

BU COMPUTER SCIENCE 1998 PROXY TRACE

*Release notes**

Adam Bradley
artdodge@cs.bu.edu

Computer Science Department
Boston University
Boston, MA 02215

August 1999

Abstract

In a recent paper [1], we performed a variety of analyses upon user traces collected in the Boston University Computer Science department in 1995 and 1998. A sanitized version of the 1995 trace has been publicly available for some time; the 1998 trace has now been sanitized, and is available from:

- <http://www.cs.bu.edu/techreports/1999-011-usertrace-98.gz>
- <ftp://ftp.cs.bu.edu/techreports/1999-011-usertrace-98.gz>

This memo discusses the format of the released version of the log, and includes additional discussion of how the data was collected, how the log was sanitized, what this log is and is not useful for, and areas of future research interest.

*This work was partially supported by NSF research grants CCR-9706685 and CCR-9501822, and by Microsoft Research and Hewlett-Packard Laboratories.

1 Data Collection

The trace was collected in the Boston University Computer Science Department's undergraduate workstation lab. The lab consists of a cluster of 37 Sun SPARCstation 2's running Solaris. Traces were collected from April 4, 1998 through May 22, 1998, and included 306 unique users. Students were given the option to "opt out" of the study (without so much as their decision to do so being logged), so the trace is probably not exhaustive.

The trace was collected using lightweight non-caching HTTP/1.1 proxy servers running on each workstation. The proxies logged all requests on a shared NFS filesystem. We discuss the difficulties of data collection using a proxy server in section 3.3 of [1]. It is recommended that researchers familiarize themselves with all of section 3 of that paper before trying to derive any conclusions from these traces.

2 Log Format

The trace is stored as a single gzipped text file. Each transaction in the trace is stored as a single line of text (CR-terminated). Fields ("columns" of data) are each separated by a single space. No field contains any white space (spaces or tabs). Timestamps are presented as decimal fixed-precision floating point numbers, expressing seconds and microseconds since the UNIX epoch¹.

- Request : Method (GET, HEAD, POST...)
- Request : Host: field (DNS name of server, **hashed**)
- Request : URL (in absolute-URI format, **hashed**)
- Request : Referer (in absolute-URI format, **hashed**)
- Connection : Client IP address (**hashed**)
- Connection : Server IP address (**hashed**)
- Connection : Server Port number (usually 80)
- Request : User name (**hashed**)
- Response : Status code (200, 304, etc; see below)
- Response : Response content-length (in bytes; see below)
- Timestamp : Proxy started sending request to server
- Timestamp : Proxy started receiving response from server
- Timestamp : Proxy finished receiving response from server

This is almost all of the information that was present in the basic trace from which [1] was derived. Three types of information have been completely removed: *First*, we removed several additional timestamps that were used to examine the performance of the data-collecting proxy and do some simple latency-related calculations that were not included in the paper. Proper use of those timestamps would require knowledge regarding the internal architecture of the proxy. *Second*,

¹a.k.a. "The Dawn of Time", 0:00:00 Jan 1 1970 GMT

additional IP address and port information that could be used to identify persistent connections and provide additional key information had we decided to use a central rather than a distributed proxy. This information is not needed to identify individual client machines from the provided data set. *Third*, fields that were invariant across the whole trace. There were two such fields:

- Client HTTP Version : HTTP/1.0
- User-Agent : Mozilla/4.04 [en] (X11; I; SunOS 5.4 sun4c)

3 Discussion

3.1 Anonymizing

Fields marked as “**hashed**” have been anonymized using a one-way hash function. Specifically, we have stored:

$$value_{hashed} = checksum_{MD5}(concatenate(value_{original}, nonce))$$

Where “*nonce*” is a binary string of non-trivial length. A single *nonce* value was selected randomly (such that each is “unguessable”) for each “column” of the dataset; this preserves the ability to compare members of a column for equality without leaking information regarding equality across columns. (There is one exception: the URI and Referer fields are hashed using the same nonce, allowing you to identify referring documents.) The *nonces* were generated using Linux’s `/dev/random` device (which uses hardware interrupts and other sources of noise to populate an entropy pool), were never stored on disk (or even swap; the anonymizer was locked into main memory using `mlockall()`), were never displayed or recorded, and were immediately discarded upon completion of the anonymization.

3.1.1 Limitations

Since (for obvious reasons) the URLs are only available in this hashed form, you will be unable to:

- Look for URL sub-strings (f.e., “cgi-bin” or “.jpg”).
- Separate URLs from query-strings (“http://resource?query”)
- Infer anything regarding length or depth of the URI or its components, hostname and `abs_path`
- Infer two URLs have equivalent paths but differing hostnames (f.e. “http://www.cs.bu.edu/X” and “http://www-cs.bu.edu/X”)

Anonymization techniques which preserve at least some of these and other interesting properties are being examined; we hope traces released in the future and re-releases of this trace will be processed with such techniques.

3.2 Methodology, Caveats, Exceptions, and other tidbits

Presented in no particular order.

Proxy Logs vs Client Logs Since this log ONLY captures network accesses made by the client, views of documents which were “served” directly out of the browser’s cache without revalidation do not appear in this trace.

Timestamps Workstation clocks are not tightly synchronized with each other, or with any external time source. They are generally within a few seconds of each other, but this is not uniformly the case. In other words, timestamps only provide a total ordering of requests on a single workstation, not a global total ordering.

Periodicity A fourier analysis of the amount of traffic (requests and bytes) will show strong daily and weekly components. This can be explained by the lab's hours of operation. Note that the lab being "closed" does not uniformly cause activity to go to zero; terminal assistants and others in our department have off-hours access rights. Further informal examination shows variability across multiple time-scales, which is a property of statistically self-similar traffic [2], although no attempt was made to validate this finding or estimate the *degree* of self-similarity.

Conditional Requests Browser cache validations appear in the trace; successful validations are indicated by a response-code of 304 (not modified), but unsuccessful validations are indistinguishable from compulsory cache misses.

Cache-Control Pragma and Cache-Control headers can skew popularity profiles of resources by forcing them to be re-validated (and thus seen by the proxy) at a rate much higher than cachable documents which the user *views* with the same frequency. Cache-control headers were not stored in the logs.

Request Parallelism The workstation proxies were running with 10 service threads; this put an upper bound on the number of requests that each proxy could handle in parallel. Take note that the proxy multiplexed *requests* and not *connections* to the threads, so a higher number of parallel connections was possible.

Proxy Auto-Configuration Using the PAC option afforded us a tremendous amount of flexibility in deploying the proxy; for example, should we develop Mozilla 5.0 or IE5.0 logging modules, we could selectively direct those clients to ignore the proxy, while telling older non-logging clients to use it. (Environments with multiple installed versions of Netscape are not uncommon in academic computer labs.)

No Referring Document Specified Some requests are sent with no value in the "Referer" (sic) field. These were noted in the logs with a Referer value of "(no-referer)". The **first** hashed "Referer" value in the published trace corresponds with this value.

Failed ident lookups The RFC1413 ident service does not produce correct results under certain corner cases (for example, premature closes of the client connection). Such cases are stored with a username of "(nousername)". The **eighth** hashed username value in the published trace corresponds with this value.

User Sessions We have no direct way of distinguishing the end of a user session from a long think-time. Since the workstations were all set up to only allow a single logged-in user at a time, one simple and effective heuristic is to examine the streams for each individual workstation, and mark a "new session" whenever the username field changes. (If you take this approach, be mindful of the "(nousername)" issue mentioned above.)

Aborted Connections Some cases of early client aborts were detected by the proxy, and are indicated by the response status code being stored as a negative number. The abort detection code used was fairly primitive, and should be regarded as an underestimate. It is not believed that any “false aborts” are recorded in the log, although “*fault* aborts” may be present as a result of poorly-formatted responses from inbound servers. (Sadly, there is an abundance of broken HTTP applications [3], and there is no canonical list of work-arounds employed by browsers, so the proxy may have given up on non-compliant responses that Netscape could have handled.)

Content Lengths Some responses did not specify the response content-length. These response are recorded as having a content-length of -1. Some of these are “proper” cases, like 304 responses (which by definition carry no entity); others are cases where the server is relying on either a “chunked” send to the proxy, or is closing the connection to indicate the end of the stream of content. Current versions of the proxy collect content-length information in these cases as well, but the code was not in place to do so when this trace was collected.

Proxy Software The proxy software used to collect the data is called “REFLEX”, and is being developed primarily by the author of this paper. REFLEX is designed as a general HTTP server architecture, and the non-caching proxy functionality is implemented as a service module. It is currently being extended to act as a HTTP/1.1 (RFC2616) compliant caching proxy and to collect more extensive data for additional study². It will eventually be released under the GNU GPL.

4 Future Directions

We intend to conduct another round of data collection. We have devised techniques for collecting more extensive information about user browsing patterns, and intend to collect a richer set of metadata as well (including information about request parameters, document properties, content negotiation, explicit cachability controls, cookies, and other items of interest). Some research topics of particular interest include:

- **Real-world Cachability:** Matching URIs provides a very optimistic upper bound on cache hit rates, byte hit rates, and cache miss costs. Real HTTP/1.1 caches have to worry about document lifetimes, content negotiation, scoping (shared vs. non-shared caches), explicit cachability controls, validation costs, and the bigger problem that servers may not provide the cache with enough information to behave correctly (for example, some caches assume that responses to requests bearing cookies are uncacheable because some sites fail to specify “Vary: Cookie” or set any explicit cache controls in responses that customize content based upon cookie values).
- **Explaining Zipf-like Popularity Distribution:** While many traces show Zipf-like popularity distributions of names across time, this may be influenced not only by general popularity skew but also by document introduction/removal, rate of change, and other properties. We intend to study revision popularity and its correlation with name popularity, as well as long-term variations in the content of short-window popularity profiles.
- **Search Utilities:** Proxies have the potential to provide valuable information if they are engineered to collaborate with non-interactive or offline-processing web applications like robots/crawlers, indexes/archives, and search engines.

²We hope to begin a new round of data collection experiments late this fall

References

- [1] Paul Barford, Azer Bestavros, Adam Bradley, and Mark Crovella. Changes in web client access patterns : Characteristics and caching implications. *World Wide Web*, 2(1,2):15–28, 1999.
- [2] Mark Crovella and Azer Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, December 1997.
- [3] Balachander Krishnamurthy and Marin Arlitt. Pro-cow: Protocol compliance on the web, August 3 1999.