern servers. We note that our caching results may be quite different if the population of users was larger.

5 Server Structure

Web traffic is composed of a series of HTTP transactions that in turn utilize TCP for reliable data transfer. HTTP uses one transaction per web object. Early HTTP used a separate TCP connection for each HTTP transaction, however with persistent HTTP connections an arbitrary number of HTTP transactions can be conducted over a single TCP connection. Figure 11 shows the total number of TCP connections we observe for the sampled week of each month of our dataset, as well as the average number of HTTP transactions per TCP connection. (Note, since these results depend on a solid understanding of connections, we do not include data before November 2006 as discussed in § 2.) The plot shows a fairly stable number of connections and average transactions per connection rate except for one impulse between November and December 2007. At that point we observe the re-use of TCP connections dropped by an order of magnitude. This results in a dramatic increase in the number of web connections observed. Note, figure 1 shows that the overall number of HTTP requests does not differ greatly across this event. That is, the same amount of content is being transferred, but the particulars of the underlying delivery has changed.

We believe this change in delivery pattern is due to a software change on the web clients. To verify this we determined the top 100 servers in each month by the overall number of requests (regardless of number of connections). For each server we then calculate the average number of transactions per connection. We find 66 servers to be common across the top 100 lists from the two months. We then calculate the difference in the average number of requests per connection for each of these 66 servers and find that in all but one instance the average drops in December. And, in over 70% of the cases the average requests per connection drops by at least 10 requests. This indicates that the use of persistent connections has dropped across the board and is not caused by some popular server curtailing support for persistent connections or some heavy-hitter client. We therefore conclude that this is a client policy change that is quite likely caused by a institute-wide web browser upgrade.[3]

We next turn our attention to how web sites are structured to serve content. Figure 12 shows the distribution of the number of hostnames each server IP address takes on over the course of each year in our study. CDN hosts can accommodate a wide range of logical hostnames using a server with a single IP address. The distributions are similar across the years with around 80% of the server IP addresses we encounter mapping to a single server hostname. While roughly 10% of the server IPs map to two hostnames we observe a small percentage ($\approx$5%) of server IPs accommodating three or more IP addresses. Further, there are a handful of IP addresses that serve traffic for a large number of hostnames. Our data shows that the maximum number of hostnames observed for a single IP address is 477, 878, 1784 and 1353 for the years 2006–2009 respectively. This shows a definite increase over time. (Note, since the data only covers half of 2009 the number may well increase when the entire year is considered.)

[3] The ICSI system administrators report a minor version upgrade of Firefox (from 2.0.0.8 to 2.0.0.10) during this timeframe. While we find nothing in the Firefox change-log that indicates a difference in the use of persistent HTTP connections we believe the defaults more than likely changed in 2.0.0.10 given the observed behavior so dramatically changes at the time of the upgrade.

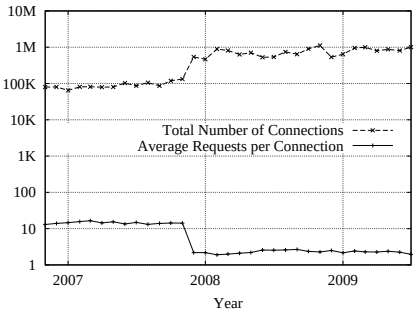

Fig. 11. Conns. / week and requests / conn.

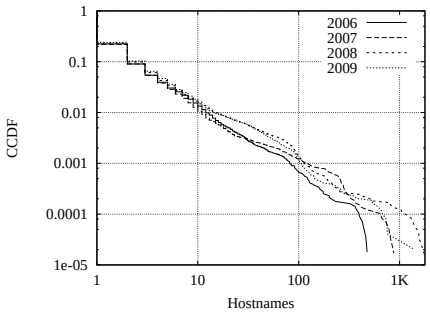

Fig. 12. Unique hostnames per IP

In addition to using one IP address to serve content from multiple server hostnames CDNs also use multiple IP addresses to serve content from a common server hostname. Generally this is done to transmit content from a server that is close to the requesting client and/or for load balancing purposes. Figure 13 shows the distribution of the number of IP addresses used for each hostname observed for each year in our dataset. The figure shows that 80–90% of the hostnames are served by one IP address. Another 5–10% of the server hostnames are handled by two IP addresses. Approximately 5% of the server hostnames are associated with three or more IP addresses over the course of the year. While relatively rare we find many hostnames that have dozens to hundreds of IP addresses over the course of a year. In particular, we note that we observe a maximum of 144, 183, 340 and 388 IP addresses for a single hostname in 2006–2009 respectively. This shows an increase in the number of hosts brought to bear to deliver content over the course of our study for some hostnames. While this trend is not general across all hostnames we believe it may indicate larger and more dynamic CDN behavior.

We next wanted to assess the degree to which content providers are relying on the content delivery networks (CDNs) to deliver their data. In this initial exploration we focus on the Akamai CDN but intend to broaden our consideration as part of our future work. A colleague provided us with a list of partial Akamai hostnames manually gathered for another project [15]. The partial hostnames in this set were determined from downloading known Akamai-based web pages from 300-400 locations around the world. The hostnames represent 12K Akamai IP addresses [15] which represents an undercount of Akamai's footprint (e.g., Akamai is cited as having 40K servers in [13]). Therefore, our accounting of Akamai traffic is highly likely to be an underestimate. We correlate the Akamai hostnames and the DNS logs produced in conjunction with the web logs by Bro. For each web log we use the corresponding DNS log to find resolutions for the Akamai names and record the associated IP addresses. We can then easily assign web traffic as Akamai traffic or non-Akamai traffic.

Figure 14 shows the percentage of the bytes fetched in response to `GET` requests that are handled by Akamai servers. Across the time period of our study we find that

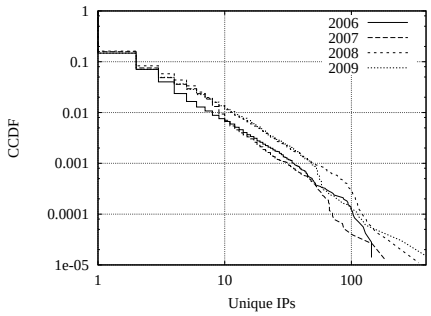

Fig. 13. Unique IPs per hostname

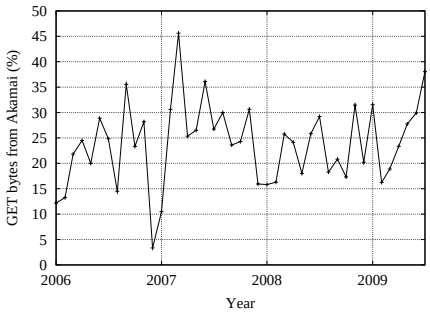

Fig. 14. Traffic using Akamai CDN servers

in general 15–30% of the bytes are delivered by Akamai.[4] However, recall that this is a lower bound due to our classification methodology. Also note that over the years of our study ICSI users accessed Akamai served content from over 9K distinct Akamai servers. The number of Akamai servers observed in a each year in our dataset is: 2.5K, 3.4K, 4.5K and 3K for 2006–2009 respectively. (Note, the 2009 count is for only half the year and so may well increase if the entire year is considered.)

6 Related Work

The large body of literature dedicated to empirical evaluations of web traffic is too vast to catalog in the space available here. A good overview, including pointers to much of the literature, appears in [9]. The topics that have been studied are diverse and numerous, with some *examples* being: characterization and modeling work [5, 3], performance analysis [6], analysis of web applications [17], analysis of web technologies [14], assessments of web caching [16, 4] and studies of the HTTP protocol itself [11, 8]. Our work is quite similar, but serves to add additional longitudinal data to the community's body of work.

7 Summary

In this paper we have employed a three and a half year longitudinal dataset of web activity to assess web operations from numerous angles. This study represents both a reappraisal of previous work and a broadening of the viewpoint in the temporal dimension. We find that some aspects of web traffic have been fairly static over time (e.g., distribution of transaction types) while others have changed (e.g., average size of `GET` and `POST` transactions). We also develop a view of the structure of the web, including an initial understanding of the behavior of browser caches and the impact of content

[4] While we see a dip to 3% at the end of 2006, this is caused by the traffic spike at the same time—as shown in figure 3—which represents traffic caused by a single client to a non-Akamai server. This client excluded, 13% of the traffic involves Akamai.

distribution networks, which we find to be more prominent as time progresses. While there are obviously more aspects of web operations to assess than could be fit in this initial paper, we believe our contribution will be useful in grounding the community's mental models and experiments in long-term empirical observation.

References

1. Z. Al-Qudah, M. Rabinovich, and M. Allman. Web Timeouts and Their Implications. In *Passive and Active Measurement Conference*, Apr. 2010.
2. M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton, and P. Hurtig. Early Retransmit for TCP and SCTP, Jan. 2009. Internet-Draft draft-ietf-tcpm-early-rexmt-01.txt (work in progress).
3. M. Arlitt and C. Williamson. Internet Web Servers: Workload Characterization and Implications. *IEEE/ACM Transactions on Networking*, Oct. 1997.
4. P. Barford, A. Bestavros, A. Bradley, and M. Crovella. Changes in Web Client Access Patterns: Characteristics and Caching Implications. *Proc. WWW*, 2, 1999.
5. P. Barford and M. Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *ACM SIGMETRICS*, pages 151–160, July 1998.
6. P. Barford and M. Crovella. Measuring Web Performance in the Wide Area. *Performance Evaluation Review: Special Issue on Network Traffic Measurement and Workload Characterization*, Aug. 1999.
7. R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, Jan. 1997. RFC 2068.
8. B. Krishnamurthy and M. Arlitt. PRO-COW: Protocol compliance on the web: A longitudinal study. In *USENIX Symp. on Internet Technologies and Sys.*, 2001.
9. B. Krishnamurthy and J. Rexford. *Web Protocols and Practice*. Addison-Wesley, 2001.
10. A. Mondal and A. Kuzmanovic. Removing Exponential Backoff from TCP. *ACM Computer Communication Review*, 38(5), Oct. 2008.
11. H. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. Lie, and C. Lilley. Network Performance Effects of HTTP/1.1, CSS1, and PNG. In *ACM SIGCOMM*, Sept. 1997.
12. V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Comuter Networks*, 31(23–24), 1999.
13. A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the Electric Bill for Internet-Scale Systems. In *ACM SIGCOMM*, Aug. 2009.
14. F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann. The New Web: Characterizing AJAX Traffic. In *Passive and Active Measurement Conf.*, 2008.
15. S. Triukose, Z. Wen, , and M. Rabinovich. Content Delivery Networks: How Big is Big Enough?, 2009. ACM SIGMETRICS poster.
16. C. Wills and M. Mikhailov. Studying the impact of more complete server information on Web caching. In *Proc. of the 5th Intl. Web Caching and Content Delivery Workshop*, 2000.
17. M. Zink, K. Suh, Y. Gu, and J. Kurose. Watch Global, Cache Local: YouTube Network Traces at a Campus Network - Measurements and Implications. In *Proc. Fifteenth Annual Multimedia Computing and Networking (ACMMCN)*, 2008.