# WWW Traffic Reduction and Load Balancing through Server-Based Caching

Azer Bestavros
*Boston University*

*This caching protocol exploits the geographic and temporal locality of reference exhibited in client-access patterns. It automatically and dynamically disseminates popular data toward consumers.*

A recent solicitation from the National Science Foundation deemed two research topics critical for projected applications of the National Information Infrastructure:[1]

- New techniques for organizing cache memories and other buffering schemes to alleviate memory and network latency and to increase bandwidth.
- Partitioning and distribution of system resources throughout a distributed system to reduce the amount of data that must be moved.

Most caching studies for large distributed information systems concentrate on client-based caching, which caches recently and frequently accessed information at the client (or at a proxy thereof) in anticipation of future accesses to that information (see the "Client-based caching research" sidebar). However, these techniques are myopic: they focus exclusively on caching at a particular client or set of clients, and hence are likely to have limited impact.

The Object Caching Environments for Applications and Network Services (Oceans) group, part of Boston University's Computer Science Department, has developed a more global solution that allows the replication of information on a supply-and-demand basis. This process is controlled by servers, which unquestionably have a better view of data-access patterns than clients have.


1063-6552/97/$10.00 © 1997 IEEE

## Client-based caching research

Usually, client-based caching research has dealt with distributed file systems[1] such as the Sun NFS,[2] the Andrew File System,[3] and the Coda system.[4] Recently, some researchers have tried to extend these techniques to distributed information systems such as FTP and HTTP.

Peter Danzig, Richard Hall, and Michael Schwartz have studied caching to reduce the bandwidth requirements for the FTP protocol on the NSFnet.[5] Their study shows that a hierarchical caching system that caches files at core nodal switching subsystems reduces the NSFnet backbone traffic by 21%. Swarup Acharya and Stanley Zdonik have studied how data placement and replication affect network traffic, using file access patterns to suggest a distributed dynamic replication scheme.[6] Christos Papadimitrou, Srinivas Ramanathan, and P. Venkat Rangan have suggested a more static solution based on fixed network and storage costs for the delivery of multimedia home entertainment.[7] Susan Eggers and Randy Katz have simulated a two-level caching system that reduces network and server loads.[8] Matthew Blaze has proposed a dynamic hierarchical file system that supports demand-driven replication, whereby clients can service requests issued by other clients from the local disk cache.[9] Michael Dahlin and his colleagues have suggested a similar cooperative caching idea.[10]

Steven Glassman presented one of the earliest attempts at caching on the Web. His method organized *satellite relays* (proxy caches) into a tree-structured hierarchy, with cache misses in lower relays percolating up through higher relays until the requested object is found.[11] The performance of this caching system for a single relay with a rather small cache indicated that maintaining a fairly stable 33% hit rate is possible. Using a Zipf-based model, Glassman extrapolated the performance of such a system for large caches. (A Zipf-based model is a simulation model that uses a Zipf distribution, whose parameters are estimated empirically from traces, to synthetically model the popularity of documents.) In particular, he estimated that with an infinite-size cache, the maximum achievable hit rate is 40%. Mimi Recker and James Pitkow made another early attempt to characterize Web access patterns to engineer Web caching systems.[12] They based their model for Web information access on two metrics borrowed from psychology research on human memory: the frequency and recency rates of past accesses.

Stephen Williams and his colleagues present a taxonomy (and compare the performance) of a number of proxy cache replacement policies.[13] Their work suggests that proxy cache management should consider document sizes when making replacement decisions. Jean-Chrysostome Bolot and Philipp Hoschka have confirmed this idea.[14] They showed the usefulness of using information about document size and network load in the replacement algorithms of Web caches, based on time-series-analysis techniques of Web traffic. Marc Abrams and his colleagues have presented similar results about the inadequacy of classic LRU (least recently used) cache replacement.[15]

### References


1. J.H. Howard et al., "Scale and Performance in a Distributed File System," *ACM Trans. Computer Systems*, Vol. 6, No. 1, Feb. 1988, pp. 51–81.
2. R. Sandber et al., "Design and Implementation of the Sun Network File System," *Proc. Usenix Summer Conf.*, Usenix Assoc., Berkeley, Calif., 1985, pp. 78–91.
3. J.H. Morris et al., "Andrew: A Distributed Personal Computing Environment," *Comm. ACM*, Vol. 29, No. 3, Mar. 1986, pp. 184–201.
4. M. Satyanarayanan et al., "Coda: A Highly Available File System for a Distributed Workstation Environment," *IEEE Trans. Computers*, Vol. 39, No. 4, Apr. 1990, pp. 447–459.
5. P. Danzig, R. Hall, and M. Schwartz, "A Case for Caching File Objects inside Internetworks," Tech. Report CU-CS-642-93, Dept. of Computer Science, Univ. of Colorado, Boulder, Colo., 1993.
6. S. Acharya and S.B. Zdonik, "An Efficient Scheme for Dynamic Data Replication," Tech. Report CS-93-43, Dept. of Computer Science, Brown Univ., Providence, R.I., 1993.
7. C.H. Papadimitriou, S. Ramanathan, and P.V. Rangan, "Information Caching for Delivery of Personalized Video Programs on Home Entertainment Channels," *Proc. Int'l Conf. Multimedia Computing and Systems*, IEEE CS Press, 1994, pp. 214–223.
8. D. Muntz and P. Honeyman, "Multi-Level Caching in Distributed File Systems or Your Cache Ain't Nuthing but Trash," *Proc. Winter 1992 Usenix*, Usenix Assoc., 1992, pp. 305–313.
9. M.A. Blaze, *Caching in Large Scale Distributed File Systems*, PhD thesis, Computer Science Dept., Princeton Univ., Princeton, N.J., 1993.
10. M.D. Dahlin et al., "Cooperative Caching: Using Remote Client Memory to Improve File System Performance," *First Symp. Operating Systems Design and Implementation (OSDI)*, Usenix Assoc., 1994, pp. 267–280.
11. S. Glassman, "A Caching Relay for the World Wide Web," *Proc. First Int'l Conf. WWW*, North-Holland, Amsterdam, 1994, pp. 69–76.
12. M.M. Recker and J.E. Pitkow, "Predicting Document Access in Large, Multimedia Repositories," Tech. Report VU-GIT-94-35, Graphics, Visualization, and Usability Center, Georgia Tech, Atlanta, 1994.
13. S. Williams et al., "Removal Policies in Network Caches for World-Wide Web Documents," *http://ei.cs.vt.edu/%7esucceed/96WAASF1/*, Virgina Polytechnic Inst. and State Univ., Blacksburg, Va., 1996.
14. J.-C. Bolot and Philipp Hoschka, "Performance Engineering of the World Wide Web: Application to Dimensioning and Cache Design," *Proc. Fifth Int'l Conf. WWW*, Elsevier Press, Amsterdam, 1996; *http://www5conf.inria.fr/fich_html/papers/P44/Overview.html*.
15. M. Abrams et al., "Caching Proxies: Limitations and Potentials," *Proc. Fourth Int'l Conf. WWW*, O'Reilly & Associates, Sebastopol, Calif., 1995, pp. 312–319.

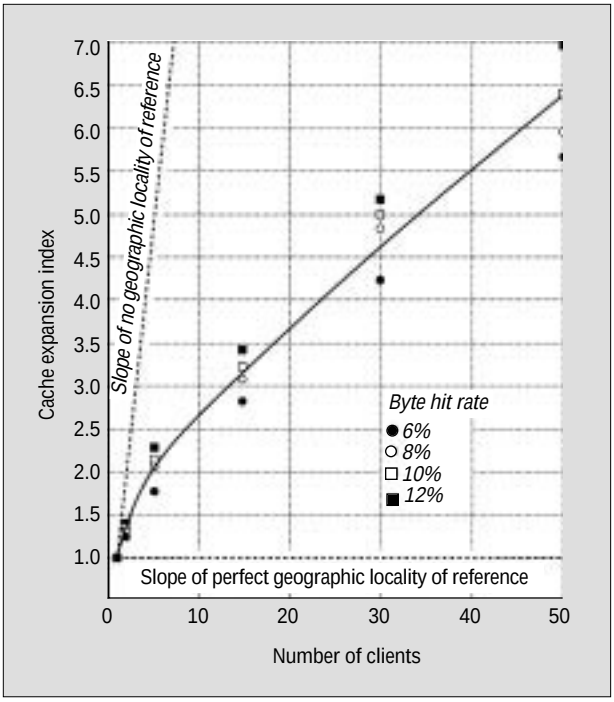


Figure 1. The cache expansion index (the rate at which the proxy cache should be inflated to maintain a constant byte hit rate).

Our protocol reduces the load on popular servers by duplicating (on other servers) a small percentage of their data. The extent of this duplication (how much, where, and on how many sites) depends on two factors: the server's popularity and the expected reduction in traffic if the dissemination is in a particular direction. In other words, our protocol provides a mechanism that automatically and dynamically disseminates popular data toward consumers—the more popular the data, the closer it gets to the clients. Demand-based dissemination of information from producers to consumers is not a new idea: it is used in the retail and newspaper businesses, among other things. In this article, I propose the same philosophy for distributed information systems. (See the "Related work" sidebar for research similar to ours.)

We used the World Wide Web as the underlying distributed computing resource to be managed. First, the WWW offers an unmatched opportunity to inspect a wide range of distributed object types, structures, and sizes. Second, thousands of institutions worldwide have fully deployed the WWW, which gives us an unparalleled opportunity to apply our findings to a real-world application.

Using extensive log data from HTTP servers, we've developed an analytical model of our protocol. This model demonstrates how to implement such dissemination, both efficiently and with minimal changes to the Internet's prevailing client-server infrastructure. Trace-driven simulations based on the data have quantified the potential performance gains from dissemination.

## Client-based caching and locality of reference

In a comprehensive study of client-based caching for the Web, the Oceans group established its limited effectiveness for very large distributed information systems.[2] The study measured the effectiveness of session caching, host caching, and LAN proxy caching, using a unique set of 5,700 client traces (almost 600,000 URL requests) that we obtained by instrumenting Mosaic.[3] We concluded that LAN proxy caching is ultimately limited by the low level of sharing of remote documents among clients of the same site. This finding agrees with Steven Glassman's predictions[4] and was further confirmed for general proxy caching by Marc Abrams and his colleagues.[5]

To understand client-based caching's limited effectiveness, we need to consider how *locality of reference* contributes to enhanced cache performance. Access patterns in a distributed information system (such as the WWW) exhibit three locality of reference properties: *temporal*, *geographical*, and *spatial*. Temporal locality implies that recently accessed objects will likely be accessed again. Geographical locality implies that an object accessed by a client will likely be accessed again by nearby clients. This property is similar to the processor locality of reference exhibited in parallel applications.[6] Spatial locality implies that an object near a recently accessed object will likely be accessed.

If client-based caching occurs on a per-session basis (that is, the cache is cleared at the start of each client session), only temporal locality of reference can be exploited. The results of our client-based caching study suggest that for a single client, the temporal locality of reference is quite limited, especially for remote documents. In particular, we found that even with an infinite cache size, the average *byte hit rate* is limited to 36% and could be as low as 6% for some client traces. (The byte hit rate is a "normalized" hit rate that takes into account the size of the objects that are accessed.) This poor performance could be attributed to the "surfing" behavior of clients, which implies that recently examined documents are rarely revisited.

If client-based caching is on a per-site basis (that is, all clients share a common proxy cache), you would expect improved cache performance as a result of the geographical locality of reference. However, the results of our study suggest that for remote documents, the amount of sharing between clients is limited. In particular, we found that even with an infinite proxy cache size, the average byte hit rate improves only from 36% to 50%.

Table 1. Summary statistics for log data used in this article.

| | CS-WWW.BU.EDU Data set 1 | | CS-WWW.BU.EDU Data set 2 | | WWW.STONES.EDU Data set 3 | |
|---|---|---|---|---|---|---|
| Days | 56 | | 182 | | 110 | |
| URL requests | 172,635 | | 585,739 | | 4,068,432 | |
| Mbytes transferred | 1,447 | | 6,544 | | 112,015 | |
| Average daily transfer in Mbytes | 26 | | 36 | | 1,018 | |
| Files on system | 2,018 | | 2,679 | | N/A | |
| Files accessed (remotely) | 974 | (656) | 1,628 | (1,032) | N/A | (1,151) |
| Size of file system (amount accessed) in Mbytes | 50 | (37) | 62 | (42) | N/A | (402) |
| Unique clients (10+ requests) | 8,123 | | 8,474 | | 60,461 | |

Figure 1, based on the data in our study, illustrates this point. It plots the *cache expansion index*: the rate at which the proxy cache (called *LAN cache* in our study) should be inflated to maintain a constant byte hit rate. The figure indicates that the CEI is proportional to the number of clients ($N$) sharing the proxy cache. For a large number of clients concurrently using the cache, the CEI is linearly related to $N$ and does not seem to level off.

Figure 1 does not imply that the relationship between the CEI and $N$ will continue to be linear for even larger values of $N$; it only implies that for the levels of concurrency likely to be aggregated at a single site through a proxy cache, the relationship is linear. To get a higher level of concurrency, we need to think about an economy of scale far beyond what a proxy cache at, say, an organization can provide.

Therefore, temporal and geographical locality of reference for WWW access patterns are not strong enough to result in an effective caching strategy at a single client or site. However, we can exploit temporal and geographical locality of reference on a much larger scale (for example, thousands of clients), as I'll explain.

## The premise of dissemination

To quantify the available locality of reference that could be exploited on the WWW, we collected extensive server traces from the HTTP server of Boston University's Computer Science Department (*http://cs-www.bu.edu*) and from the HTTP server of the Rolling Stones Web site (*http://www.stones.com/*). Table 1 summarizes these traces. Unless I state otherwise, this data drove our model validation and trace simulations.

Carlos Cunha, Mark Crovell, and I have documented the highly uneven popularity of various Web documents.[3] This study confirmed the applicability of Zipf's law[7,8] to Web documents. Zipf's law originally applied to the relationship between a word's popularity in terms of rank and its frequency of use (it has subsequently been applied to other examples of popularity in the social sciences.). It states that if you rank the popularity of words in a given text ($\rho$) by their frequency of use ($P$), then $P \sim 1/\rho$. This distribution is a parameterless hyperbolic distribution—that is, $\rho$ is raised to exactly –1, so that the $n$th-most-popular document is exactly twice as likely to be accessed as the $2n$th-most-popular document.

Our data shows that Zipf's law applies quite strongly to Web documents serviced by Web servers. Figure 2a demonstrates this for all 2,018 documents accessed in data set 1 (see Table 1). The figure shows a log-log plot of the total number of references ($y$ axis) to each document as a function of the document's rank in overall popularity ($x$ axis). The tightness of the fit to a straight line is strong ($R^2 = 0.99$), as is the line's slope: –0.95 (shown in the figure). So, the exponent relating popularity to rank for Web documents is very nearly –1, as Zipf's law predicted. Out of some 2000+ files available through the WWW server, the most popular 256-Kbyte block of documents (that is, 0.5% of all available

## Related work

James Gwertzman and Margo Seltzer's research[1] is closest to ours. In particular, they propose *geographical push-caching*. This method lets servers decide when and where to cache information, based on geographical information: the distance in miles between servers and clients. Their work assumes that the physical distance between two Internet nodes correlates with the number of hops between these nodes. Abdelsalam Heddaya and Sulaiman Mirdad attempt to achieve load-balancing among servers through automatic document dissemination by adding a caching component to routing protocols.[2] Such an approach requires the implementation of new transport protocols and thus does not apply to current TCP/IP-based networks.

*References*


1. J.S. Gwertzman and M. Seltzer, "The Case for Geographical Push-Caching," *Proc. Fifth Workshop on Hot Topics in Operating Systems (HotOS-V)*, IEEE Computer Society Press, Los Alamitos, Calif., 1995, pp. 51–55.

2. A. Heddaya and S. Mirdad, "Wave: Wide-Area Virtual Environment for Distributing Published Documents," *Proc. SIGCOMM'95: Workshop on Middleware*, Cambridge, Mass., 1995; *http://www.cs.bu.edu/faculty/heddaya/Papers-NonTR/middleware95.html*.

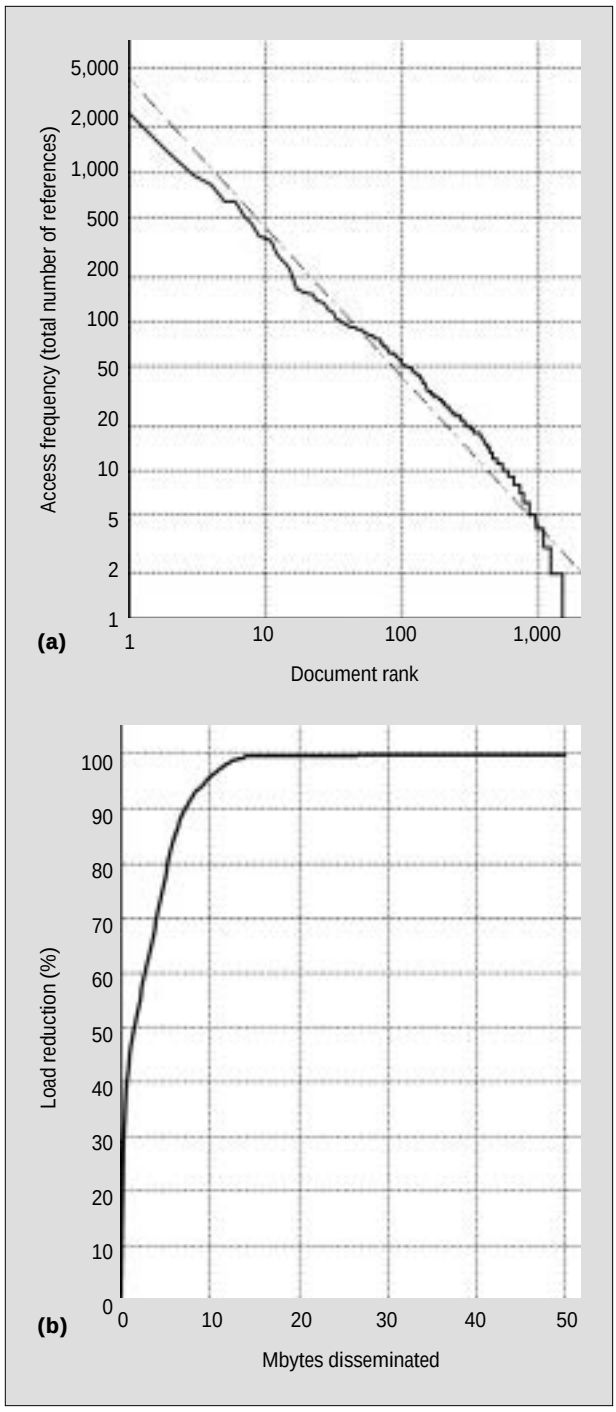


Figure 2. The popularity of documents versus their rank (a), and the potential load reduction from dissemination (b).

documents) accounted for 69% of all requests. Only 10% of all blocks accounted for 91% of all requests!

This observation leads to the following question: How much bandwidth could be saved if requests for popular documents from outside the LAN were handled at an earlier stage (for example, using a proxy at the "edge" of the organization)? Figure 2b shows the percentage of the server load (measured in total bytes serviced) that would be saved if some other server serviced various block sizes, starting with the most popular blocks and progressing to less popular blocks.

We corroborated our observations by analyzing the HTTP logs of the Rolling Stones server from November 1, 1994, to February 19, 1995. As Table 1 shows, this server—unlike the cs-www.bu.edu HTTP server—is intended to serve remote clients exclusively. It is a very popular server, servicing more than 1 Gbyte of multimedia information per day to tens of thousands of (distinct) clients. (During the analysis period, the server handled 1,009,146,921 bytes per day, and 60,461 clients retrieved at least 10 files.)

Figure 3a shows the access frequency for all documents that were serviced at least once. Figure 3b shows the percentage of the remote bandwidth that would be saved if some other server handled various block sizes of decreasing popularity. Of the 400 MBytes of information accessed at least once (the total number of bytes available from that server is much larger than 400 MBytes), only 21 MBytes (5.25%) were responsible for 85% of the traffic.

A closer look at the logs of *http://cs-www.bu.edu*, which is a typical server that caters primarily to local clients, reveals three classes of documents. Figure 4 shows the ratio of remote-to-local (and local-to-remote) accesses for each of the 974 documents accessed at least once. Of these documents, 99 had an access ratio larger than 85%—these are *remotely popular documents*; 510 documents had an access ratio smaller than 15%—these are *locally popular documents*; 365 documents had an access ratio between 85% and 15%—these are *globally popular documents*. Servers could easily classify documents into these three classes (based on temporal and geographical locality of reference), to decide which documents to disseminate and where to disseminate them.

An important factor that could affect our dissemination protocol's performance is the rate at which popular documents are updated. The more frequently these documents are updated, the more frequently the home server must refresh the replicas at the proxies. In a related study, we monitored the *date of last update* of remotely, locally, and globally popular documents for a six-month (26-week) period.[9] The period started with the 56-day period of Table 1, and continued for another 126 days. Both remotely popular and globally popular documents were updated very infrequently (less than 0.5% update probability per document per day). Locally popular documents were updated more frequently (about 2% update probability per document per day). (We counted multiple updates to a document in one day

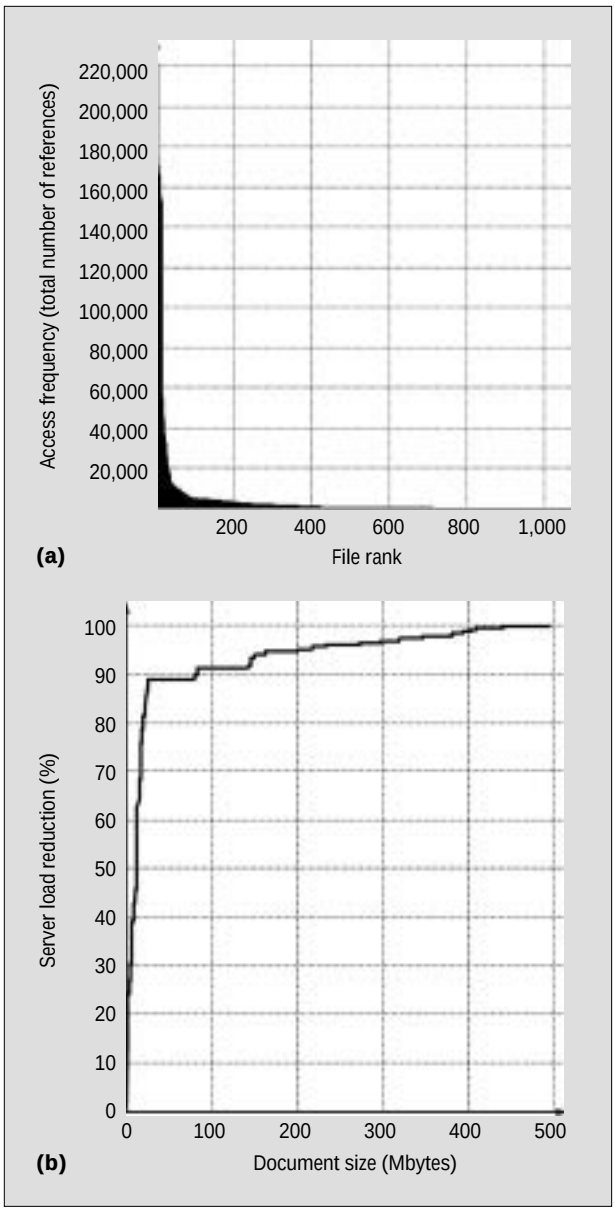


Figure 3. Access frequency (a), and projected bandwidth reduction (b), for the www.stones.com server.

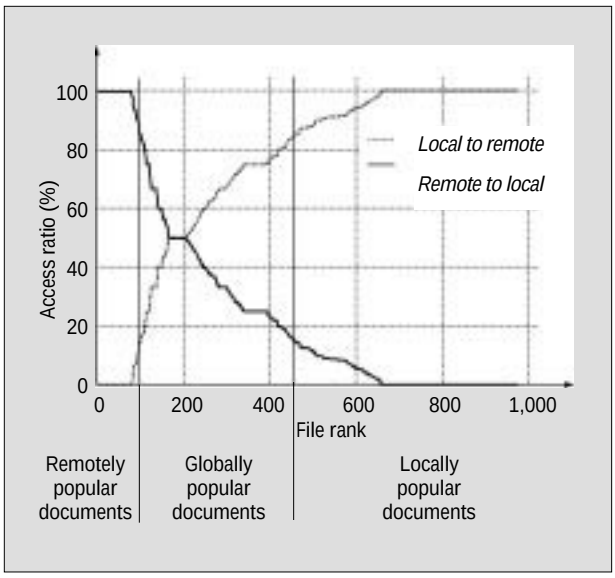


Figure 4. Local versus remote popularity of documents.

as one update.) This result suggests that the documents most likely to be disseminated are the ones least likely to change. James Gwertzman and Margo Seltzer have further investigated the significance of these findings, in the context of WWW cache consistency.[10]

## System model and analysis

In our model, *home servers* (producers) disseminate information to *service proxies* (agents) closer to *clients* (consumers). We assume a many-to-many mapping between home servers and service proxies. A service proxy and the set of home servers it represents form a *cluster*. We model the WWW as a hierarchy of such clusters.

Our notion of a service proxy is similar to that of a client proxy, except that the service proxy acts on behalf of a cluster of servers rather than a cluster of clients. In practice, we envision service proxies to be information outlets that are available throughout the Internet, and whose bandwidth could be, for example, rented. Alternately, service proxies could be public engines, part of a national computer information infrastructure, similar to the NSF backbone. For the remainder of this article, *proxy* means a service proxy.

Our model does not limit the number of proxies that could be used to "front end" a particular server. Each server in the system might belong to a number of clusters, and thus might have a number of proxies acting on its behalf, thereby disseminating its documents along multiple routes (or toward various subnetworks). A server can use (through bidding, for example) a subset of these proxies to disseminate its data to clients.

For a given home server, we view the WWW clientele as a tree rooted at the server. The tree's leaves are the clients, and the internal nodes are the potential proxies. In "Performance evaluation," I'll describe an efficient technique for building such a tree. Furthermore, by analyzing the access patterns of clients (as recorded in the server logs), we can optimally locate the set of nodes to use as proxies for that home server. (We did this for the *http://cs-www.bu.edu* home server, whose 26-week clientele tree consisted of more than 34,000 nodes.)

### A resource allocation strategy at service proxies

Let $C = S_0, S_1, S_2, \ldots, S_n$ denote all the servers in a cluster, where server $S_0$ is distinguished as the proxy of cluster $C$. Let $R_i$ denote the total number of bytes per

unit time (say, one day) serviced by $S_i$ in $C$ to clients outside that cluster. Furthermore, let $H_i(b)$ denote the probability that a request for a document on $S_i$ can be serviced at proxy $S_0$ as a result of disseminating the most popular $b$ bytes from $S_i$ to $S_0$. Finally, let $B_i$ denote the number of bytes that proxy $S_0$ duplicates from server $S_i$, and let $B_0$ denote the total storage space available at proxy $S_0$ (that is, $B_0 = B_1 + B_2 + \ldots + B_n$). By intercepting requests from outside the cluster, $S_0$ should be able to service a fraction, $\alpha_C$, of these requests.

$$\alpha_C = \frac{\sum_{i=1}^{n} R_i \times H_i(B_i)}{\sum_{i=1}^{n} R_i} \tag{1}$$

The objective of $S_0$ is to allocate storage spaces $B_1$, $B_2$, …, $B_n$ to maximize the value of $\alpha_C$, subject to the constraint that $B_0 \le B_1 + B_2 + \ldots + B_n$. This is a constrained-maximum problem, which can be solved using the Lagrange multiplier theorem. Thus, the maximum for $\alpha_C$ occurs when for all $j = 1, 2, \ldots, n$,

$$\frac{\delta}{\delta B_j}\alpha_C = \mathrm{k}, \text{ for a constant k}$$

$$\frac{\delta}{\delta B_j}\left(\frac{\sum_{i=1}^{n} R_i \times H_i(B_i)}{\sum_{i=1}^{n} R_i}\right) = \mathrm{k}$$

$$h_j(B_j) = \mathrm{k} \cdot \frac{\sum_{i=1}^{n} R_i}{R_j} \tag{2}$$

where $h_j(B_j)$ denotes the probability density function corresponding to $H_j(B_j)$. Equation 2 uses the value of k (the Lagrange multiplier) to satisfy $B_0 = B_1 + B_2 + \ldots + B_n$.

Our desire to make our protocol useful restricts the type of assumptions we could make. Thus, in our protocol, we have avoided using any parameters that could not be readily estimated from available logs of network protocols (for example, HTTP and FTP). However, future work along the same lines could use other information to better tune the system. For example, if information about the communication cost between servers, proxies, and clients is available, our protocol could be easily adapted to weigh such knowledge into our resource-allocation methodology.

## Analysis under an exponential popularity model

We'll use an exponential model to approximate the function $H_i(b)$. We assume that for $i = 1, 2, \ldots, n$, $H_i(b) = 1 - e^{-\lambda_i \cdot b}$, where $\lambda_i$ is the distribution's constant. The probability density function to $H_i(b)$ is $h_i(b)$, where

$$h_i(b) = \frac{\delta}{\delta b} H_i(b) = \lambda_i e^{-\lambda_i \cdot b} \tag{3}$$

Given a server $S_j$, where $1 \le j \le n$, we substitute for $h_j(b)$ in Equation 2 to get a value for $B_j$.

$$B_j = \log\left(\frac{\lambda_j}{\mathrm{k}} \frac{R_j}{\sum_{i=1}^{n} R_i}\right)^{\frac{1}{\lambda_j}} \tag{4}$$

Equation 4 specifies a set of $n$ equations to ration the total buffering space $B_0$ available at $S_0$ among the servers $S_i$, for $i = 1, 2, \ldots, n$. To do so, we must find the value of the constant k. We do this by observing the requirement that $B_0 \le B_1 + B_2 + \ldots + B_n$, which results in this expression for k:

$$\mathrm{k} = \frac{1}{\sum_{i=1}^{n} R_i}\left(\frac{\prod_{i=1}^{n} (\lambda_i R_i)^{\frac{1}{\lambda_i}}}{e^{B_0}}\right)^{\frac{1}{\sum_{i=1}^{n} \frac{1}{\lambda_i}}} \tag{5}$$

Substituting for k from Equation 5 into Equation 4, we get the optimum storage capacity to allocate on $S_0$ for a particular server $S_j$, where $1 \le j \le n$.

These calculations require that $R_i$ and $\lambda_i$ be estimated, for $i = 1, 2, \ldots, n$. This can be easily and efficiently computed from the server logs. Actually, Figure 2 was produced by a program that computed these parameters for cs-www.bu.edu. Moreover, our measurements suggest that these parameters are quite static, in that they change only slightly over time. Hence, the calculation of $R_i$ and $\lambda_i$, as well as the allocation of storage space on $S_0$ for servers $S_i$, for $i = 1, 2, \ldots, n$, need not be done frequently. They could occur either off line or periodically (for example, weekly).

## Special cases

To help explain our demand-based document-dissemination protocol, I'll present three special cases.

### *Equally effective duplication*

Let $\lambda_i = \lambda$ for $i = 1, 2, \ldots, n$. That is, let's assume that the

reduction in bandwidth that results from duplicating some number of bytes from a particular server $S_j$ is equal to the reduction in bandwidth that results from duplicating the same number of bytes from any other server $S_i$ for $i = 1, 2, \ldots, n$. Substituting in Equation 5 and then Equation 4, we get

$$B_j = \frac{B_0}{n} + \frac{1}{\lambda}\log\frac{R_j}{\sqrt[n]{\prod_{i=1}^{n} R_i}} \tag{6}$$

Equation 6 suggests that popular servers are allocated extra storage capacity on the proxy. This extra storage depends on two factors: $1/\lambda$, which is a measure of duplication effectiveness, and

$$\log\left(R_j \Big/ \sqrt[n]{\prod_{i=1}^{n} R_i}\right)$$

which reflects a server's popularity relative to the geometric mean of all servers in the system. This dual dependency on duplication effectiveness and relative popularity gives us a handle on how to extend our results for arbitrary distributions of $H_i(b)$. In particular, if the skewness of $H_i(b)$ can be measured for a particular server (by analyzing its logs, as suggested earlier), this measure can be used instead of $1/\lambda$.

### *Equally popular servers*

Let $R_i = R$ for $i = 1, 2, \ldots, n$. That is, let's assume that all servers in the system are equally popular. Substituting in Equation 5 and then in Equation 4, we get

$$B_j = \frac{1}{\sum_{i=1}^{n}\frac{\lambda_j}{\lambda_i}}\left(B_0 + \sum_{i=1}^{n}\frac{1}{\lambda_i}\log\frac{\lambda_j}{\lambda_i}\right) \tag{7}$$

Equation 7 suggests that servers whose data are accessed more uniformly (that is, servers with a smaller $\lambda$) should be allotted more storage capacity on the proxy as long as the proxy's total capacity is large enough (that is, $B_0 \gg n/\lambda_i$). However, if the server's capacity is not big enough, servers with intermediate values for $\lambda$ should be favored. Figure 5 shows the optimal storage capacity to allocate to server $S_j$ for various values of $\lambda_j$, assuming that all other $n - 1$ servers have equal $\lambda_i$ and that $B_0 = 1/\lambda_i$ (tight) or $B_0 = 10\ 1/\lambda_i$ (lax), for $1 \le i \le n$ and $i \ne j$.

### *Symmetric clusters*

To appreciate the effectiveness of demand-based document dissemination, let's consider a symmetric cluster, where all servers have identical values for $R_i$ and $\lambda_i$. From Equation 5 and Equation 4, we get

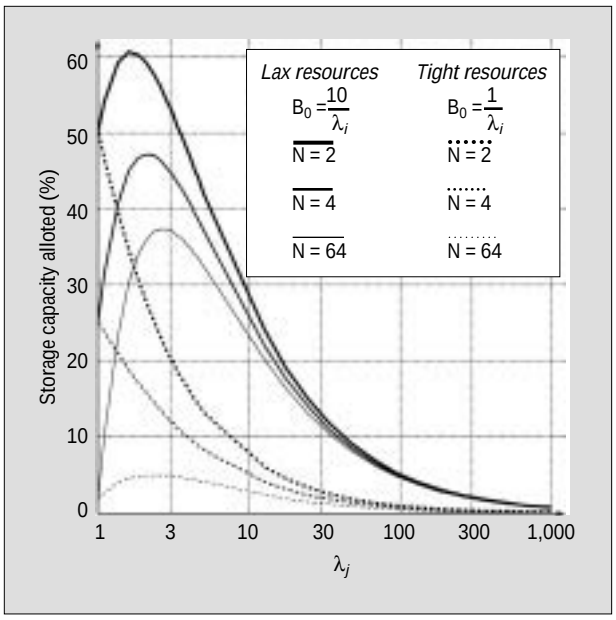


Figure 5. Storage allocation for $R_i = R$ and $B = 1/\lambda_i$. $\lambda_j$, the constant of the $H_j(b)$ distribution introduced in Equation 3, measures how skewed $H_j(b)$ is.

$$B_j = \log\left(\frac{\lambda}{\frac{\lambda}{n}\cdot e^{-\frac{\lambda}{n}B_0}}\frac{R}{\sum_{i=1}^{n} R}\right)^{\frac{1}{\lambda}} = \frac{B_0}{n} \tag{8}$$

As expected, Equation 8 equally allocates storage on $S_0$ for all the servers in the cluster. By substituting the value of $B_j$ into Equation 1, we get

$$\alpha_C = \frac{\sum_{i=1}^{n} R \times H\left(\frac{B_0}{n}\right)}{\sum_{i=1}^{n} R} = 1 - e^{-\lambda\frac{B_0}{n}} \tag{9}$$

Equation 9 could be used to estimate the storage requirements on the proxy as a function $\alpha$:

$$B_0 = \frac{n}{\lambda}\log\frac{1}{\alpha_C} \tag{10}$$

Assume, for example, that the *http://cs-www.bu.edu* server is one of 10 servers whose most popular data are duplicated on a proxy. Equation 10 suggests that to reduce the remote bandwidth by 90% on all servers, the proxy must secure 36 Mbytes to be divided equally among all servers. This assumes a value of $\lambda = 6.247 \times 10^{-7}$, which we estimated from the HTTP demon logs on the *cs-www.bu.edu* server. With a storage capacity of 500

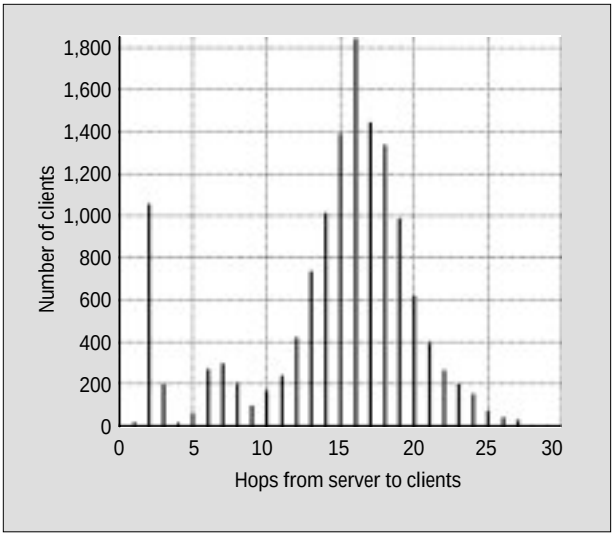


Figure 6. How far away are clients?

Mbytes, a proxy could shield 100 servers from as much as 96% of their remote bandwidth.

These numbers raise a legitimate question: If one proxy serves 96% of all remote accesses to 100 servers (or even 90% of all accesses to 10 servers), won't that proxy be a performance bottleneck? Yes, unless the process of disseminating popular information continues for another level, and so on. If that is not possible, then another solution would be for the proxy to dynamically adjust the level of shielding it provides for its constituent servers. In other words, when the proxy becomes overloaded, $B_0$ decreases, thus forcing more of the requests back to the servers.

## Performance evaluation

To evaluate our dissemination protocol, we had to devise a realistic clustering of the Internet to reflect our dissemination model. Several criteria could determine this clustering. One criteria could be geographical information (for example, distance in miles between clients).[11] Another could be the institutional or network boundaries (for example, one service proxy per institution or network). We based the clustering on the structure of the routes between a server and its clients. Such a clustering (based on the distance between the server and client measured in *actual route hops*) lets us quantify the savings achievable through dissemination, because it lets us measure the bandwidth saved in *bytes* × *hops* units.

Using the `record route` option of TCP/IP, it is possible to build a complete server-proxy-client tree. We built such a tree for all *traceable clients* of *http://cs-www.bu.edu*. A traceable client is a client for which `traceroute()` succeeded in identifying a route that remained consistent over several trace-route experiments. Almost 90% of all bytes transferred from *http://cs-www.bu.edu* transferred to traceable clients. For

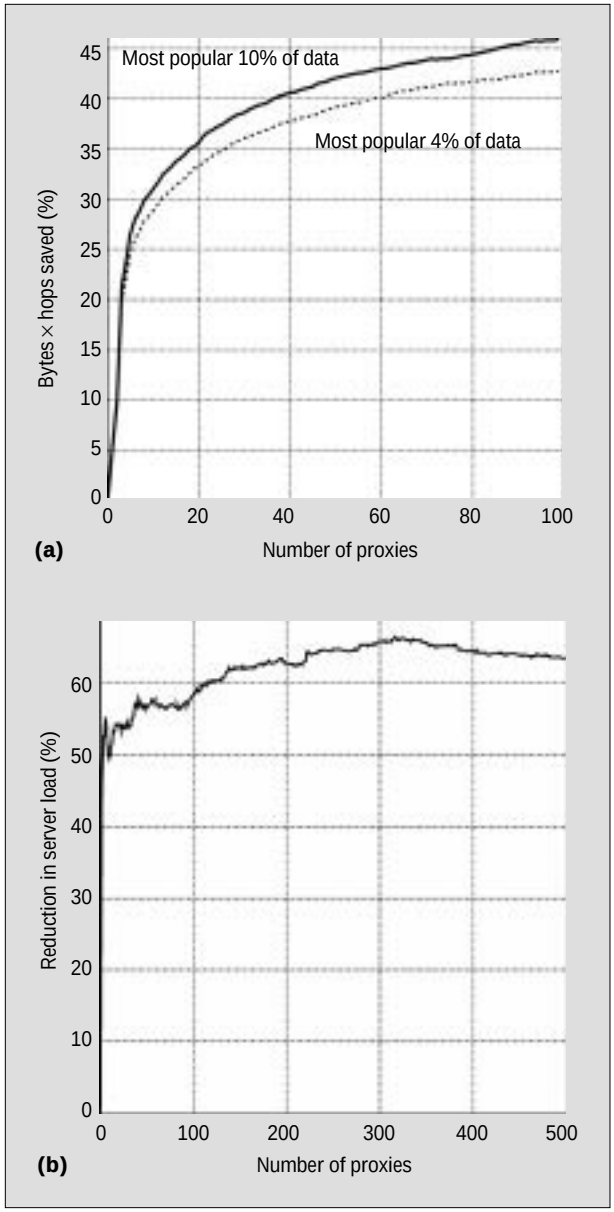


Figure 7. The results of dissemination: reductions in (a) bandwidth and (b) home server load.

data set 2 of Table 1, the constructed tree consisted of more than 18,000 nodes.

Figure 6 shows a histogram of the number of clients versus their distance in hops from the *http://cs-www.bu.edu* server. This histogram shows three distinct client populations. The first population is within two to three hops and represents on-campus clients at Boston University. The second is within four to nine hops and represents clients on the New England Academic and Research network (NEARnet). The third is greater than nine hops away and represents WAN clients. For servers where WAN clients generate most of the demand, Figure 6 suggests that popular documents could be disseminated as much as eight to nine hops from the server. Such a dissemination would result in large savings. Specifically,

replicating the most popular 25 Mbytes from *http://www.stones.com* on proxies eight to nine hops closer to clients would yield a whopping daily network bandwidth savings of more than 8 Gbytes × hops.

We simulated our dissemination strategy based on the structure of this server-to-client routing tree. (Analysis of *http://cs-www.bu.edu* logs suggests that the tree's shape—especially internal nodes—and the load distribution are quite static over time.) The 26-week traces obtained from *http://cs-www.bu.edu* (data set 2 in Table 1) drove our simulations.

Our simulations disseminated replicas of the most popular files on *http://cs-www.bu.edu* every week, down to proxies closer to the clients. The location of such proxies depended on the demand during the previous month from the various parts of the tree. (Our protocol proved to be quite robust with respect to the frequency of dissemination and the history length used to establish file popularity profiles. Because of space limitations, I haven't included simulation results to support this.)

Our general dissemination protocol assumes a hierarchy of proxies. Our implementation of this idea was simpler; we flattened the hierarchy by having the home server use the access patterns of all its clients to compute the ultimate placement for the replicas. In other words, our simulations did not consider multilevel proxies, as would be possible under the general dissemination model.

We assumed that any internal node is available as a service proxy. In a real system, this assumption might not be valid because internal nodes are routers, unlikely to be available as service proxies. We envision that in practice, the set of proxies available for rent by a particular server could be matched up to the optimum places determined by our protocol.

The simulations required clients to always request service from the home server. If the requested documents are available at proxies closer to the client, the home server forwards the request to the proxy that is closest to the client. This forwarding mechanism is supported by the HTTP protocol and requires much less overhead than the more general hierarchical name-resolution strategy.[12]

The simulations also assumed that the cost of propagating updates to proxies can be ignored. To validate this assumption, consider the ratio of a popular document's number of changes per unit time to its number of accesses per unit time. As I noted earlier, popular documents are updated least. So, that ratio will be very small

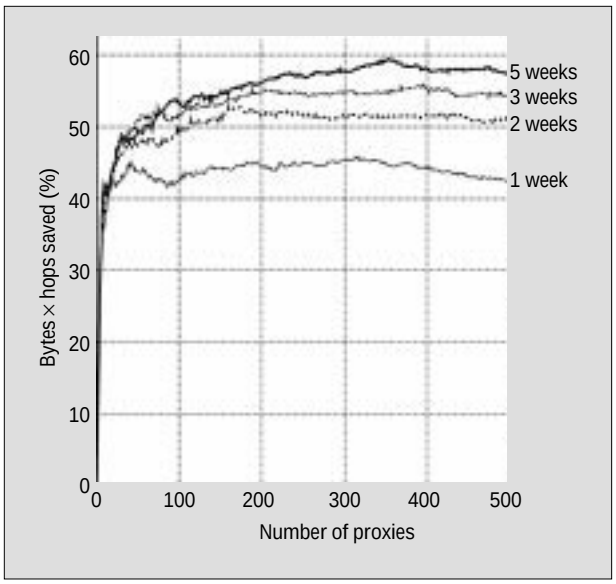


Figure 8. The algorithm's sensitivity to the length of the history that is used to compute the popularity profile.

(we measured it at less than 0.001 for popular documents). Thus, the inaccuracy of our simulations (by not taking into account the cost of propagating updates) is insignificant.

Figure 7a shows the reduction in bandwidth (measured in bytes × hops) that is achievable by disseminating different amounts of the most popular data. The figure shows two curves. In the first, the system disseminates the most popular 10% of the data; whereas in the second, it disseminates the most popular 4% of the data. A larger number of proxies results in a deeper dissemination and, thus, larger bandwidth savings. Figure 7b shows the reduction in home server load (measured in bytes) achievable by disseminating 4% of the most popular data.

Figure 7a shows that the dissemination of a very small amount of data causes most of the bandwidth savings. For example, increasing the level of dissemination by 150% (from 2 Mbytes per service proxy to 5 Mbytes per service proxy) results in a meager 6% additional bandwidth savings. Also, most of the saved bandwidth results from using a small number of service proxies. For example, tripling the number of service proxies from 20 to 60 saves only 21% additional bandwidth. Finally, Figure 7 shows that the payoff per replica decreases as the number of replicas increases.

Figure 8 shows our algorithm's sensitivity to changes in the popularity profile for documents on the home server. In particular, it shows the performance results for four experiments, in which the dissemination occurred once a week, using the logs of the previous $n$ weeks to compute the popularity profile, for $n = 1, 2, 3,$ and 5. For these experiments, the system disseminated the 10% most popular data (computed over the $n$-week period). The figure shows that the longer the history

that is used to compute the popularity profile, the better the performance, especially when the number of proxies is large.

Our simulation disseminated the same data to all proxies. Better results are attainable if the dissemination strategy takes greater advantage of the geographic locality of reference (by disseminating different data to different proxies based on the access patterns of clients served by each proxy).

These experiments demonstrate the potential savings from dissemination. More experiments are needed to generalize these findings by considering traces from a larger set of servers. Also, more experiments are needed to study the effects of dissemination from competing service proxies. Finally, the Web is evolving in terms of types of information and how this information is presented (for example, the increases in Common Gate Interface queries, Java applets, and scripts). This evolution might require the development of more elaborate dissemination protocols.

The basic premise of our work is that servers are in a unique position to decide which documents are worth replicating through dissemination, how many replicas should be disseminated, and where to place these replicas. This, however, does not imply that clients and their proxies do not (or should not) play a role in improving the performance of information retrieval and in alleviating communication bottlenecks.

For example, our work is not a substitute for client-based caching; rather, it is a complement. Client-based caching's primary purpose is to reduce the latency of information retrieval, whereas information dissemination's primary purpose is to reduce traffic and balance the load among servers. Of course, these purposes are not completely independent—a protocol whose primary purpose is to reduce traffic and balance load is likely to improve the latency of information retrieval. Nevertheless, the highest reduction in latency will likely result from a protocol whose primary purpose is to do just that (client-based caching in this case).

Another example of how our work complements client-based protocols concerns dynamic server selection, introduced by Mark Crovella and Robert Carter.[13] The primary purpose of a client-based dynamic server-selection protocol is to determine on the spot which one of several servers to use to retrieve a document replicated on all the servers. This selection is based on dynamic measurements that attempt to locate the server with the least congested route. Such a technique could complement dissemination by allowing servers to communicate with clients the location of all (or some) replicas that are deemed close to the client. The client-based server-selection protocol would have the final decision as to which replica to retrieve based on the dynamic conditions of the network.

The protocol I've presented exemplifies the potential of client-access patterns in engineering scaleable protocols and services in large-scale distributed information systems (such as the WWW). Ocean is developing prototypes to test and evaluate similar protocols.


## Acknowledgments

I would like to thank all the members of the *Oceans* group (*http://cs-www.bu.edu/groups/oceans*) for their feedback and many discussions on this work. I particularly thank Carlos Cunha for his trace-collection efforts and for helping with the trace-driven simulation of the dissemination protocol presented in this article. This work has been partially supported by NSF Grant CCR-9308344.


## References


1. M. Foster and R. Jump, NSF Solicitation 94-75, *STIS Database*, Nat'l Science Foundation, Arlington, Va., 1994.

2. A. Bestavros et al., "Application Level Document Caching in the Internet," *Proc. Second Int'l Workshop on Services in Distributed and Networked Environments (SDNE '95)*, IEEE Computer Society Press, Los Alamitos, Calif., 1995, pp. 166–173.

3. C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW Client-Based Traces," Tech. Report TR-95-010, Computer Science Dept., Boston Univ., Boston, 1995.

4. S. Glassman, "A Caching Relay for the World Wide Web," *Proc. First Int'l Conf. WWW*, North-Holland, Amsterdam, 1994, pp. 69–76.

5. M. Abrams et al., "Caching Proxies: Limitations and Potentials," *Proc. Fourth Int'l Conf. WWW*, O'Reilly & Associates, Sebastopol, Calif., 1995, pp. 312–319.

6. S.J. Eggers and R.H. Katz, "A Characterisation of Sharing in Parallel Programs and Its Application to Coherence Protocol Evaluation," *Proc. 15th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, 1988, pp. 373–382.

7. B.B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freedman and Co., New York, 1983.

8. G.K. Zipf, *Human Behavior and the Principle of Least-Effort*, Addison-Wesley, Reading, Mass., 1949.

9. A. Bestavros, "Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information Systems," *Proc. ICDE '96: 12th Int'l Conf. Data Eng.*, IEEE CS Press, 1996, pp. 180–187.

10. J. Gwertzman and M. Seltzer, "World Wide Web Cache Consistency," *Proc. 1996 USENIX Tech. Conf.*, Usenix Assoc., Berkeley, Calif., 1996, pp. 141–152.

11. J.S. Gwertzman and M. Seltzer, "The Case for Geographical Push-Caching," *Proc. Fifth Workshop on Hot Topics in Operating Systems (HotOS-V)*, IEEE CS Press, 1995, pp. 51–55.

12. P. Danzig, R. Hall, and M. Schwartz, "A Case for Caching File Objects inside Internetworks," Tech. Report CU-CS-642-93, Computer Science Dept., Univ. of Colorado, Boulder, Colo., 1993.

13. M. Crovella and R. Carter, "Dynamic Server Selection in the Internet," *Proc. Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS '95)*, IEEE Communication Soc., New York, 1995, pp. 158–162.

**Azer Bestavros** is on the faculty of Boston University's Computer Science Department. His research mainly concerns real-time systems and distributed systems, and is partially funded by research grants from the National Science Foundation, the US Army Research Office, and GTE. His work on large-scale distributed information systems is conducted with Boston University's Oceans research group, which he cofounded in 1994. He is the editor of the *Newsletter of the IEEE Computer Society Technical Committee on Real-Time Systems* and maintains its archives at Boston University. He obtained his SM and PhD from Harvard University in 1988 and 1992. He can be reached at the Computer Science Dept., Boston Univ., MA 02215; best@cs.bu.edu.