

# A Multicollaborative Push-Caching HTTP Protocol for the WWW

Alejandro López-Ortiz  - Daniel M. Germán 

## Abstract:

We propose a caching protocol designed to automatically mirror heavily accessed WWW pages in a distributed and temporal fashion. The proposed caching mechanism differs from proxy type mechanisms in that it caches according to load pattern at the server side, instead of access patterns at the client-side LAN, in a Demand-based Document Dissemination (DDD) system fashion. This type of server initiated caching scheme has been termed push-caching. As well, the proposed caching scheme incorporates topological caching functions. The proposed protocol is orthogonal to other extensions to the HTTP protocol and other caching schemes already in use.

## Table of Contents

1. [Introduction and Motivation](#)
  2. [Traffic Overload](#)
  3. [A multi-collaborative cache for the World Wide Web](#)
    1. [Overview](#)
    2. [High Level Specification](#)
    3. [A serving cache](#)
    4. [Algorithm](#)
  4. [Benefits of Multicollaborative Push-Caching](#)
    1. [Demand Based Modeling](#)
    2. [Geographic/Topological Caching](#)
  5. [Conclusions](#)
  6. [References](#)
- 

## Introduction and Motivation

The World Wide Web has seen a dramatic increase in popularity during the past sixteen months. As a result of this success, there have been interruptions of service from the part of heavily accessed Web sites, such as Cool Site of the Day, WWW indexing services and other popular servers. It is clear that most organizations do not have the bandwidth or computational resources required to support the heavy loads associated with a popular site.

The Net has been identified as a great resource for the dissemination of information: a printing press on everybody's hands. Ironically, the more popular a personal site is, the likelier it is to be discontinued, due to server load. Only large commercial operators can meet the computing requirements of a frequently accessed site. Distributed methods of information broadcast, such as a posting to an Usenet newsgroup or a radio broadcast have a fix set of demands on the broadcaster which is independent of the number of

people who actually read or listen to the information distributed. On the other hand, other protocols such as WWW and cable TV require additional investment for every new user. In the latter case, costs are simply passed on to the consumer. But as the WWW is based on the free distribution of information, similar cost recovery schemes are not equally feasible.

As well, transmission of information is highly redundant say, as opposed to Usenet. While a posting to a newsgroup essentially travels any given part of the network but once, a WWW page accessed by two users from the same organization at the same time generates two independent transmissions [Gwe94, Gwe95]. This duplication of broadcast is not unlike that generated by FTP. Indeed, the Alex caching scheme [Cat92] was designed to alleviate these problems while at the same time providing a more familiar user interface to FTP transmissions. As FTP traffic has not increased rapidly enough to be a threat to network bandwidth, Alex has remained subutilized in spite of its clear benefits. On the other hand, traffic on the WWW is fast reaching the critical point of saturation.

To this regard, the National Science Foundation deemed as a critical research topic for the National Information Infrastructure to “develop new technologies for organizing cache memories and other buffering schemes to alleviate memory and network latency and increase bandwidth” (iNSF94) as quoted in [Bes95]).

Because of these considerations several research groups are studying the impact on traffic by proxy/cache additions to the HTTP protocol. It has been shown that the addition of *organization edge* proxies for incoming traffic, as well as remote cache servers, would result in a noticeable reduction in expected traffic [Bes95].

Currently, some popular browsers, such as Netscape, and some large organizations, such as DEC [Jon94], provide some degree of local caching for their users. As traditionally configured, Netscape retains the last few images and text files accessed in a cache directory which is accessible only to the requestor. This results in a reduction of network traffic and latency. Load in the server, however, is only marginally reduced as the cache is accessible to one user alone. Similarly, the internal DEC network is equipped with a caching relay for the organization (not unlike that of Netscape) to which all internal user requests are first directed. If a hit occurs, the information is served to the user, else the relay host forwards the request to the actual server indicated by the URL. Again, the impact on server load is minor.

At this time, there are proposed modifications to the HTTP protocol, currently implemented in several HTTP servers, that make caching possible for pages which are frequently modified. This new command in the protocol, called “if-modified-since” allows a caching client, the so-called proxies, to verify if a cached document has been recently modified. If so, it requests a fresh version of it, otherwise it serves it locally to the user, without generating additional external traffic.

In early August 1994, we started tracing access patterns at a local Web server, and by late October 1994, our measurements confirmed the observations stated in [CBP94]. Thus it became clear that a form of server initiated caching would eventually be necessary. The last fourteen months have, if anything, exceeded our expectations, and more than justify the requirement of server-side caching. However, at this time, there seems to be little work on this area, and none whatsoever at the level of HTTP specifications.

In this work we present a push-caching scheme which, as opposed to [Gwe95], is additional to current client-side proxying schemes.

We propose a collection of collaborative proxies that cooperate in caching of WWW documents. The protocol is designed to implement a Demand-based Document Dissemination system or DDD (as proposed by [Bes95]), meaning that the request for caching is issued by the server depending on local load and size statistics. Among the advantages of a DDD system are that automatic mirroring is provided as well as the efficient use of resources, since documents are cached according to demand and geographic proximity.

A significant difference with other caching schemes is that in this multicollaborative system, caching is initiated by the server. As the server is aware of the modification frequency of a document (say by verifying the last modified date), it avoids many of the problems posed by “mutable” documents in [Bes95] by means of not caching highly mutable documents. (Mutable documents are those which are frequently “updated”.) As well, this scheme partially implements the ideas of geographic and topological caching.

In section 2, we define the different types of traffic generated by Web accesses. Section 3 describes the caching scheme and discusses specific issues of its implementation. In section 4 we describe the results of computer modelling of the caching scheme.

## Traffic Overload

Traffic generated by WWW transactions can be classified in four different classes.

1. LAN External Traffic
2. LAN Internal Traffic
3. WAN Single Source Traffic
4. WAN Multi Source Traffic

This distinction is very relevant, as different types of solutions are required to reduce different types of traffic. First, let us specify what we mean by each category.

**LAN External Traffic** is generated by external users accessing locally maintained documents in different internal hosts. Thus, if in the local network configuration the server host A is connected to a host B which in turn is connected to the external Internet gateway server C, each access to a document in A causes network traffic to be increased locally between A, B and C; which in most cases entail a degradation in transmission speed for local users connected to the subnetworks A-B and B-C.

**LAN Internal Traffic** are local users repeatedly accessing, across an organization’s internal network, an internal or external WWW document. Once again, traffic will be increased on the local subnetworks that contain gateways connecting the client host to the server host.

**WAN Single Source Traffic** is generated by users from the same first or second level domain, such as a country or organization, accessing the same WWW page. In this case we have an international or backbone wire carrying several copies of the same page within relatively short spans of time (see figure 2).

**WAN Multi Source Traffic** occurs when a popular WWW page is widely accessed across the globe in such a way that no individual organization generates a significant percentage of the traffic, while at the

same time, traffic is high enough to bring the server to a halt. In this case, the main objective is to reduce server load rather than network load.

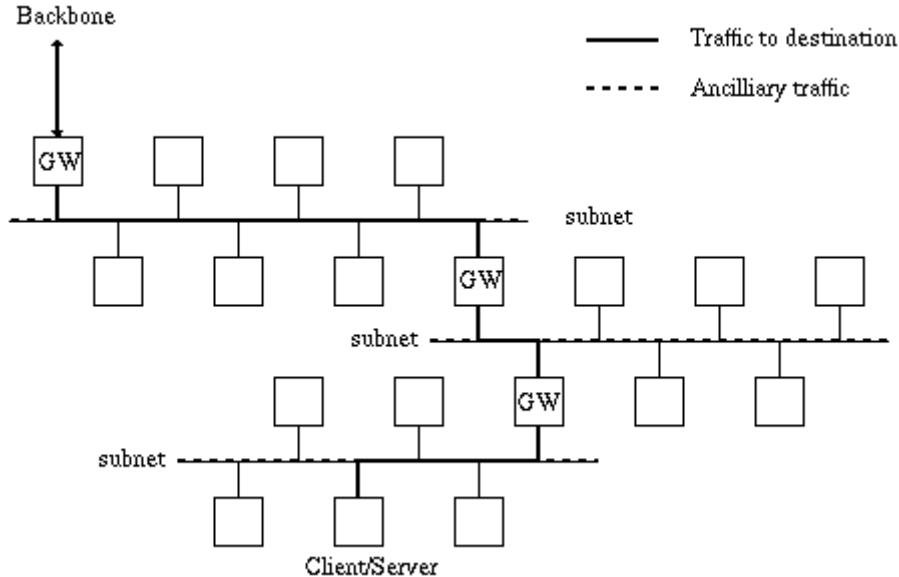


Figure 1. LAN Internal/External Traffic

In the case of figure 1, we see that a client requesting a document generates LAN Internal Traffic that could be avoided by proxying in an internal gateway such as with CERN proxy, or by browser caching, such as the one used by Netscape. These two schemes, depending on the specific page being accessed, may also reduce WAN Single and Multi Source Traffic.

The same computer, serving a document generates LAN External Traffic which can be avoided by means of server side caching at the gateway between the LAN and the backbone or better, even further down, if a cache is known to exist closer to the client.

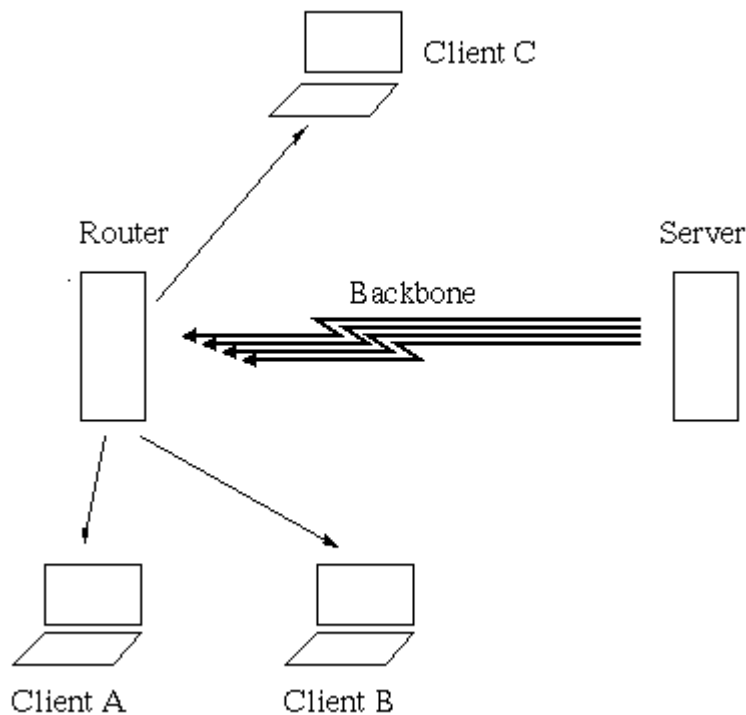


Figure 2. WAN Single Source Traffic

# A multi-collaborative cache for the World Wide Web

## Overview

Our proposal is designed to obviate LAN External Traffic and WAN Single and MultiSource Traffic, and it is compatible with Netscape and CERN type proxying. As well, it takes advantage of the CERN supported **If-Modified-Since** modifications.

The server only forwards clients to caching servers which are known to hold fresh copies. At the caching side, copies are held as long as indicated in a header of each file served or until the space is needed, whichever occurs first. As the server knows the vital statistics of each file (such as size, last modification date, and access frequency, and in some instances the expiry date of a document), it can automatically place a expiry date derived from these figures using a formula of the form

$$\text{Expiry Date} = (\text{Today}) + (\text{Frequency of Accesses}) + (\text{Time since last change}) + (\text{Size});$$

where each of the terms might be weighted in an appropriate form.

The proposed protocol is somewhat akin to proxies, most of which have been investigated in terms of physical proximity be it in LANs or slightly wider geographical areas [Bes95, Gwe94, Gwe95]. To our knowledge, none of these proposals have gone beyond a detailed description of the problem. However, those studies provide valuable data for the design of a caching protocol.

## High Level Specification

The caching scheme works as follows:

1. The client requests a document to a server via HTTP. If the client can push-cache the file, it informs the server via the modifier commands on the request line. Say,

```
slip26.ISP.net% telnet daisy.uwaterloo.ca 80
GET /~alopez-/test.html HTTP/N.N
push-caching-proxy PCP.ISP.net
```

The server then serves the document to the caching proxy which in turn serves it to the client.

2. If the server does not have a cache copy suitable for the client, according to local size, access statistics, and geographic/topological considerations, it serves the document along with an expiry date (or a no-cache pragma).

Otherwise, the server redirects the client to a push-caching-proxy that has a current copy of the information, according to the server's own local tables. It then replies to the original request with the URL of such caching-proxy. The procedure of choosing the caching-proxy can involve parameters such as heuristic physical proximity of the proxy selected to the client. A possible format for the caching redirection is to reply with an URL, say,

```
HTTP/N.N 305 Push Cached
Content-type: command
Location: http://PCP.ISP.net/cachedir/daisy.uwaterloo.ca/~alopez-/test.html
```

3. The client then goes to the proxy and requests the same information.
4. The proxy finds out whether it still has the information. If it does, it serves the information back to the client.
5. Otherwise, it replies with an error code.
 

```
306 Cached file has been removed
```
6. In the latter case, the client reissues the request to the server, with an "issue" modifier command. The server then replies back with the information, so no further redirection exists. If the client is able to cash, it is incorporated to the server tables as a temporary holder of the information, and the PCP server is removed from the server's table.

**Details.** Since hyperdocuments are normally composed of text and images, the whole collection of files that compose the hyperdocument can be cached at the same time; hence, only one request to the original server will be necessary, on the behalf that the proxy will contain a valid copy of all the files. The client then will query the proxy for all required files before going to the main server.

### Advantages.


- Files will travel shorter distances if the server can decide on the physical location of the client and proxies. Domain names provide a ready approximation of this information at a national level. Since

transoceanic links are severe bottlenecks, this step alone improves throughput.

- Big files such as images, sound files and movie files are the main targets for this kind of caching, since they create a significant percentage of traffic.
- At this time, a large amount of resources are required by the owners of a popular site, even if they are providing a free service. The proposed scheme distributes the load in an egalitarian form to recent users of the information.

**Drawbacks.** As the server can use local time and size statistics, the server can ensure that small or lightly accessed documents are not cached, thus resulting in no increase of traffic. To our knowledge the only drawback is that all clients should ideally become potential cache servers, which increases the complexity of software on the client side.

## A serving cache

Most of the current caching research has concentrated on the client side. However, in big organizations, with many WWW servers  significant traffic is generated in the local network while serving external requests. This traffic, termed LAN External Traffic, may become a significant percentage of the traffic broadcasted over the local network.

The cache proposed in the previous section provides a natural solution to this problem. Within an organization, a server on the boundary of the local network may cache often accessed documents as requested by the server. When an external hit occurs, the server forwards the request to the boundary server thus obviating the need for internal traffic. The set of documents maintained in the external host changes dynamically as well. In principle, all of LAN External Traffic can be avoided by means of a boundary caching server. As studies show that a 70-80% of traffic is generated by accesses to a few documents [Bes95, Gwe95] it is possible to reduce LAN External Traffic loads by that amount with relatively little additional investment. As the popularity of the Web continues to grow, it is expected that the percentage of savings as part of the total traffic (external or otherwise) will also increase.

## Algorithm

1. The client requests a page to the “authoritative” server. The server then redirects it to the caching-proxy, similarly to the algorithm for collaborative caching proxies.
2. The caching-proxy receives all the requests, and serves them. Whenever necessary it goes to the “authoritative” server for a fresh copy of the data.

Advantages:

- This scheme avoids unnecessary traffic in the local network, normally generated by external requests.
- This scheme can also be used within an organization that has a large number of servers that are queried from within, by means of caching at gateway servers between local subnetworks (see figure 1).
- This model can be combined with a normal caching-proxy that caches external pages going *inside* the organization, providing a both-ways cache.
- In organizations in which the existence of WWW servers is not a critical part of their mission, the outgoing-proxy can provide a way to adapt the traffic to the resources available, so the network does not become overloaded with unwanted network activity.

- In principle, a server may inform a PCP that a document has been updated and serve a fresh copy.

Disadvantages:

- It might require additional storage in the gateway machine, and in some extreme high traffic loads a dedicated machine will be necessary. However, substantial savings on network traffic and latency for Web and non Web-users more than justify this cost.

## Benefits of Multicollaborative Push-Caching

We ran simulations under different criteria for forwarding requests to a caching site. We used access logs to daisy.uwaterloo.ca which is a DEC Alpha 3000/800S running NCSA HTTPd 1.3 server and connected to a 1.5Mbs local ethernet to a shared T1 line to the Internet backbone. It serves an average of 5,000 files a day for a total of 50MB. Even though the site is located outside the US, the vast majority of the accesses originate there, as it is common for sites which are mostly in English.

Because daisy's connection to the backbone is relatively fast, the number of requests is not altered by traffic load. This might skew some of the statistics when we assume that the same log was created under a slower connection. For one, often a second request occurs close to the first one, which would not be possible if the first file is being slowly transmitted. Secondly, users tend to browse more pages on fast sites than on slow ones.

On the other hand, if the caching protocol proposed is implemented, users would experience reduced latency and thus demand pages in a pattern similar to the one registered on daisy's logs which compensates some of the high traffic patterns registered.

## Demand Based Modeling

The first simulation uses demand based accesses patterns. That is, a maximum desirable load was set, and whenever this value was exceeded all requests were forwarded to a caching site whenever one existed. It turns out that even for a relatively slow 64kbs connection, the threshold is rarely surpassed. The savings reported were of the order of 3.3%. At this time then, latency is correlated to throughput rather than bandwidth, even for relatively slow connections.

However, the amount of traffic on the Web has been estimated to double every 11 to 16 weeks. At this rate of growth the benefits of Demand Based Caching will become non-negligible in March 96, significant by mid June, and almost a necessity by the end of next year (see figure 3).



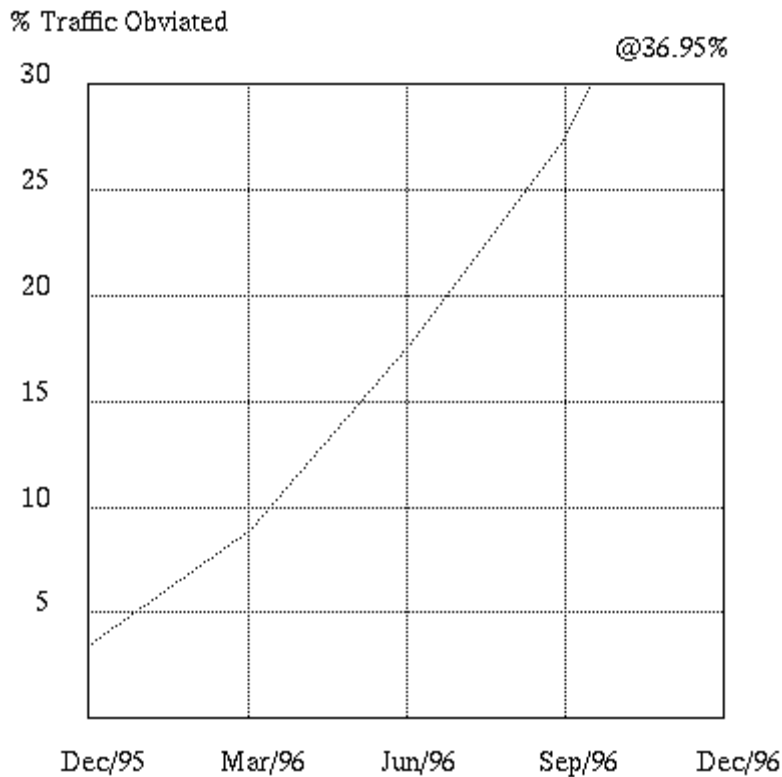


Figure 3. Amount of Traffic Obviated by DDD

Within a year, even sites supporting 500kbs with a maximum 30% network load devoted to Web traffic would benefit from nonnegligible reductions under this demand based caching scheme.

Notice also that this scheme allows for servers inside a LAN to refer all requests to a proxy in the same network acting as gateway between the LAN and the external Internet connection. This has the effect of eliminating LAN External Traffic almost on its entirety (initial requests are still served locally).

## Geographic/Topological Caching

As we mentioned before, an important consideration is to reduce the amount of traffic on the backbone. For this, clients need to be referred to proxies which are, if possible, located in close geographical proximity. The problem of determining geographic proximity has been studied in the past, particularly for the purposes of automated multicast distribution. At this time, it seems that there is no single efficient scheme for determining geographic proximity in all cases [GS95]. Among the cheapest and most efficient is the use of topological information given by IP addresses. While indeed is true that two computers within the same subnetwork domain need not be geographically close, it is typically the case that at least there is high connectivity among them.

In these simulations, when a document is requested, the server uses the IP address to determine if the document has been recently requested by a computer on the same network as the current requestor.

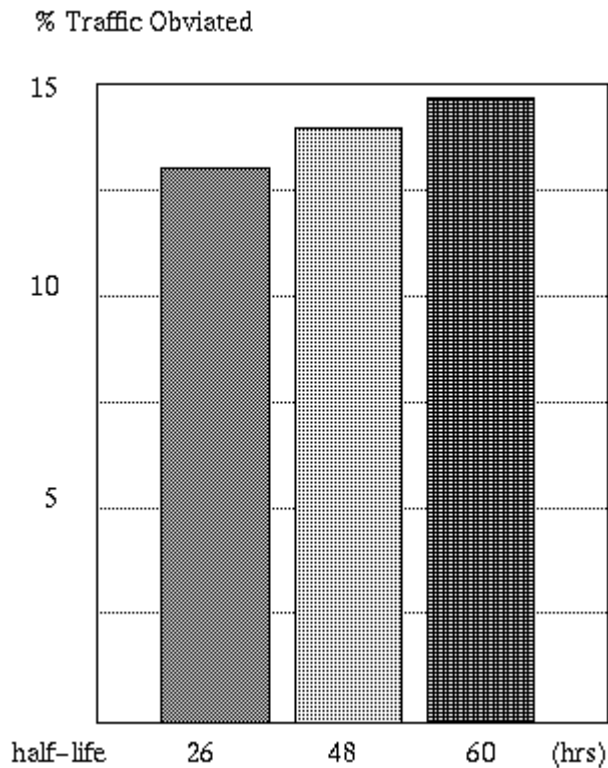


Figure 4. Amount of Traffic Obviated by Geographic/Topological Caching

If so, the server forwards the request to that machine which serves the request. If not, the document is served, and the cache tables are updated to reflect the caching of the document on the local caching proxy indicated by the client. Cached documents are assumed to decay with time. We assumed three different half-lifespans for cached documents of 26, 48 and 60 hours. The reduction in traffic in the first case was 12.62%, in the second 14.06% and on the third 14.85%.

Most important is the fact that the savings were reflected not only in the local server load, but also across the network as requests were forwarded to a nearer server to the client.

Thus geographic caching reduces WAN Internal and External Traffic, as well as LAN External Traffic.

## Conclusions

We introduced a scheme that can be reasonably implemented over the current infrastructure provided by HTTP. This scheme uses server based proxying which complements client side proxying in a natural way. It also incorporates demand-based document dissemination and geographic-based caching fairly naturally as well as “reverse” proxying, in the sense that client side proxies reside typically on the boundary of the client network and the Internet, and the proposed server based proxy defaults to a host on the boundary of the **server** network and the Internet.

## References

**Bal95** T. Ballardie. Core Based Tree (CBT) Multicast, Architectural Overview and Specification.

*Internet Draft*, April, 1995.

**Bes95**

A. Bestavros, Demand-based Document Dissemination for the World-Wide Web. *Technical Report 95-003, Computer Science Department, Boston University*. February, 1995.

**BCC95**

A. Bestavros, R. L. Carter, M. E. Crovella, C. R. Cunha, A. Heddaya, S. A. Mirdad, Application-Level Document Caching in the Internet. *Technical Report BU-CS-95-002, Computer Science Department, Boston University*, Revised March, 1995.

**Cat92**

V. Cate, Alex - A Global Filesystem, *Proceedings of the Usenix File Systems Workshop*, May 1992, pp1-11.

**CBP94**

K. C. Claffy, H-W, Braun, George C. Polyzos, Tracking Long-Term Growth of the NSFNET. *Communications of the ACM*, August 1994, pp34-45.

**DWA94**

M. D. Dahlin, R. Y. Wang, T. E. Anderson, D.A. Patterson, Cooperative Caching: Using Remote Client Memory to Improve File System Performance. *Proceedings of the First Symposium on Operating Systems Design and Implementation (OSDI)*, pp.267-280, 1994.

**DEF95**

S. Deering, D. Estrin, D. Farinacci, Van Jacobson, C. Liu, L. Wei, Protocol Independent Multicast (PIM): Motivation and Architecture. *Internet Draft*, January, 1995.

**DEF95a**

S. Deering, D. Estrin, D. Farinacci, Van Jacobson, C. Liu, L. Wei, Protocol Independent Multicast (PIM): Protocol Specification. *Internet Draft*, January, 1995.

**NSF94**

M. Foster, R. Jump. NSF Solicitation 94-75. *STIS database*, May 1994.

**Gla94**

S. Glassman, A Caching Relay for the World Wide Web, *Proceedings of the First International Conference on the World-Wide Web (WWW94)*, Elsevier, May, 1994.

**GoL91**

R. Golding, D. D. E. Long, Accessing Replicated Data in a Large-Scale Distributed System. *Technical Report 91-01, Concurrent Systems Laboratory, University of California, Santa Cruz*, January, 1991.

**GS95** J. D. Guyton, M. F. Schwartz. Locating nearby copies of replicated Internet servers. *Technical Report CU-CS-762-95, University of Colorado at Boulder*, 1995.

**Gwe94**

J. Gwertzman, M. Seltzer. The Case for Geographical Push-Caching, in VINO: The 1994 Fall

Harvest, *Technical Report TR-34-94*, Center for Research in Computing Technology, Harvard University, December, 1994.

### Gwe95

J. Gwertzman. Autonomous Replication in Wide-Area Internetworks *Technical Report TR-19-95*, Center for Research in Computing Technology, Harvard University, April, 1995.

### Jon94

R. Jones, Digital's World-Wide Web Server. A Case Study. Proceedings of the *First International Conference on the World-Wide Web (WWW94)*, Elsevier, May, 1994.

### Kri95

D. M. Kristol, A Proposed Extension Mechanism for HTTP, *Internet Draft*, January, 1995.

### LA94

A. Luotonen, K. Altis, World-Wide Web Proxies, [http://www.city.net/cnx/kevin\\_altis/papers/Proxies/](http://www.city.net/cnx/kevin_altis/papers/Proxies/), May, 1994.

### PR94

J. E. Pitkow, M. M. Recker, A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns. Proceedings of the *First International Conference on the World-Wide Web (WWW94)*, Elsevier, May, 1994.

### Sed94

J. Sedayao, Mosaic Will Kill My Network! Studying Network Traffic, Proceedings of the *First International Conference on the World-Wide Web (WWW94)*, Elsevier, May, 1994.

### WE95

L. Wei, D. Estrin, The Trade-offs of Multicast Trees and Algorithms. *Internet Draft*, March, 1995.

## Footnotes

Alejandro López-Ortiz

Research Scientist, Open Text Corp. 180 Columbia St. W. Waterloo, Ontario N2L 3L3. Canada.  
E-mail: alexlo@opentext.com. Part of this research was done while at the Department of  
Computer Science, University of Waterloo.

Daniel M. Germán

Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1. Canada.  
E-mail: dmg@csg.UWaterloo.ca

...servers

The University of Waterloo has a total of 148 servers spread around the campus.

*Alex Lopez-Ortiz*

*Thu Dec 28 23:36:17 EST 1995*