Web Traffic Logs: An Imperfect Resource for Evaluation

Brian D. Davison (davison@cs.rutgers.edu)
Rutgers, The State University of New Jersey
USA

Abstract

Replaying or simulation using web traffic logs are common methods of evaluating proxy caches. Request traces are also used to characterize typical web workloads. Unfortunately, most common, publicly available request logs have serious deficiencies because of what is omitted from the logs. In this paper we enumerate a number of these deficiencies and examine the benefits of more complete logs. Additionally, we investigate some of the additional insights available from logs containing full content, and how they might apply to content-based prefetching proxies.

Table of Contents

Introduction

Almost all web system performance evaluation concentrates on the use of HTTP traffic logs. This paper argues, however, that most HTTP traffic logs are flawed. These deficiencies include inaccuracies, but are mostly errors of omission, and should be taken into account when considering logs for performance evaluation or workload characterization. The significance of this paper is twofold: 1) it raises issues of HTTP log utility for performance evaluation and workload characterization; and 2) it demonstrates the kind of information available when analyzing full-content logs (that is, those that contain both headers and bodies of HTTP requests and responses), especially for content-based prefetching caches.

We use both proxy and origin server logs in our analysis. To demonstrate the impact of the flaws we find, we use full-content proxy trace logs captured locally. Inaccuracies that we examine include logs that are out of date, include requests out of order, and don't reflect actual page modification times. Additionally, we point out that logs are often captured for different client populations than a desired target client population, and that difficulties exist in client workload analysis because of a lack of a consistent user to client ip mapping. Omissions that we cite include lack of HTTP headers such as `Referer:` and `User-agent:` tags, `Pragma: no-cache` instructions (or other cachability tags under HTTP 1.1), cookies,

and full URLs (including query parameters).

When full content logs are available for analysis, the request traffic can be more fully characterized. Noted characterizations include analyses based on the contents of the `Referer:` field, the potential for content-based prefetching, number of links per page requested, and truly cachable responses.

In our concluding remarks, we recommend that additional studies with larger full-content traces be performed, and that proxies should pass as much header information as provided by the client so that smart upstream proxies or servers can utilize that information to improve performance.

Background

In dynamic systems such as the Internet, it is common practice to periodically record samples of activity. Those samples are then used to characterize the activity in the system and to evaluate new mechanisms for use in the system. This is certainly true of HTTP traffic. We are not the first to note difficulties with standard proxy-cache log files. For example, Slettjord [30] notes:

> We would also like to know how many cachable documents that are non-cachable due to last-modified or expire headers that are set in such a way that an object expires before it is served from the origin server. This would give us an rough idea of how widespread cache-busting techniques are. But this information is not currently available from the log-file.

Sources of web traffic logs

On the WWW, logs of HTTP traffic are recorded continuously as a function of most origin web servers as well as intermediate proxies. The primary function of these logs is to chronicle the operation of these systems; however, as samples of HTTP activity, logs generated by these systems (and others) are also used for characterization, evaluation and usage reporting. Occasionally, researchers will capture HTTP traffic via other means, such as from augmented client browsers (e.g. [32, 20, 8]) or packet sniffing (as in [31, 14, 12, 17]).

Analysis of web server logs for the purposes of reporting traffic patterns for advertising or customer analysis is common. Additionally, proxy cache logs are sometimes analyzed for the calculation of popular web sites. Neither of these purposes are the topic of this paper and will not be mentioned further.

Instead, the rest of this paper will focus on the use of HTTP traffic logs from various sources for characterizing web traffic patterns and trends (as in [12, 29, 28, 23]); for building analytical models of same that can be used to generate artificial logs with the same patterns; and for testing, evaluating, and tuning systems such as proxy caches, switches, and web servers (as in [2, 1, 3, 11, 19]).

Trace logs to be examined

Log	Type	Software	Date	Number of Clients	Number of Requests (HTML-only/all)
	Description			Comments	
O1	Origin	WebSite	August 1997-April 1998	2968	48k/189k
	small software company			approximately 9% of HTML requests from spiders	
O2	Origin	Apache	September 1998-October 1998	27776	?/466k
	Rutgers Computer Science			very little cgi or cookies	
P1	Proxy	Squid 1.1	April 1998-July 1998	8	13k/34k
	Rutgers Computer Science users			Full headers and links	
P2	Proxy	custom	October 1998-January 1999	58	23k/73k
	Mostly Rutgers CS users			Full HTML content and headers, HTTP 1.0	

Table 1: The set of trace logs analyzed in this paper and a few of their characteristics.

Table 1 lists the various logs used in this paper. Note that log O2 used the common practice of generating a separate file with URLs and their referers, but without a timestamp. Thus, matching the original request with each referer is a non-trivial and error-prone process. WebSite logs, on the other hand, do provide referer tags.

Related work

Most performance evaluation has been based on HTTP 1.0 traffic logs. The new features of HTTP 1.1 [18] are novel enough to significantly change traffic patterns. Cáceres, Krishnamurthy and Rexford [9] propose techniques to convert HTTP 1.0 logs into semi-synthetic HTTP 1.1 logs.

Cáceres et. al. [8] demonstrate that low-level details, including the presence of cookies in HTTP headers, are significant factors in the performance of caching systems. Finally, Krishnamurthy and Rexford [25] discuss robust mechanisms for cleaning HTTP logs.

Problems with HTTP logs

Inaccuracies

HTTP logs provide snapshots of the use of web resources at a particular time. Unfortunately, since on average the lifetime of a web page is short (less than two months [33, 21, 24, 16]), any captured log loses its value quickly as more references within it are no longer valid, either by changing content or by becoming inaccessible. For example, when replaying the

request trace P1 in late August, 1998 (requests only 1-4 months old), approximately 10% of the requests resulted in a 400-class error. Likewise, Buff et. al. [7] finds that about 25% of the references in a week-old trace from a major ISP are out-dated (and thus removed them for the purposes of using the trace to build a proxy cache model).

Most logs need some kind of clean-up before analysis can be performed [25]. However, it is possible for an otherwise clean log to reflect an inaccurate request ordering as shown in Figure 1. Note that the request for the main page follows requests for three images on that main page. This because Squid records the timestamp for the completion of each request. Fortunately, this log includes processing time so the request time can be calculated and a correct ordering generated. This is important for modeling in general, and prefetching systems in particular as temporal and spatial patterns may not be preserved under a request completion ordering. Note also that the current practice of logging requests with a timestamp down to the millisecond is not always sufficient to distinguish between requests as processor and network speeds increase.

```
893252015.307 14 TCP_HIT/200 227 GET
    http://images.go2net.com/metacrawler/images/transparent.gif - NONE/- image/gif
893252015.312 23 TCP_HIT/200 4170 GET
    http://images.go2net.com/metacrawler/images/head.gif - NONE/- image/gif
893252015.318 38 TCP_HIT/200 406 GET
    http://images.go2net.com/metacrawler/images/bg2.gif - NONE/- image/gif
893252015.636 800 TCP_REFRESH_MISS/200 8872 GET
    http://www.metacrawler.com/ - DIRECT/www.metacrawler.com text/html
893252015.728 355 TCP_HIT/200 5691 GET
    http://images.go2net.com/metacrawler/images/market2.gif - NONE/- image/gif
893252016.138 465 TCP_HIT/200 219 GET
    http://images.go2net.com/metacrawler/templates/tips/../../images/pixel.gif -
    NONE/- image/gif
893252016.430 757 TCP_REFRESH_HIT/200 2106 GET
    http://images.go2net.com/metacrawler/templates/tips/../../images/ultimate.jpg -
    DIRECT/images.go2net.com image/jpeg
```

Figure 1: This excerpt from proxy log P1 generated by Squid 1.1 records the timestamp, elapsed time, client, code/status, bytes, method, URL, client-username, peerstatus/peerhost and object type for each request. It is also is an example of how requests can be logged in an order inappropriate for replaying in later experiments.

Finally, proxy cache trace logs are inaccurate when the proxies return stale objects since they may not have the same characteristics as current objects. In fact, since proxy logs don't reflect actual page content change times, prefetching simulations that use trace logs cannot know when they have prefetched and are storing what will later be used as a stale object. Even when run conservatively, caches sometimes return stale data, much to the consternation of users trying to use their browser to automatically open via FTP the latest draft of a colleague's paper. To determine exactly how much stale data a proxy cache is generating, one might request the object from the origin and compare it to what the cache returns (as suggested in [15]).

When using logs for evaluation, it is important to consider the client population and workload generated by it [26]. The patterns of usage and performance seen by a first-level workgroup or enterprise proxy cache may differ considerably from that of a high-level cache that serves only as a parent to other caches, and not to clients directly. Likewise, the logs of a university httpd server with mostly static pages will generate different browsing patterns among clients than those of a highly-dynamic (and thus less cachable) pages of a commercial

site. Additionally, cultural issues may affect the workload generated (e.g. users in Brazil access different kinds of sites than users in the U.S. [4].).

Finally, when analyzing logs to build user models, one must consider which logs to use carefully, as in many logs a single client may represent the combined requests of multiple users because of the use of proxies or multi-user machines. A single user may also be represented by multiple unique client ids as a result of dynamic IP address allocation (common in dialup connections, but also sometimes used for infrequently used LAN connections).

Omissions

Most proxy and origin servers record only a small portion of each HTTP request and/or response, and even when they support the extended log format [22], they are usually not configured to record more than that shown in figure 1. Logs generated by httpd servers sometimes contain `User-agent:` and `Referer:` tags, but these are often not associated with particular requests (as in the case of log O2). The `User-agent:` header can provide information as to the client capabilities, and might help explain the client distribution of HTTP/1.1 support, or the effect of well-known browser bugs. The `User-agent:` is also useful when cleaning logs of crawler activity. Examples of `Referer:` tag use will be shown in the next section.

Caches that prefetch on the basis of the contents of the pages being served (termed *content-based prefetching*), such as CacheFlow [10] and Wcol [13], need at least to have access to the links within web pages -- something that is not available from server logs. Even if page contents were logged (as in [14, 27] and log P2), caches that perform prefetching may prefetch objects that are not on the user request logs and thus have unknown characteristics such as size and web server response times.

When we replayed a relatively small trace log (as part of separate work), we found that the average size hit for the proxy cache was larger than the average miss. When this happens, one is understandably suspect. In this case the average had moved because a few large requests were repeated approximately a dozen times. Upon further examination, this effect was the result of one user in an authoring mode, reloading a page that had some unusually large images many times while the page was under development. Without those images, the average hit and miss sizes for that trace were more reasonable. This anomaly demonstrates the utility of additional request headers; in this case the knowledge and re-use of `Pragma: no-cache` would have eliminated the problem. Since logs generally do not include this kind of additional request header information, those requests that were originally forced to bypass the cache (with a `Pragma: no-cache` header) no longer have to do so in a replayed situation, and thus artificially inflate the resulting hit rates. In fact, this is typical of the larger problem in that many logs don't show whether a response is cachable -- cookies are not present, URLs that are queries have their parameters stripped, etc.

This means that proxy logs should not be used for comparing the performance of the system that generated the log to others that use the log for simulation. Such comparisons (e.g. as in [19]) are unreasonable, since in most cases we don't know if the origin servers support HTTP/1.1, don't know if the client did a refresh, don't know about cookies, etc., all of which

affect cachability in the real world but are not present in captured trace-based simulations.

Many logs do not show the service or transmission times. This makes it difficult to estimate the ‘‘thinking time’’ of a user accurately, or the user’s bandwidth. Neither log O1 or O2 record this, although Apache [5] has the option, but is not used by default. Finally, note that most logs are missing some requests (i.e. a httpd log only shows requests to that httpd server; the typical proxy cache log shows only port 80 requests and ignores FTP, SSL, etc. which are valid, but less common web transport mechanisms).

Insights from additional content

When additional content is available, analysis may provide new insights. For example, one might find that client prefetching based on the contents of the current page is valuable, since approximately 80% of requests are made from the current page. While this is consistent with a user study [31] it can also be confirmed by calculating the percentage of HTTP responses of type text/html that had requests with a `Referer:` tag, as shown below:

- O1: 83.4% of HTML pages had `Referer:` tags (after eliminating the 9% of HTML spider requests)
- O2: 68.8% of all requests had `Referer:` tags (not calculated on just text/html, and without eliminating robot requests)
- P1: 78.8% of HTML pages (no spiders)
- P2: 78.6% of HTML pages (no spiders)

This is evident in logs that show the `Referer:` request tag, and generally the percentage of referer tags over all requests was approximately ten percentage points higher. The `Referer:` tag can do more -- it can provide hints of the client’s true browsing pattern (since many requests, such as [back], are normally handled internally by the browser cache).

	P1	P2
Percent with cookies:	17.4	44.1
Percent with Expires:	1.5	7.8
Percent with Last-Modified:	71.6	65.8
Percent with ETag:	35.4	30.7
Percent with HTTP/1.1:	49.9	0

Table 2: Additional statistics available from augmented proxy logs. Note that since trace P2 was collected by an HTTP/1.0 proxy, there could be no HTTP/1.1 responses. For comparison, note that Cáceres et. al. report that over 30% of ISP traces contained cookies.

In table 2 some additional statistics can be found that may be relevant for cache design or workload modeling. The cachability of a workload is also often of interest. A common mechanism used by caches is to employ a stoplist on the URLs. A conservative cache operator might set the stoplist to contain ‘‘?’’, ‘‘cgi’’, ‘‘.asp’’ which would cause the cache to never cache any URL with those substrings. In log P1, 8.3% of the URLs contained a

stoplisted substring or was the result of a POST. In log P2, this was 13.7%. Since both of these logs are augmented with all request and reply headers, we can get a better estimate (although still incomplete) of uncachable objects by also discarding those with ‘‘no-cache’’, ‘‘set-cookie’’, ‘‘max-age=0’’, ‘‘Expires: 0’’, and ‘‘Expires: Thu, 01 Jan 1970’’. For this augmented stoplist, our uncachable percentage rose to 19.3% and 31.6%, respectively. Finally, if we make the HTTP/1.0 assumption that cookies signify uncachable data as well, it rises to 26.1% and 54.4% respectively. Note however, that these are representative but incomplete statistics and analyses that are possible with additional content.

Real-time prefetching systems rely on the existence of ‘‘thinking time’’ -- the time between page requests -- to provide time in which to prefetch the objects likely to be requested next. Past research [12, 14] suggests a heavy tailed distribution of thinking times with a mean of approximately 30 seconds. Logs P1 and P2 have averages of 79.4 and 42.6 seconds, respectively for the thinking time between HTML page requests (using the traditional 30 minute threshold for session breaks). When calculated as the time between requests for HTML pages, the average rises slightly, to 82.8 and 46.8 seconds, respectively.

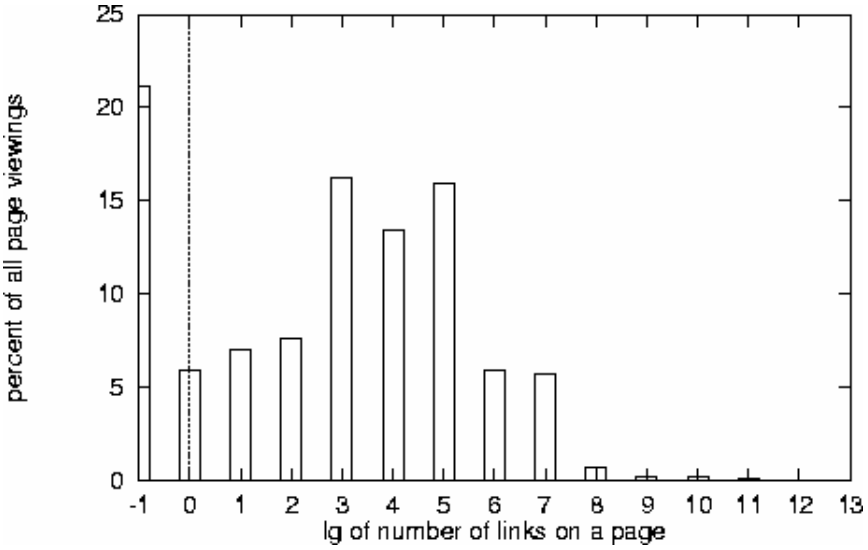

Figure 2: The distribution of the number of links per non-error HTML page in trace P1.

Figure 2 shows the distribution of the number of links per page from trace P1, which has an average of 25.4 unique, non-self-referential links per HTML page (and does not include embedded tags, e.g. images and sounds). Note that this histogram reflects the distribution of pages requested by users, versus the more or less static distribution of pages on the web (such as described by Bray [6]).

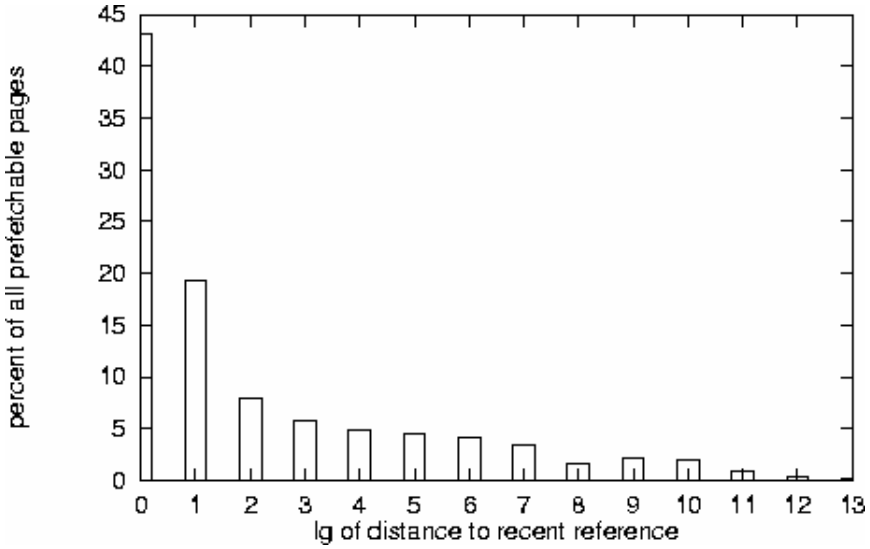

Figure 3: The distribution of the distance, in terms of the number of HTML requests back in time from which the current page could have been prefetched.

A more complex analysis is shown in figure 3. The set of prefetchable pages are those that are considered unlikely to have adverse side-effects (e.g. no cgi). It shows that more than 62% of prefetchable pages can be reached by examining the links of the current page and it's immediate predecessor in the request stream.

Summary

This paper has described a number of common deficiencies in proxy and origin server logs. While some of these omissions may be a conscious decision towards privacy, others are likely to be simply a function of system defaults. In any case, the lack of information can seriously affect the resulting characterization of web traffic using those logs.

For most analyses, server logs that are augmented with all request and reply headers can provide stronger and more detailed characterizations than those typically made available. For prefetching systems, even full-content logs may be insufficient. In general, more information allows for more accurate theories and models, and are necessary for postulating and evaluating more complex caching mechanisms, such as content-based prefetching.

Therefore, we recommend performing additional studies with larger full-content traces, as the results presented here are only representative of what can be determined with correct and complete logs. We also recommend that proxies pass as much header information as provided by the client so that smart systems upstream can utilize it.

Finally, we have argued that proxy logs cannot be used for comparing the performance of the system that generated the log to others that use the log for simulation. Such comparisons are unreasonable because the logs lack information that affect cachability in the real world but are not present in captured trace-based simulations.

References

1 Jussara Almeida and Pei Cao. Measuring Proxy Performance with the Wisconsin Proxy Benchmark. In *Third International WWW Caching Workshop*, Manchester, England, June 1998. Also available as Technical Report 1373, Computer Sciences Dept., Univ. of Wisconsin-Madison, April, 1998.

2 Jussara Almeida and Pei Cao. Wisconsin proxy benchmark 1.0. Available from http://www.cs.wisc.edu/~cao/wpb1.0.html, 1998.

3 Jussara Almeida and Pei Cao. Measuring Proxy Performance with the Wisconsin Proxy Benchmark. *The Journal of Computer Networks and ISDN Systems*, To appear, 1999.

4 Virgílio Augusto F. Almeida, Márcio G. Cesário, Rodrigo C. Fonseca, Jr. Wagner Meira, and Cristina D. Murta. Analyzing the behavior of a proxy server in light of regional and cultural issues. In *Third International WWW Caching Workshop*, Manchester, England, June 1998.

5 Apache Group. Apache HTTP server documentation. Available at http://www.apache.org/, 1998.

6 Tim Bray. Measuring the web. In *Proceedings of the Fifth International World Wide Web Conference*, Paris, France, May 1996.

7 Robert Buff, Arthur Goldberg, and Ilya Pevzner. Rapid, Trace-Driven Simulation of the Performance of Web Caching Proxies. Submitted to WISP'98, March 1998.

8 Ramón Cáceres, Fred Douglis, Anja Feldmann, Gideon Glass, and Michael Rabinovich. Web proxy caching: the devil is in the details. In *Workshop on Internet Server Performance (WISP'98)*, Madison, WI, June 1998.

9 Ramón Cáceres, Balachander Krishnamurthy, and Jennifer Rexford. HTTP 1.0 Logs Considered Harmful. In *World Wide Web Consortium Workshop on Web Characterization*, Cambridge, MA, November 1998. Position paper.

10 CacheFlow Inc. CacheFlow products web page. http://www.cacheflow.com/products/, 1998.

11 Pei Cao. Characterization of Web Proxy Traffic and Wisconsin Proxy Benchmark 2.0, In *World Wide Web Consortium Workshop on Web Characterization*, Cambridge, MA, November 1998. Position paper.

12 Lara D. Catledge and James E. Pitkow. Characterizing Browsing Strategies in the World Wide Web. *Computer Networks and ISDN Systems*, 26(6):1065-1073, 1995.

13 Ken-ichi Chinen and Suguru Yamaguchi. An interactive prefetching proxy server for improvement of WWW latency. In *Proceedings of the Seventh Annual Conference of the Internet Society (INET'97)*, Kuala Lumpur, June 1997.

14 Carlos R. Cunha, Azer Bestavros, and Mark E. Crovella. Characteristics of WWW Client-based Traces. Technical Report TR-95-010, Computer Science Department, Boston University, July 1995.

15 Brian D. Davison. Simultaneous Proxy Evaluation. In *Proceedings of the Fourth International Web Caching Workshop (WCW99)*, March 1999.

16 Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey C. Mogul. Rate of Change and other Metrics: a Live Study of the World Wide Web. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS '97)*, December 1997. An extended version is available as AT&T Labs - Research Technical Report 97.24.2.

17 Anja Feldmann. Continuous online extraction of HTTP traces from packet traces. In *World Wide Web Consortium Workshop on Web Characterization*, Cambridge, MA, November 1998. Position paper.

18 R. Fielding, J. Gettys, Jeffrey C. Mogul, H. Frystyk, and Tim Berners-Lee. Hypertext Transfer Protocol -- HTTP/1.1. RFC 2068, http://nic.ddn.mil/ftp/rfc/rfc2068.txt, January 1997.

19 Arthur Goldberg, Ilya Pevzner, and Robert Buff. Characteristics of Internet and Intranet Web Proxy Traces. In *Computer Measurement Group Conference CMG98*, Anaheim, CA, December 1998.

20 Steven D. Gribble and Eric A. Brewer. System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS '97)*, December 1997.

21 James Gwertzman and Margo Seltzer. World-Wide Web Cache Consistency. In *Proceedings of the USENIX Technical Conference*, San Diego, CA, January 1996.

22 Phillip M. Hallam-Baker and Brian Behlendorf. Extended log file format. W3C Working Draft WD-logfile-960323, available from http://www.w3.org/TR/, 1996.

23 Bernardo A. Huberman, Peter L. T. Pirolli, James E. Pitkow, and Rajam M. Lukose. Strong regularities in world wide web surfing. *Science*, 280:95-97, April 1998.

24 Brewster Kahle. Preserving the Internet. *Scientific American*, 276(3):82-83, March 1997.

25 Balachander Krishnamurthy and Jennifer Rexford. Software Issues in Characterizing Web Server Logs. In *World Wide Web Consortium Workshop on Web Characterization*, Cambridge, MA, November 1998. Position paper.

26 Ian Marshall and Chris Roadknight. Linking cache performance to user behaviour. In *Third International WWW Caching Workshop*, Manchester, England, June 1998.

27 Jeffrey Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy. Potential Benefits of Delta-encoding and Data Compression for HTTP. In *Proceedings of ACM SIGCOMM*, pages 181-194, September 1997. An extended and corrected version appears as Research Report 97/4a, Digital Equipment Corporation Western Research Laboratory, December, 1997.

28 James E. Pitkow. In search of reliable usage data on the WWW. In *Proceedings of the Sixth International World Wide Web Conference*, Santa Clara, CA, April 1997.

29 James E. Pitkow and C. M. Kehoe. Emerging trends in the WWW user population. *Communications of the ACM*, 36(6):106-108, 1996.

30 Lars Slettjord. Web-cache statistics meeting at JENC8. From the TF-CACHE mailing list. Available at http://www.terena.nl/tech/projects/choc/statistics-recommendations.txt, May 1997.

31 Linda Tauscher and Saul Greenberg. How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human Computer Studies*, 47(1):97-138, 1997.

32 Roland O. Wooster, Stephen Williams, and Patrick Brooks. HTTPDUMP: Network HTTP packet snooper. Working paper available at http://www.cs.vt.edu/~chitra/work.html, April 1996.

33 Kurt Worrell. Invalidation in large scale network object caches. Master's thesis, University of Colorado, Boulder, CO, 1994.

Published in the *Proceedings of the Ninth Annual Conference of the Internet Society (INET'99)*, San Jose, June 22-25, 1999.