In Proceedings of the ACM/IASTED/ISMM Conference on Distributed Multimedia Systems and Applications, Stanford, CA, Aug 1995.

Demand-based Data Dissemination in Distributed Multimedia Systems*

Azer Bestavros
(best@cs.bu.edu)

Computer Science Department
Boston University
Boston, MA 02215

Abstract

We offer a global solution that replicates multimedia information on a supply/demand basis by disseminating it from its producers to servers that are closer to its consumers. The level of dissemination depends on the relative popularity of documents and the expected reduction in traffic that results from their replication. We used extensive HTTP logs from academic and commercial servers to validate an analytical model of server popularity and file access profiles. Using that model we show that our protocol has the potential of reducing network traffic considerably, while keeping servers load-balanced.

Keywords: Caching; replication; resource allocation; load balancing; WWW proxies; performance evaluation.

1 Introduction and Motivation

Current protocols for accessing distributed information are inefficient, wasteful of bandwidth, and exhibit a large degree of performance unpredictability. Furthermore, the growing disparity between the volume of data that becomes available and the retrieval capacity of existing networks is a critical issue in the design and use of future distributed information systems. The best "living" proof of the seriousness of this problem is the fate of many multimedia Internet servers: they are unreacheable as soon as they become popular.

To tackle this problem, we propose a novel protocol for improving the availability and responsiveness of distributed multimedia information systems. We use the World Wide Web (WWW) 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, the WWW is fully deployed in thousands of institutions worldwide, which gives us an unparalleled opportunity to apply our findings to an already-existing real-world application. The basic idea of our protocol[1] is to *off-load* popular servers by duplicating (on other servers) only a small percentage of the data that such servers provide. The extent of this duplication (how much, where, and on how many sites) depends on two factors: the popularity of the server and the expected reduction in traffic if dissemination is done in a particular direction. In other words, our protocol provides a mechanism whereby "popular" data is disseminated automatically and dynamically towards consumers—the more popular the data, the closer it gets to the clients.

Early research on caching and replication to improve the availability and performance of scalable distributed file systems is described in [9]. Recently, there have been some attempts at extending caching and replication to distributed information systems (*e.g.* FTP and HTTP) [6, 1, 11]. Multi-level caching was studied in [10]. In [4, 5], a dynamic hierarchical file system, which supports demand-driven replication is proposed, whereby clients are allowed to service requests issued by other clients from local disk cache. The proposed research work of Gwertzman and Seltzer sketched in [8] is the closest to ours. They propose the implementation of *geographical push-caching*, which allows servers to decide when and where to cache information based on geographical information such as the distance in miles between servers and clients.

Figure 1 shows the frequency of remote access of individual 256KB multimedia object (or document) blocks

*This work has been partially supported by NSF CCR-9308344

Proceedings of IASTED/ISMM/ACM International Conference on Distributed Multimedia Systems and Applications, August 7-9, Stanford, CA, USA.

[1] Due to space limitations, we describe our protocol only informally. For more details, we refer the reader to our work in [3].

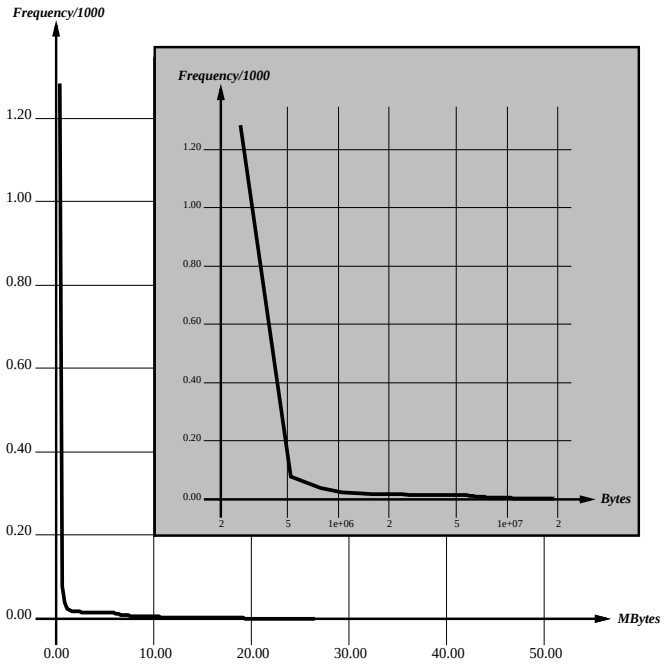

Figure 1: Popularity of blocks on `cs-www.bu.edu`

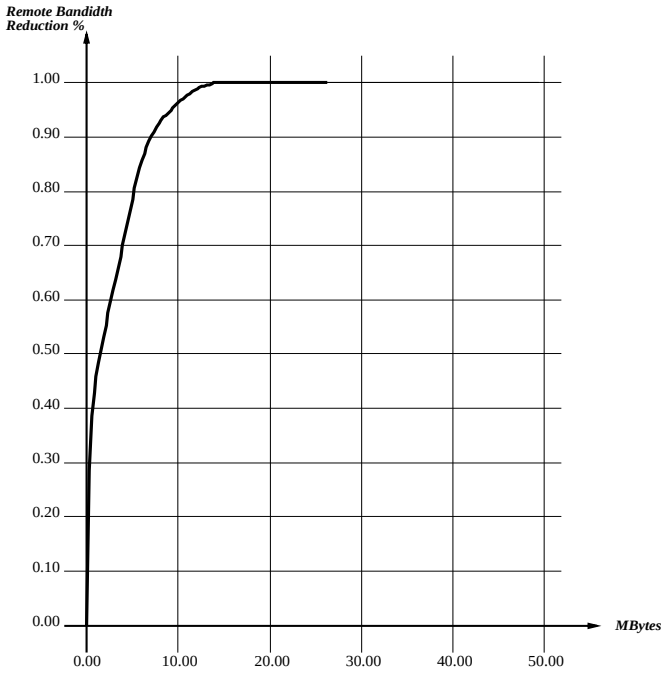

Figure 2: Bandwidth savings against size of proxy

available through `http://cs-www.bu.edu`. The horizontal axis depicts these blocks in a decreasing *remote popularity*. Out of some 2000+ files available through the WWW server only 656 files were remotely accessed at least once. The size of these 656 files totalled some 36.5 MBytes, which represents 73% of the 50+MBytes available through the server. Alone, the most popular 256KB block of documents (that is 0.5% of all available documents) accounted for 69% of all requests. Only 10% of all blocks accounted for 91% of all requests! These observations lead to the following question: How much bandwidth could be saved if requests for *popular* documents from outside the LAN are handled at an earlier stage (*e.g.* using a proxy at the "edge" of the organization)? Figure 2 shows the percentage of the remote bandwidth that would be saved if various block sizes of decreasing popularity are serviced at an earlier stage.

The above observations have been corroborated by analyzing the HTTP logs of the Rolling Stones server `http://www.stones.com/` from November 1, 1994 to February 19, 1995. Unlike the `cs-www.bu.edu` HTTP, this server is intended to serve exclusively remote clients. It is a very popular server with more than 1 GigaByte of multimedia information per day (exactly 1,009,146,921 Bytes/day) serviced to tens of thousands (distinct) clients (namely 60,461 clients retrieved at least 10 files during the duration of the analysis). Of the 400 MBytes of information accessed at least once[2] during the analysis period, only 21 MBytes (5.25%) were responsible for 85% of the traffic.

[2] Notice that the total number of bytes *available* from that server is much larger than 400 MBytes.

2 System Model and Analysis

We model the WWW (Internet) as a hierarchy of clusters. A cluster at any particular level of this hierarchy consists of a number of servers. One of these servers acts as a front-end service *proxy* for the cluster and is, thus, a server at the next level up in the hierarchy. Let $\mathcal{C} = \mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ denote all the servers in a particular cluster, where $\mathcal{S}_0$ is distinguished as the proxy of $\mathcal{C}$. In our model, a cluster corresponds to an institution or an organization. For example, we may model all the WWW servers at Boston University as servers within a cluster, with a particular machine (say `www.bu.edu`) acting as a proxy for the whole institution. In the meantime, one of the servers in the Boston University cluster (say `cs-www.bu.edu`) may itself be a proxy for another cluster of servers (say the various LANs within the CS department). The correspondence between clusters and organizations is only for the purpose of presentation. In practice, we envision proxies to be commercial engines, whose bandwidth could be *"rented"*. Alternately, proxies could be public engines (part of a national computer information infrastructure, similar to the NSF backbone. Our model does not limit the number of proxies that could be used to "front-end" a particular servers. In particular, a server may exist in multiple clusters, and its data may end-up being disseminated along multiple routes.

Let R_i denote the total number of bytes per unit time (say one day) serviced by server $\mathcal{S}_i$ in a cluster $\mathcal{C}$ to clients outside that cluster. Furthermore, let $H_i(b)$ denote the probability that a request for a document on $\mathcal{S}_i$ will be intercepted by proxy $\mathcal{S}_0$ by duplicating the

most popular b bytes of the documents stored on $\mathcal{S}_i$. An example of this probability function is shown in figure 2. Finally, let B_i denote the number of bytes that proxy $\mathcal{S}_0$ duplicates from server $\mathcal{S}_i$ and let B_0 denote the total storage space available at proxy $\mathcal{S}_0$ (*i.e.* $B_0 = B_1 + B_2 + \ldots + B_n$). By intercepting requests from outside the cluster, we may expect $\mathcal{S}_0$ to be able to service a fraction of these requests. Let $\alpha_{\mathcal{C}}$ be that fraction.

$$\alpha_{\mathcal{C}} = \frac{\sum_{i=1}^{n} R_i \times H_i(B_i)}{\sum_{i=1}^{n} R_i} \quad (1)$$

The objective of $\mathcal{S}_0$ is to allocate storage spaces $B_1, B_2, \ldots, B_n$ so as to maximize the value of $\alpha_{\mathcal{C}}$. The maximum for $\alpha_{\mathcal{C}}$ occurs when for all $j = 1, 2, \ldots, n$:

$$\frac{\delta}{\delta B_j}\alpha_{\mathcal{C}} = k, \text{ for some constant k}$$

$$\text{Thus, } h_j(B_j) = k \cdot \frac{\sum_{i=1}^{n} R_i}{R_j} \quad (2)$$

where $h_j(B_j)$ denotes the PDF corresponding to $H_j(B_j)$. In equation 2 the value of k is chosen so as to satisfy the constraint $B_0 = B_1 + B_2 + \ldots + B_n$. Using an exponential model to approximate the function, we get that for $i = 1, 2, \ldots, n$, $H_i(b) = 1 - e^{-\lambda_i . b}$ where λ_i is the distribution's constant. The PDF corresponding to $H_i(b)$ is $h_i(b)$, where

$$h_i(b) = \lambda_i e^{-\lambda_i . b} \quad (3)$$

Given a particular server $\mathcal{S}_j$, where $1 \leq j \leq n$, we substite for $h_j(b)$ in equation 2, obtaining the following formula.

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

Equation 4 specifies a set of n equations to ration the total buffering space B_0 available at $\mathcal{S}_0$ amongst the servers $\mathcal{S}_i$, for $i = 1, 2, \ldots, n$. In order to do so, we must find the value of the constant k. This can be done by solving for the equality $B_0 \leq B_1 + B_2 + \ldots + B_n$, which results in the following expression for k.

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

Substituting the value of k from equation 5 into equation 4, we get the optimum storage capacity to allocate on $\mathcal{S}_0$ for a particular server $\mathcal{S}_j$, where $1 \leq j \leq n$.

The above calculations require that R_i and λ_i be estimated, for $i = 1, 2, \ldots, n$. This can be done by analyzing server logs. As a matter of fact, figures 1 and 2 were produced by programs that computed these parameters for `http://cs-www.bu.edu`.

In order to appreciate the effectiveness of our demand-based document dissemination, we consider a symmetric cluster, where all servers have identical values for R_i and λ_i. From equations 5 and 4, we get

$$k = \frac{\lambda}{n} \cdot e^{-\frac{\lambda}{n}B_0}$$

$$B_j = \frac{B_0}{n} \quad (6)$$

As expected, equation 6 provides equal allocation of storage on $\mathcal{S}_0$ for all the servers in the cluster. By substituting the value of B_j into equation 1, we get:

$$\alpha_{\mathcal{C}} = 1 - e^{-\lambda\frac{B_0}{n}} \quad (7)$$

Equation 7 could be used to estimate the storage requirements on the proxy as a function α.

$$B_0 = \frac{n}{\lambda}\log\frac{1}{\alpha_{\mathcal{C}}} \quad (8)$$

Equation 8 suggests that if (say) the `cs-www.bu.edu` server is only one of 10 servers, whose most popular data are duplicated on a proxy, then in order to reduce the remote bandwidth by (say) 90% on *all* servers, the proxy must secure 36 MBytes to be divided equally amongst all servers. This assumes a value of $\lambda = 6.247 \times 10^{-7}$, which was estimated from the HTTP demon logs on the `cs-www.bu.edu` server. With a storage capacity of 500 MBytes, a proxy could shield 100 servers from as much as 96% of their remote bandwidth. These numbers, of course, raise a legitimate question: If 96% of all remote accesses to 100 servers (or even 90% of all accesses to 10 servers) are now to be served by *one* proxy, isn't that proxy going to become a performance bottleneck? The answer is, of course, 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, if (or when) it is determined that the proxy is overloaded, then B_0 could be reduced, thus forcing more of the requests back to the servers.

3 Conclusion and Future Work

Recently, there has been a flurry of research on replication techniques to reduce network traffic and minimize the latency of distributed multimedia information retrieval systems. Most of these techniques have concentrated on client-based caching, whereby recently/frequently accessed information is cached at the client (or at a proxy thereof) in anticipation of future accesses to the same information [2] . We believe that these local (myopic) solutions, focussing exclusively on a particular client or set of clients, are likely to have limited impact. In this paper we offer a more global solution that

allows the replication of information to be done on a supply/demand basis. In particular, we propose a hierarchical demand-based replication strategy that *optimally* disseminates information from its producer to servers that are closer to its consumers. The level of dissemination depends on the popularity of that document (relative to other documents in the system), and the expected reduction in traffic that results from its replication.

Demand-based dissemination of multimedia information from producers to consumers is not a new idea: it is used in the retail of commodities, newspaper distribution, among other things. In this paper, we propose to use the same philisophy for distributed information systems. In particular, we have presented an analytical model (supported by data from actual logs of typical institutional and dedicated servers) that demonstrates how such dissemination could be done, both efficiently and with minimal changes to the prevailing client-server infrastructure of the Internet.

There are many reasons for advocating the development of an automated information dissemination protocol as a way of controlling traffic as opposed to simply increasing the available bandwidth in the system. First, we believe that adding servers (*i.e.* proxies) to the internet is much cheaper than adding (upgrading) internet links [6]. Second, we believe that increasing the available bandwidth is a temporary solution; it's only a matter of time before the added bandwidth is consumed by the ever increasing number of users.

In this paper we considered only one of many problems that need to addressed in future distributed information systems, such as those intended to contribute to the solution of the National Grand Challenges of the High Performance Computing and Communications initiative. While it may be possible to develop an integrated solution for a collection of these problems, we believe that it is important to keep (as much as possible) the solution of *orthogonal problems* independent, and thus composable. For example, we believe that the problem of information dissemination from producers to consumers is orthogonal to the problem of mapping from "logical" to "physical" object names. To be scalable, a distributed information system (such as the WWW) must provide its users with naming conventions and name-resolution protocols that provide *location/replication transparency*. Such a naming protocol should not be dependent on (say) a particular dissemination/replication protocol, like the one presented in this paper. Other orthogonal problems include that of *clustering* (grouping servers/clients into dynamic clusters to reduce traffic) and resource discovery (locating nearby copies of replicated resources) [7]. Much work needs to be done to identify and tackle such *orthogonal* problems and then compose their solutions.

Acknowledgments:

I would like to thank all members of the *Oceans* research group: Mark Crovella, Abdelsalam Heddaya, Bob Carter, Sulaiman Mirdad, and Carlos Cunha, for the many discussions and feedback on this work. Also, I would like to thank Stephan Fitch and Stan Sclaroff for providing me with the Rolling Stones logs.

References

[1] Swarup Acharya and Stanley B. Zdonik. An efficient scheme for dynamic data replication. Technical Report CS-93-43, Brown University, Providence, Rhode Island 02912, September 1993.

[2] Azer Bestavros. Application level document caching in the internet. In *IEEE SDNE'96: The Second International Workshop on Services in Distributed and Networked Environments*, Whistler, British Columbia, June 1995.

[3] Azer Bestavros. Demand-based resource allocation to reduce traffic and balance load in distributed information systems. In *Proceedings of SPDP'95: The 7^{th} IEEE Symposium on Parallel and Distributed Processing*, San Anotonio, Texas, October 1995.

[4] Matthew Addison Blaze. *Caching in Large Scale Distributed File Systems.* PhD thesis, Princeton University, January 1993.

[5] Michael D. Dahlin, Randolph Y. Wang, Thomas E. Anderson, and Dacid A. Patterson. Cooperative caching: Using remote client memory to improve file system performance. In *First Symposium on Operating systems Design and Implementation (OSDI)*, pages 267–280, 1994.

[6] Peter Danzig, Richard Hall, and Michael Schwartz. A case for cashing file objects inside internetworks. Technical Report CU-CS-642-93, University of Colorado at Boulder, Boulder, Colorado 80309-430, March 1993.

[7] James Guyton and Michael Schwartz. Locating nearby copies of replicated internet servers. Technical Report CU-CS-762-95, University of Colorado at Boulder, Boulder, Colorado 80309-430, February 1995.

[8] James Gwertzman and Margo Seltzer. The case for geographical push-caching. Technical Report HU TR-34-94 (excerpt), Harvard University, DAS, Cambridge, MA 02138, 1994.

[9] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems*, 6(1):51–81, February 1988.

[10] D. Muntz and P. Honeyman. Multi-level caching in distributed file systems or your cache ain't nuthing but trash. In *Proceedings of the Winter 1992 USENIX*, pages 305–313, January 1992.

[11] Christos H. Papadimitriou, Srinivas Ramanathan, and P. Venkat Rangan. Information caching for delivery of personalized video programs on home entertainment channels. In *Proceedings of the International Confrence on Multimedia Computing and Systems*, pages 214–223, May 1994.