In Search of Reliable Usage Data on the WWW

James Pitkow
Xerox Palo Alto Research Center
Palo Alto California 94304 USA
pitkow@parc.xerox.com

Abstract

The WWW is currently the hottest testbed for future interactive digital systems. While much is understood technically about how the WWW functions, substantially less is known about how this technology is used collectively and on an individual basis. This disparity of knowledge exists largely as a direct consequence of the decentralized nature of Web. Since each user of the Web is not uniquely identifiable across the system and the system employs various levels of caching, measurement of actual usage is problematic. This paper establishes terminology to frame the problem of reliably determining usage of WWW resources while reviewing current practice and their shortcomings. A review of the various metrics and analyses that can be performed to determine usage is then presented. This is followed by a discussion of the strengths and weaknesses of the hit-metering proposal [Mogul and Leach 1997] currently in consideration by the HTTP working group. Lastly, new proposals, based upon server-side sampling are introduced and assessed against the other proposal. It is argued that server-side sampling provides more reliable and useful usage data while requiring no change to the current HTTP protocol and enhancing user privacy.

1 Terminology

Despite several efforts and widespread agreement on the need to establish a lingua-franca for usage and demographic collection, consensus does not exist [W3C 1996]. One of the more recent and comprehensive efforts [Novak and Hoffman 1996] provides a baseline of terminology specifically designed for the advertising and measurement communities. With the intent of generating consensus, this paper will build upon their framework for classifying visitors and terminology, clarifying and introducing new terminology as needed. Readers familiar with the notions of cookies, hits, and the problems local caches and proxy-caches have on reliably determining usage may wish to skip to the next section statistical analysis.

1.2 Visitors and Cookies

The central issues regarding the classification of visitors to a Web site are the ability to uniquely identify visitors and the ability to do so reliably, especially across multiple visits to a site. Visitors to WWW sites can be separated into the following categories: unidentified, session, tracked, and identified [Novak and Hoffman 1996]. For each class of visitors, a definition is provided followed by a discussion of the methods used to achieve each level of knowledge about visitors to a site.

A *unidentified visitor* is a person who visits a Web site where no information is available about the visitor. This type of visitor does not truly exist on the Web since the Internet Protocol requires at least a machine name to return the requested information. This form of return address reveals information about the users in a similar manner to that of a phone number, where a one-to-one or a many-to-one correspondence may exist between the address and the number of users at the address. Unidentified visitors may indeed exist in other interactive digital systems, where a user's anonymity is explicitly preserved. It is arguable that anonymous proxies, programs that acts as a intermediary between clients and servers, enable users to experience the Web in an anonymous manner. While this form of server may exist to a limited number of users, it does not scale well to handle the entire user population since it effectively doubles the amount of traffic required to request a page (by first sending a request to the anonymous server which then sends a second request to the actual server) and requires a fair amount of centralization of resources, another potential bottleneck.

A *session visitor* is a visitor to a Web site where an identifier is created either explicitly via cookie[1] generation or inferred through heuristics as discussed below. This is the default type of visitor on the WWW today. Several revealing pieces of information are typically available which enable the heuristic identification of users even if cookies are not used. With each request, information about the machine name from which the visitor made the request, the type of software the visitor is using to experience the WWW, the operating system on which the software operates, and the page viewed prior to the current request is typically known. The latter piece of information is called the referrer field. While this ancillary information may enable a user to be identified within a session, it is not guaranteed to be accurate nor will it be able to reliably identify the same user in future sessions. Some heuristics for identifying users without cookies include:

- The use of Internet protocols to help determine if the user is the sole user of the machine making the request, e.g., identd, finger, etc. If a one-to-one correspondence exists between a visitor and a machine, the machine essentially becomes a unique identifier, and the visitor becomes a 'tracked visitor' as described below. These techniques fail if a visitor exists behind a proxy (as is the case if Figure 1), shares machines with other users, or the machine being used to experience the Web does not support or allow these protocols.

- To uniquely identify users suspected of existing behind proxies (see Figure 1), session limits, the site's topology (the global hyperlink structure across pages), and browser characteristics can be used. One such algorithm implemented in [Pirolli, Pitkow, and Rao 1996] checks that each incoming request is reachable from the set of already visited pages. This is done by consulting the site's topology. If all subsequent requests are made to pages that the visitor could have reached by selecting a hyperlink embedded in any of the already requested pages, the user is assumed to be the sole visitor behind the site. If requests from the same machine name occur for pages that are not reachable from the set of hyperlinks embedded in the pages already visited, multiple visitors are suspected. Multiple visitors are also suspected when pages are requested that have already been visited. The algorithm treats these cases as separate visitors and adds subsequent page to each visitor's path based upon the topology of the site. A least recently used policy is used to add pages to visitors if ambiguity exists between which user could have made the request. Visitors who do not request pages within a certain time limit are assumed to have left the site. Appropriate time-out periods are typically determined by inspection of the distribution of time between all page requests to a site. While the above algorithm performs reasonably well, it is heuristic in nature and has not been shown to reliably identify users with any measure of accuracy, especially across sessions.

A *tracked visitor* is a visitor who is uniquely and reliably identifiable across multiple visits to a site. In the earlier days of the Web, the tracking of visitors was often accomplished by inserting identifiers into the URLs issued by the server and channeling all subsequent requests through a CGI script. Not only was this method expensive computationally to the sever, but it defeated intermediary caching and did not correctly handle the exchanging of URLs between people, i.e., the person using a URL of this sort mailed to them by a friend could be incorrectly tracked as the friend. These days, this form of identification is typically accomplished by setting the expiration of an issued cookie into the far future. Thus, each time a visitor returns to the site, the same identifier will be used.

The increased use of this technique to track users has not been without notice by the user community, where rather rudimentary but effective practices have emerged that periodically erase cookies stored on the visitor's filesystem or do not permit the storing of cookies between sessions by disabling write

---

[1] Cookies are server generated identifiers that enable the management of state between visitors and servers. They were initially designed to implement shopping baskets for the Web but have found a killer application in the tracking of user behavior. When a visitor requests a page, a server can return an identifier, a.k.a. cookie, with conditions on when and how the identifier is to be used.

permissions to the appropriate files. These practices have the effect of causing the site issuing the cookie to issue another identifier, resulting in potential over-inflation of the number of unique visitors to the site. Commercial software that performs cookie obfuscation is also emerging, e.g., PGPCookie.Cutter [PGP; 1996].

An *identified Visitor* is a tracked visitor where additional information is available. This is the most common type of visitor when persistent identifiers are employed to monitor usage. While it may appear that tracked visitors would be more common, additional information as mentioned in the above section of session visitors accompanies each request. Rough estimates of the demographics of a user can be made since the entity that owns the domain from which the request was issued can be determined somewhat reliably from InterNIC's publicly accessible domain registration database. Once the entity have been established, this information can be matched against other databases that enable the construction of user profiles. For example, if a request comes from a visitor behind a corporate proxy named xyz.com, via InterNIC's database, one can discover that the actual company that owns that domain is FooBar Corp., and then lookup information on FooBar Corp. in other sources. Many of the commercially available log file analysis programs come with databases that match domain names to core demographics, e.g., Interse [Interse 1996].

Of course, the most obvious method of collecting additional demographics of users is by asking the actual users of the site. This is routinely accomplished via online registration forms. However, GVU's most recent WWW User Survey data show that 33% of the over 14,500 respondents have falsified the information for online registration forms at least once [Pitkow and Kehoe 1996]. Over 10% reported that they provided incorrect information over 25% of the time. Although online registration is currently common practice and will remain so for the foreseeable future,

> *the information collected from online registration systems needs to be thoroughly examined on a per-site basis before reliable statements about the users of a site can be made from this information.*

Other methods for collecting demographics of users at a site include Universal Registration Systems, e.g. I/PRO's I/COUNT [I/PRO 1996]. These systems require a user to register only once in exchange for an identifier. This identifier can then be used across the set of participating sites. While the goal is to 1) make registration easier for users and 2) provide sites with valuable demographic information, scalability issues have hindered the development of most of these types of systems, and as such, few Universal Registration Systems exist in practice today.

The above classification of visitors and the accompanying definitions should provide the necessary framework to move forward and provide definitions for the terms that are used to measure the items visitors request.

1.2 Accesses and Caches

When a visitor requests and item from a Web site and the server returns the item to the user, an *access* is said to have occurred. In the common speak of the Web, this event is fondly referred to as a '*hit*.' It is widely recognized that the reporting of hits as a measure of usage is meaningless for the following reasons. First, when a user accesses a page, the client sends a request to the server in the form of a URL. If the item being requested is an HTML page, it may include other URLs embedded into its content, which are often images, audio files, video files, applets, as well as other text. When the client gets the requested HTML page back from the server, it reads the contents of the page and makes subsequent requests for the embedded URLs. Thus, a page with three embedded images results in a total of four hits as measured by the server (one hit for the HTML page, and one hit each of the three images). Since the composition of pages differs within a site as well as across sites, the comparison of hits reported by different sites is useless. For example, in Figure 1, when Visitor 1 requests Page A, Server Z will record four hits but Server Z will only record two hits when Page C is requested. What sites really want to compare is the

number of pages requested by visitors, which is often called *page views*. Page views can be crudely measured by excluding non-HTML requests from the tabulation process.

The second major problem with hits occurs because of the various levels of caching that occur on the Web. Caching is a mechanism that attempts to decrease the time it takes to retrieve a resource by storing a copy of the resource at a closer location. Prior to the fall of 1994, most browsers did not include local caches. As a result, each time a page was accessed by a user, a separate page request would need to be issued to the original site and the contents returned each time to the user. To remedy this inefficient situation, browser software began to integrate caches that utilized the user's local disk space as well as in-memory caches that stored pages in memory rather than on disk.

For example, in Figure 1, when Visitor 1 requests Page A, Page A is stored on Visitor 1's computer in the local cache. When Visitor 1 goes off to visit Page B and then returns to Page A via the 'Back' button, a request to Site Z does not need to be made. Visitor 3 on the other hand would need to issue two requests for Page A to perform the same task. Upon tabulation of the access log for Site Z, one would find two different totals for the same navigation: Visitor 1 would record only one page view for Page A whereas Visitor 3 would record two page views. Which one is correct? Depends on the definition of a page view. Since local caching has become the standard in browsers and users can select different cache management policies, *page views* typically are defined to only reflect that the page was viewed *at least once*. Given this frame work, *unique page views* represent the first time a user requests a page in a session. *Reuse page views* refers to the total times the page is viewed minus the first page view., with *total page views* being the sum of the two. Without mucking with the configuration of the server to defeat local caching, it should be clear that no guarantees can be made about the measurement of total page views.

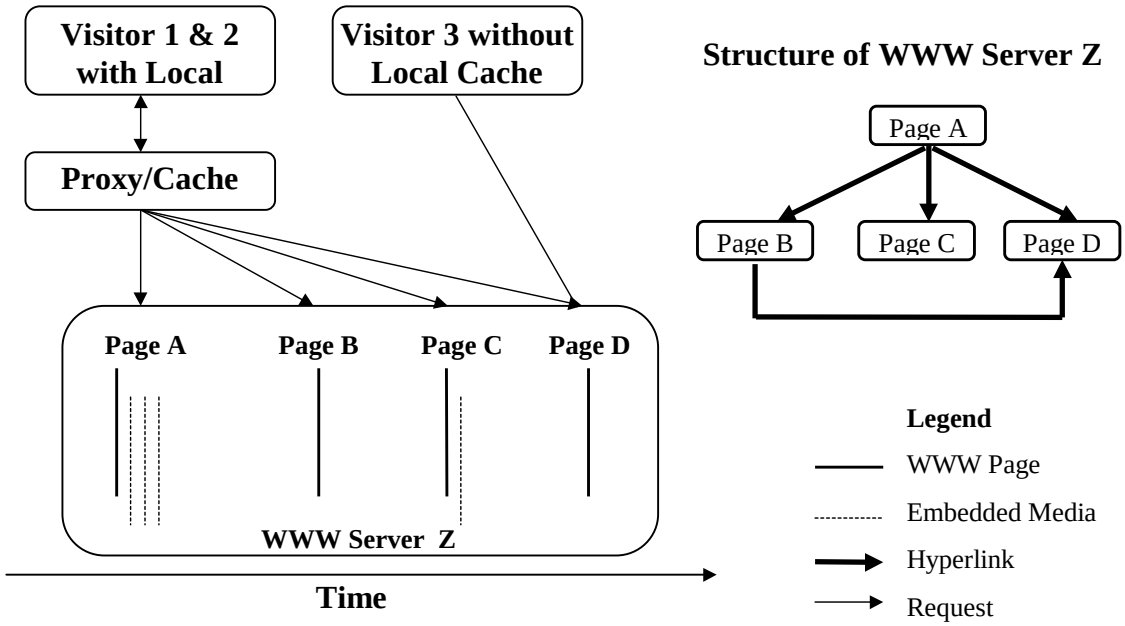

Figure1. This figure represents a visitor making requests for Pages A, B, C, and D at Site Z. Another visitor is shown making a simultaneous request for Page D. The topology of the site is depicted in the upper right hand corner. Since Visitors 1 and 2 are both behind a proxy-cache, their requests can not be differentiated without the use of either cookies or heuristics. It is also unclear from the requests make to Server Z whether Visitor 1 requested Page D by selecting the hyperlink on Page A or the hyperlink from Page B.

The other form of caching that occurs on the Web happens at the proxy level. The scenario is very similar to the one of local caches, but in this case, since the cache is shared by many users, the server may only

receive one page request even though the page was viewed by numerous users behind the proxy. In Figure 1, if Visitors 1 and 2 request the same pages with Visitor 2 making the requests at a later time than Visitor 1, the proxy-cache would only need to issue one request per page to Sever Z and use the copy stored locally to handle requests by Visitor 2. Note that combinations of local caches and proxy/caches can exist as well as multiple proxy/caches chained together. In order to help control the caching of pages by local caches as well as proxies, the HTTP has evolved to include cache specific headers.

A common resource intensive solution to the problem of reliably determining pages views and users is to use the cache specific headers in HTTP to effectively defeat all attempts at caching pages. Despite the reasons for employing such an approach, this technique is commonly referred to as *cache-busting*. This is accomplished in several ways including sending "Cache-control: proxy-revalidate", or "Expires: <past date> headers. In order for these attempts to be effective, the caches that are incurred along the way must cooperate, that is, they must obey the headers. Unfortunately, there is not way to ensure that all caches (local as well as proxy-caches) on the WWW cooperate. As a result, even with attempts to defeat caching, an unknown number of users and page views can occur without the knowledge of the originating server. This causes the total number of page views to be inaccurately reported and therefore should not be considered a trustworthy measure.

From this discussion it ought to be clear that the reliable gathering of visitor and page usage is not an easy task. The most widely used solution to these problems include the issuing of cookies to identify visitors and the defeating of caching to determine page views. This is only half of the picture however. Once the data is collected, it must be analyzed before it is of any use. The next section describes the various flavors of statistics one might want to apply to the gathered usage data.

2 Statistical Analysis

Since page requests are discrete events, they afford several forms of statistical analyzes, with descriptive statistics being the most rudimentary and most widely used for analysis of usage data on the WWW today. What one gathers from descriptive statistics of a set of events are the frequency, mean, median, mode, minimum, maximum, standard deviations, variance, and range. For accesses to WWW sites, the frequency of page views is the most commonly reported statistic, though as noted in the above section, this statistic may be difficult to determine reliably. This form of event analysis can occur on the page level as well as on the site level for all of the different types of visitors mentioned above.

2.1 Temporal Analysis

While knowledge of how frequently an event has occurred, it tells us nothing about the interactions between events. Temporal analysis of page request events can reveal several interesting metrics but assumes that visitors can be uniquely identified within sessions (session visitors, tracked visitors, or identified visitors). This requirement exists since the sequence of events is needed to construct the temporal ordering of page requests. The following discussion also assumes that the complete and correct sequence of page request is know, though as described above, this is often not possible due to local caches.

At the page level, the time spent reading a page, or *reading time*, can be measured as the inter-arrival time between the request for the page and a subsequent page request. In Figure 1, the reading time for Page A by Visitor 1 is the distance along the *x-axis (time)* between the request for Page A and Page B. This measurement is subject to a fair amount of noise as the visitor's behavior can not be always be accurately determined, e.g., the visitor could be grabbing a cup of coffee, talking on the phone, or actually reading the page. Another problem is determining when users leave the site since HTTP only sends "get me this page" messages and not "I'm leaving this page" messages. However, if a statistically valid sample size is determined and the data collected, reasonable statements about the reading times of pages can be made. [Catledge and Pitkow 1995] first reported a session time-out period of 25.5 minutes, which was 1 ½ standard deviations from the mean of 9.3 minutes between user interface events. A time-out period of 30 minutes has become the standard used by log file analysis programs, e.g., Interse [Interse 1996] and I/PRO

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.