# WebRUM: A Model for Measuring Web-Wide Resource Usage

M.P. Evans[†] and S.M. Furnell[‡]

[†] School Of Mathematics, Kingston University, Kingston, Surrey, UK
[‡]Network Research Group, Department of Communication and Electronic Engineering, University of Plymouth, Plymouth, UK.
e-mail: M.Evans@computer.org

## Abstract

Web resource usage statistics enable server owners to monitor how their users use their web sites, and for advertisers to monitor how often their advertisements are viewed. However, there is currently no way to measure how web resources are used across the whole web. The problems of capturing the required information and providing acceptable system performance present significant hurdles to the development of such a system. Overcoming these hurdles, though, would lead to a web service that could reveal the changing interests of society, and provide deep insights into the changing nature of the web, not to mention the value it would have as a marketing tool. As such, we have developed a model, called WebRUM, which can overcome these hurdles by extending a resource migration mechanism that we have previously designed. The paper describes the mechanism, and shows how it can be extended to measure web-wide resource usage. The information stored by the model is defined, and the performance of a prototype mechanism is presented to demonstrate the effectiveness of the design.

## Keywords

Web, server usage log, access log, web metrics, Request Router, distributed database, access patterns

## 1. Introduction

Web site owners require detailed statistics on the web resources (HTML documents, images, Java applets, etc.) that their users download. Usually, this information is recorded into a web server access log, which provides a history of the resources served by the server, and so helps the server owner keep track of the usage of a web site. However, of far more interest to historians, sociologists, and others interested in the changing interests of the online society, would be a system capable of measuring the resource usage of the web as a whole. Such a system would provide dynamic insights into the changing nature of web usage, as well as accurate marketing information for the web site owner, and more effective navigation for the user.

This paper presents a model for such a system. Although the model is only described, its core architecture is based on a fully tested prototype, and is capable of scaling to many times the size of today's web (Google currently indexes 2,073,418,204 HTML documents (Google, 2002a)). The paper is presented as follows: Section 2 provides a brief overview of current techniques for recording resource usage, and their limitations. Section 3 presents our model for recording resource usage across the web. Section 4 discusses the advantages of measuring web-wide resource usage by providing examples of new applications that can use the usage

information. Finally, section 5 discusses issues and further work, before the paper provides its conclusion.

## 2. Current Techniques for Measuring Resource Usage

Measuring resource usage is important for web site owners and advertisers alike, especially as the dominant revenue stream for web site owners has traditionally been advertising. However, as the following list shows, existing techniques are unsuitable for measuring usage across the web:

- *Web Server Access Logs*
  The most common technique for recording resource usage is to analyse a web server access log, which records details about the user's request, the server's response, and the resource that was requested (see section 3.2.3). However, the information contained in a web server log is specific to that server only, and is accessible only to the server owner. Capturing web-wide information is therefore impossible.

- *Web Bugs*
  A web bug is a client-based solution that subverts the HTML *IMG* element to record usage patterns of advertisements (Krishnamurthy and Rexford, 20001). A web bug is usually a one-pixel image that is transparent, and so invisible to the end user. However, the web bug is served by an advertiser's server, rather than the server of the web page in which it is embedded, and so both servers effectively record the usage of the web page. This technique works well, as the advertiser's server can capture web-wide information. However, it has raised a number of serious privacy concerns, as a web bug can effectively track an *individual user* across the web (Bugnosis, 2000). Although the companies do this to target advertisements at users more accurately based on their browsing behaviour, it is disconcerting at best to know that your movements are being recorded without your consent or even knowledge, and that a profile of your browsing behaviour that you cannot access is being kept without your permission. In addition, the information captured is not comprehensive enough to measure web-wide usage, as web bugs can only record information on the web pages that contain them, and cannot record usage of non-HTML resources.

- *Browser modification*
  A user's browser can be modified to report the user's navigation behaviour directly to a server whenever the user navigates to a new page. Google adopts this approach with its *GoogleBar* technology (Google, 2002b), but uses it to increase and continuously update its database of indexed web pages and hyperlinks. However, such an approach requires the user to explicitly install the browser update. Furthermore, it requires a large user base for it to work, and so can only be implemented by large companies such as search engines or portals, who, understandably, would be reluctant to provide open access to such information.

- *Proxy or caching proxy server*
  An intervening proxy server can record the requests that pass through it, capturing usage information from those clients connected to it. However, this approach requires

many users to explicitly connect to the proxy server, which is unlikely, and would put a large strain on the architecture of the proxy server.

# 3. A Model for Measuring Web-Wide Resource Usage

In order to accurately measure resource usage across the web, a new approach is called for that can record the web's resource usage comprehensively, efficiently, anonymously, and openly. The new approach that we envisage involves a new web-specific name resolution service that is distinct from the Internet's Domain Name System (DNS). In previous papers (Evans and Furnell (2001), Evans and Furnell (2002)), we have proposed that the DNS is becoming increasingly unsuitable for use as the web's principle name resolution service. Specifically:

- the DNS is designed to map a hostname onto an IP address, whereas the web needs a system to map a resource name onto a location;
- the DNS deliberately constrains its namespace as it only has to deal with the names of servers, whereas the web needs an unconstrained namespace to cater for all different types of resources and the needs of their owners;
- neither the DNS nor the web's main resource identifier (the URL) has any way of storing and referencing a resource's time of creation.

Our solution to these problems was the design of a new name resolution service called the *Resource Locator Service* (RLS), which was designed principally to provide an unconstrained namespace; to enable resource migration; and to locate a resource according to its position in time as well as space. Of particular use to measuring web-wide resource usage, however, is that a client must query the RLS for the location of every *resource*, in order for the RLS to support resource migration (see Evans and Furnell (2001)). The DNS, in contrast, is only queried for the IP address of every web server. As such, the RLS implicitly captures the usage of every resource requested by the clients that use it. In addition, the RLS has been designed to be scalable, and can support a system many times the size of the web. The RLS therefore provides an ideal platform for the design of a system to measure web-wide resource usage.

## 3.1 An Overview of the Resource Locator Service

The RLS has already been described in detail in Evans and Furnell (2001), and Evans and Furnell (2002). However, this section provides a brief overview.

### 3.1.1 Architectural Overview

The RLS is designed to locate a resource when given its name, and can do so transparently to either the client, or the server. It is backwards compatible with all web entities, ensuring it can be used wherever it is required. Its architecture comprises a distributed database, deployed across a network of nodes called *Locators* that form the RLS's *Locator Network*. A Locator performs a similar role to the DNS, but with granularity at the resource level.

Figure 1 shows a high level view of the RLS. The Locator Network provides the RLS's resolution service, mapping a resource's name onto its location, and redirecting the client to the resource's correct location using the HTTP redirect mechanism (Fielding et al, 1999). Although this is not the most efficient approach, it facilitates backwards compatibility, enabling all web entities to use the RLS.

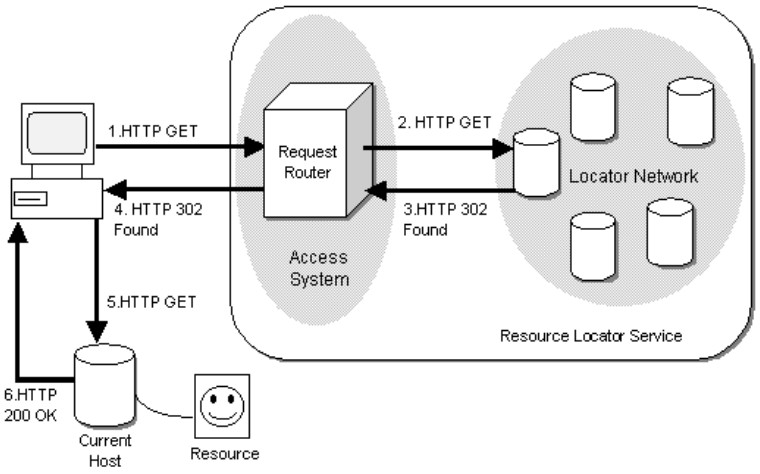


**Figure 1 - Architecture of the Resource Locator Service**

In order for the client to find the Locator that holds the required name/location mapping, some form of mediation is required between the client and the Locator Network that can transparently route the client's request without requiring any modifications to the client or server. The RLS achieves this through the use of a Request Router, which routes HTTP requests to an appropriate Locator.

### 3.1.2 The Request Router

The Request Router (RR) is the key to the system. It is a scalable component that can route a request deterministically to any of over 4 billion Locators, while adding only a single HTTP request and response as network overhead. In addition, it is extremely flexible, and can be used wherever it is required, whether it be the client, server, or elsewhere in the network. It provides transparent, scalable mediation between the web and the RLS through the use of a hash routing algorithm (based on the Cache Array Routing Protocol (CARP) (Valloppillil and Ross, 1998)), which takes a resource's name and maps it onto a hash space. The hash space is partitioned such that the name is mapped onto one and only one Locator (see Ross (1997), Thaler and Ravishankar (1997)). By requiring the Locators to be named according to a predetermined URL pattern that contains a zero-based linear numbering component (e.g. *www.locator0.net*, *www.locator1.net*, *www.locator2.net*, etc.), the name of the resource and the number of Locators in the system is all that is required to identify the appropriate Locator. In addition, the load across the nodes is equally balanced, yet there is no inter-node communication, virtually eliminating network overhead (see Evans and Furnell (2001) for more details).

### 3.1.3 Scalability

A prototype of the RLS has been developed and presented in Evans and Furnell (2001). The prototype showed that the Request Router took only 0.718 seconds to find the appropriate

Locator in a system of 100,000 Locators, each of which can potentially hold the name/location information of millions of resources (see Figure 2). Total added latency was just 1.582 seconds, but a more optimal design should significantly reduce this figure.

| Number of Locators | Download time for www.lycos.co.uk (time without RLS = 7.608 sec) | Overhead |
|---|---|---|
| 1 | 8.477 seconds | **0.869 secs** |
| 1,000 | 8.483 seconds | **0.875 secs** |
| 10,000 | 8.546 seconds | **0.938 secs** |
| 100,000 | 9.190 seconds | **1.582 secs** |
| 1,000,000 | 15.985 seconds | **8.377 secs** |

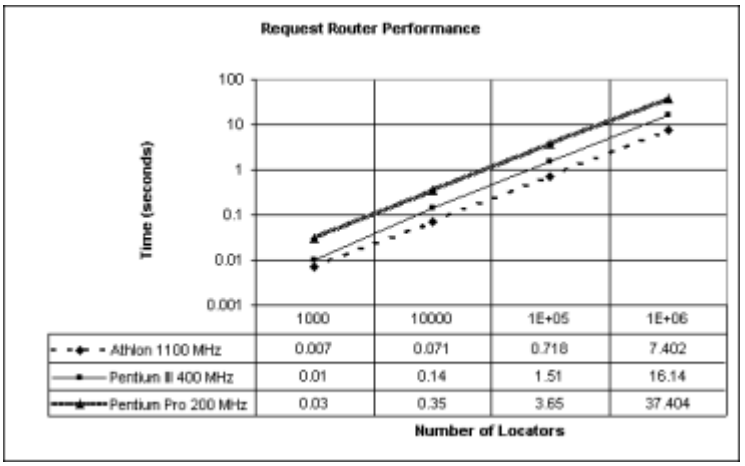


**Figure 2 - Performance of the Resource Locator Service**

### 3.1.4 Flexibility

The RR acts as an index into the distributed nodes of the Locator Network. Because it is decoupled from the RLS, it can be deployed virtually anywhere on the web. For example, it can be:

- embedded into an HTML document as a Java applet, ActiveX control or even script;
- built into a browser;
- designed as a browser plug-in;
- built into a server, or added as a server module;
- embedded within a proxy server or a reverse proxy server;

In this way, the RR can be integrated into the web at whichever part of the web's architecture it is required.

## 3.2 Extending the Resource Locator Service to Measure Web Resource Usage

As has been shown, the RLS provides an efficient and scalable platform upon which to build a system for measuring web-wide resource usage. In this section, we show how the RLS can be extended to capture web-wide usage information by recording details of the location queries made to it. In order to maintain efficiency, a separate system, called the Web Resource Usage Monitor (WebRUM), has been designed to store and manage this information.

### 3.2.1 Architectural Overview of WebRUM

WebRUM comprises a distributed database whose nodes are accessed using a Request Router specific to the system (specifically, the URL pattern *http://www.nodeX.WebRUM.net* should

be used). Each time a client queries the RLS, information about the query (such as the resource requested, a user-session identifier, etc.) is passed onto the appropriate WebRUM node using a WebRUM RR. In this way, WebRUM captures comprehensive usage information on resources from across the web (see Figure 3).

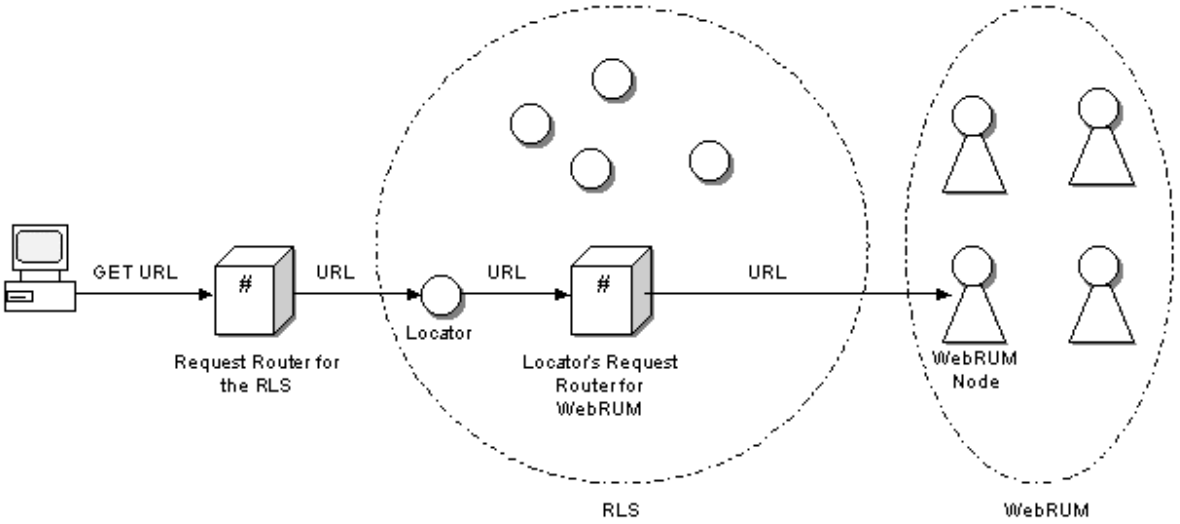


**Figure 3 - How WebRUM integrates with the RLS**

Because it uses a Request Router, all information for a specific resource is contained on one WebRUM node, significantly reducing the network overhead, and balancing the load across the entire WebRUM system. Note that the information passed to WebRUM is derived primarily from the client's header fields made in its initial request to the RLS. WebRUM can derive no information from the response of the server, as both it and the RLS are independent from the server's operation. As such, queries for resources that are not registered with the RLS (i.e. those that would cause the RLS to return an *Error 404* message) must not be passed onto WebRUM. In this way, WebRUM does not measure the usage of resources that cannot be located.

### 3.2.2 Addressing Privacy Concerns

For WebRUM to be adopted, it must enable individual users to retain their anonymity, yet still enable individual user sessions to be recorded. The RLS enforces this by making the user's IP address or hostname persistent yet anonymous (e.g. by using a hash function), and passing this onto WebRUM. In this way, the same identifier used across different HTTP requests from the same user session can be persisted, but WebRUM cannot use it to identify the specific user.

### 3.2.3 Defining the Information Stored by WebRUM

The information captured by WebRUM is a combination of that contained within existing web server log files, and the client's HTTP request header. Tables 1 and 2 show the information stored by log files that are compliant with the two most common log file formats, the Common Log Format (CLF, defined in Luotonen (1995)) and the Extended Common Log Format (ECLF, defined in Hallam-Baker and Behlendorf (1996)). As can be seen, some of

this information, such as Request Processing Time, is server-specific, and so of no use to WebRUM, while other items of information, such as Response Code, is returned by the server, and so is not accessible. As such, WebRUM must use what it can from these log files, plus some items of information extracted from the HTTP request. Table 3 shows the minimum set of information that WebRUM must store. This set can be increased by storing resource attributes in the RLS (e.g. byte size).

| Field | Description |
|---|---|
| Remote host | Remote hostname or IP address of client |
| Remote logname | The remote logname of the user (i.e. the logname the user uses to log onto their current client. In practice, this information is rarely present). |
| Authenticated username | User's username (note that this will only be present if the associated resource requires that the user authenticate herself before the resource can be transferred). |
| Date | Date and time of the request |
| Request line | The request line of the request header (i.e. Method, URI, and protocol version) |
| Response code | The HTTP status code returned to the client |
| Bytes | The content length of the resource returned to the client |

**Table 1 - Fields contained within the Common Log Format (Luotonen, 1995)**

| Field | Description |
|---|---|
| Referer | The URI of the web page from where the resource was requested |
| Request Processing Time | The length of time taken from when the request was received to when the response was sent |
| User Agent | The name and other details of the application that sent the request. Information can include such details as operating system and hardware platform upon which the application was running. |

**Table 2 – Additional Commonly Supported Fields used in ECLF Log Files (Krishnamurthy and Rexford, 2001)**

| Field | Description |
|---|---|
| Remote host | Anonymised hostname of client |
| Referer | The URI of the web page from where the resource was requested. |
| User Agent | The name and other details of the application that sent the request. Information can include such details as operating system and hardware platform upon which the application was running. |
| Date | Date and time of the request (recorded by the WebRUM node at the time it receives the information from the Locator (GMT), formatted according to RFC 1123 (Braden, 1989). |
| Request line | The request line of the request header (i.e. Method, URI, and protocol version) |
| Language | The language that the client requires the resource to be in, as specified by the Accept-Language header of the initial request |

**Table 3 - Information Stored by WebRUM**

### 3.2.4 Accessing WebRUM's Information

Accessing the information from the correct WebRUM node is very simple. To retrieve information about a specific resource, a client application needs the name of the resource (usually its URL), and a WebRUM RR. The resource name acts as an index into WebRUM, identifying the WebRUM node that stores the usage information for that resource. Once the appropriate node is found, retrieving the required information from it will depend upon the type of query the user wishes to make. To facilitate this, we intend to use the newly defined XQuery query language (Boag et al, 2002), enabling the user to send an SQL-like query formatted in XML to the appropriate WebRUM node via HTTP. At present, however, this feature is undefined.

## 4. Applications of WebRUM

Supposing that WebRUM was embraced by the web community and widely supported, what are the types of queries that will be asked of it? Some examples include:

- What are the most/least popular resources/images/web pages etc. on the web?
- How have the most/least popular resources changed over time?
- How have the most/least popular resources changed during significant events (e.g. The Queen's Golden Jubilee, September 11$^{th}$, etc.)
- Which languages are the most popular on the web?
- Which countries use the web the most/least?
- How many bytes of information are transferred in a given period of time?
- How many resources/images/web pages/etc. are there on the web?

Such statistics provide an insight into the changing nature of the online society, and of society in general. However, WebRUM can also act as a platform for the provision of other services that cannot currently be provided, as the following sub-sections describe.

### 4.1 Enhanced Navigation

WebRUM can enhance the user's ability to navigate the web by dynamically providing information about a hyperlink *before* the user clicks on it. In this way, the user can make a more informed judgement on whether or not to see the resource referenced by the hyperlink before 'rewarding' it with a hit.

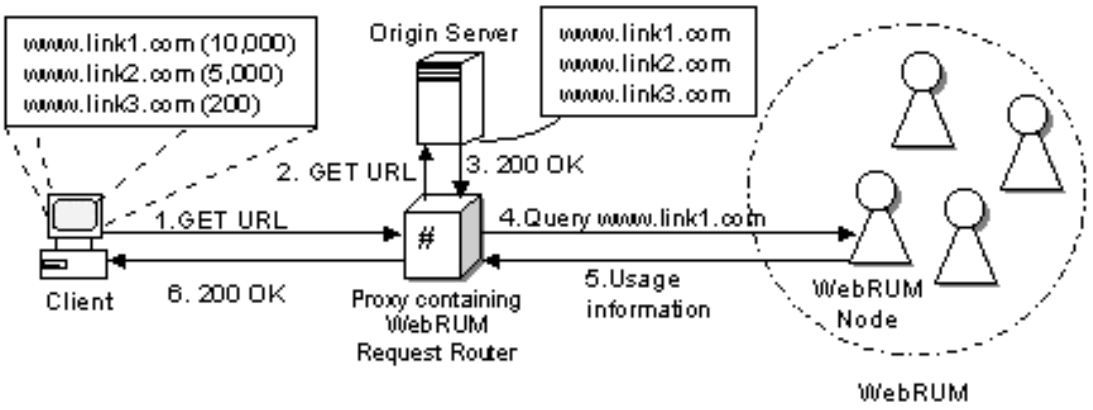


**Figure 4 - Using WebRUM to Enhance Navigation**

Figure 4 shows how this can be achieved. Here, a proxy server intercepts the client's request for a web page, and queries WebRUM to determine the number of times the hyperlinks within the web page have been clicked. The proxy then dynamically inserts the usage information into the web page, enabling the user to see at a glance the popularity of each hyperlink.
This is a trivial example of how WebRUM's information can be used to enhance user navigation. Other studies have found strong regularities in users' browsing behaviour (e.g., Hochheiser and Schneiderman, 1999, Huberman et al., 1998), indicating that more complex heuristics can be applied to WebRUM's information to further enhance navigation.

### 4.2 Charting the Web's Traffic Patterns

Many experiments have been conducted that attempt to display the structure of the web (see Dodge and Kitchin (2001) for many examples). However, WebRUM will be able to provide far more detail, providing dynamic charts that provide the web site owner with information such as:

- Which hyperlinks linking to the owner's resource provide the most/least traffic?
- How many people visit the web pages that link to the owner's resource?
- How much traffic does a web page direct to the owner's resource compared with the traffic that the page directs to other resources?
- Which hyperlinks on the web would provide the most/least traffic?

## 5. Issues and Further Work

WebRUM could change the nature of the web fundamentally, providing a unique tool for exploring the changing interests of society. However, there are a number of issues that must be overcome if it is to be deployed successfully:

- *WebRUM relies upon wide adoption of the RLS.*
  The RLS is still in prototype form, and so WebRUM cannot be developed in the near future. However, the two systems can be separated, enabling WebRUM to be self-supporting if necessary.

- *Performance*
  WebRUM can scale to support over 4 billion individual nodes. However, the processing and network demands on each node may be huge, depending upon the nature and volume of queries sent to it. In addition, collating information from across WebRUM nodes would increase the load and network overheads of the system. Further work must examine the use of caching techniques and distinct caching servers that simply return results to common queries in order to manage the load.

- *Security*
  WebRUM nodes may be at risk from Denial of Service attacks (Krishnamurthy and Rexford, 2001). In addition, care should be taken to ensure only genuine RLS Locators provide the usage information.

## 6. Conclusion

We have a presented a model for measuring web-wide resource usage that has the potential to reveal insights into society, including its hopes, fears, interests and concerns. In addition, the model can provide a powerful tool for web site owners and advertisers, providing an independent source of quality marketing information. Although there is much work still to be done, we believe the concept of measuring web-wide resource usage has the potential to transform the web.

## 7. References


Boag, S., Chamberlin, D., Fernandez, M., Florescu, D., Robie, J., Simeon, J., and Stefanescu, M. (2002), "XQuery 1.0: An XML Query Language", W3C Working Draft, 30th April 2002, http://www.w3.org/TR/xquery.

Braden, R. (1989), "Requirements for Internet Hosts - Communication layers, STD 3", RFC 1123, October 1989.

Bugnosis (2000), "Web Bug FAQ", http://www.bugnosis.org/faq.html

Dodge, M., and Kitchin, R., (2001), "Atlas of Cyberspace", Addison Wesley, 2001.

Evans, M.P., and Furnell, S.M. (2001), "The Resource Locator Service: Fixing a Flaw in the Web", Computer Networks, Vol. 37 (3-4) (2001) pp. 307–330, November 2001.

Evans, M.P., and Furnell, S.M. (2002), "A Web-Based Resource Migration Protocol Using WebDAV", in: Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii, May 7th-11th 2002, http://www2002.org/CDROM/refereed/359/index.html

Fielding, R, Gettys, J., Mogul, J.C., Nielsen, H.F., Masinter, L., Leach, P., and Berners-Lee, T. (1999), "HyperText Transfer Protocol – HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

Google (2002a), Google Search Engine web site, http://www.google.com

Google (2002b), Google Toolbar page, http://toolbar.google.com/

Hallam-Baker, P.M., and Behlendorf, B. (1996), Extended Log File Format, http://www.w3.org/TR/WD-logfile.html, March 1996

Hochheiser, H., and Schneiderman, B. (1999), “Understanding Patterns of User Visits to Web Sites: Interactive Starfield Visualizations of WWW Log Data”, In: Proceedings of ASIS '99, 1999.

Huberman, B.A., Pirolli, P.L.T., Pitkow, J.E., and Lukse, R.M. (1998), “Strong Regularities in World Wide Surfing”, Science, Vol.280, 3rd April 1998

Krishnamurthy, B., and Rexford, J (2001), “Web Protocols and Practice: HTTP 1.1, Networking Protocols, Caching, and Traffic Measurement”, Addison Wesley, 2001.

Luotonen, A., (1995), “The Common Logfile Format”, July 1995, http://www.w3.org/Daemon/User/Config/Logging.html.

Ross, K.W. (1997) “Hash Routing for Collections of Shared Web Caches”, IEEE Network, November/December 1997, pp. 37–44.

Thaler, D.G. and Ravishankar, C.V. (1998), “Using Name Based Mappings to Increase Hit Rates”, IEEE/ACM Transactions on Networking, 6(1), Feb. 1998.

Valloppillil, V., and Ross, K.W. (1998), “Cache Array Routing Protocol v1.0”, Internet Draft, draft-vinod-carp-v1-02.txt, February 26, 1998, http://www.cs-ipv6.lancs.ac.uk/ipv6/documents/standards/general-comms/internet-drafts/draft-vinod-carp-v1-03.txt