

On Reliable and Scalable Peer-to-Peer Web Document Sharing *

Li Xiao and Xiaodong Zhang
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187-8795
{lxiao, zhang}@cs.wm.edu

Zhichen Xu
Internet Systems and Storage Laboratory
Hewlett Packard Laboratories
Palo Alto, CA 94304
zhichen@hpl.hp.com

Abstract

We propose a peer-to-peer Web document sharing technique, called "Browsers-Aware Proxy Server". In this design, a proxy server connecting to a group of networked clients maintains an index file of data objects of all clients' browser caches. If a user request misses in its local browser cache and the proxy cache, the browsers-aware proxy server will search the index file attempting to find it in another client's browser cache before sending the request to an upper level proxy or the web server. If such a hit is found in a client, this client will directly forward the data object to the requesting client; or the proxy server fetches the data object from this client and then forwards it to the requesting client. Our trace-driven simulations show that the amount of sharable data in browser caches is significant and can be utilized for peer-to-peer document sharing to improve Web caching performance and scalability. We also address the issues of data integrity and communication anonymity in order to build a reliable browsers-aware proxy server.

1. Introduction

Peer-to-peer (P2P) computing is an emerging distributed computing technology that enables direct resource sharing of both computing services and data files among a group of mutually trusted clients over the Internet. Researchers and developers from industries and academia are making standards to handle arisen issues surrounding the deployment of peer-to-peer computing in applications, such as security, interoperability, reliability, and availability [6, 15]. The Peer-to-Peer Trusted Library (PtPTL) [15] is such an example to provide security support to peer-to-peer computing. However, most existing distributed computing infrastructures are

client-server oriented. Internet proxy caching is a representative example. In this study, we propose to restructure the existing proxy model for P2P Web caching.

Proxy caching is an effective solution to quickly access and reuse the cached data on the client side and to reduce Internet traffic to web servers. A group of networked clients connects to a proxy cache server, where each client has a browser with its cache buffering popular and recently requested data objects. Upon a web request of a client, the browser first checks if it exists in the local browser cache. If so, the request will be served by its own browser cache. Otherwise the request will be sent to the proxy cache. If the requested data object is not found in the proxy cache, the proxy server will immediately send the request to its cooperative caches, if any, or to an upper level proxy cache, or to the web server *without considering if it exists in other browsers' caches*. We believe there are three practical reasons for a proxy server to exclude this consideration. First, the browser caches are not shared for software simplicity and user privacy reasons; and the dynamic status in each cache is unknown to the proxy server. Second, the possibility of a proxy cache miss which is a browser cache hit may have been considered low although no such a study has been found in literature. Finally, a browser cache was initially developed as a small data buffer with a few simple data manipulation operations. Users may not effectively retain the cached data with a high quality of spatial and temporal locality. It is desirable to understand the potential performance gain by sharing data among browsers before making efforts to design and implement browsers-aware proxy servers with enforced or optional security considerations. We have following qualitative arguments for it.

First, since browser caches are not shared among themselves, the size of the proxy cache is limited, and no data consistency is maintained between browser caches and the proxy cache, it is certainly possible that a data object is stored in one or more browser caches, but has been replaced

*This work is supported in part by the U.S. National Science Foundation under grants CCR-9812187, EIA-9977030, and CCR-0098055, and by a USENIX Research Scholarship.

in the proxy cache.

Secondly, browsers provide a function for users to set the browser cache size. With the rapid increase of memory and disk capacity in workstations and PCs, and with the rapid growth of web applications, user browser cache size will tend to increase as time passes. In addition, several new software techniques are introduced for users to effectively increase the browser cache size. For example, “browser cache switch” [5], allows users to set multiple browser caches in one machine, and to switch them from one to another during web browsing. Thus, different caches can be used for different contents and for different time periods. This technique significantly increases the size of a browser cache for an effective management of multiple data types. However, the larger the browser cache size is set, the more spatial locality will be under utilized by the proxy cache server.

Thirdly, in order to help web users to effectively use and manage large browser cache data, browser software has been upgraded to include several sophisticated database functions, such as file searching, folding, and grouping. With the aid of these functions, users will pay more attention to the organized browser cache data objects, and tend to keep them in the cache much longer than to keep the unorganized data objects. However, the longer the cached data is retained, the more temporal browser cache locality will be under utilized by a proxy cache server.

Fourthly, in order to improve the browsing speed, a large memory drive can be configured to store the entire browser cache. This technique of “browser cache in memory”, has been implemented in several commercial browsers, such as Internet Explore and Netscape. This technique can be further extended to periodically save the cached data objects in a special directory in the disk. The data will be brought back from the disk to the special memory drive whenever the system is restarted or rebooted. In modern computer systems, transferring data through a moderate speed network will be significantly faster than obtaining the same amount of data from a local disk through page faults. The high speed memory access is not only beneficial to a local user, but also speeds up data accesses for remote users to share browser caches.

Finally, the number and types of web servers have increased and will continue to increase dramatically, providing services to a wider and wider range of clients. Thus, the number of unique file objects cached in client browsers has increased and will continue to increase. It is impossible for proxy caches to cover all multi-requested file objects of tremendous types even with a perfect cache replacement strategy, increasing the possibility for browser caches to keep file objects that have been replaced in proxy caches.

In summary, the quality of spatial and temporal locality in browser caches has been and will continue to be im-

proved, inevitably providing a rich and commonly sharable P2P storage among trusted Internet peers. In this study, we introduce a P2P technique to fully utilize browser caches, called “browsers-aware proxy server”. Conducting trace-driven simulations, we quantitatively evaluate its potential benefits for further improving proxy caching performance. We also address the issues of data integrity and communication anonymity in order to build a reliable browsers-aware proxy server. Our effort shares the same objective of building effective P2P infrastructure that lets users easily and reliably share files and processing powers over the Internet.

2. Browsers-Aware Proxy Server

In this design, the proxy server connecting to a group of networked clients maintains an index file of data objects of all clients’ browser caches. If a user request misses in its local browser cache and the proxy cache, the browsers-aware proxy server will search an index file attempting to find it in a client’s browser cache before sending the request to an upper level server. If such a hit is found in a client, we propose two alternative implementations to let the requesting client access the data object. First, the proxy server will inform this client to directly forward the data object to the requesting client. In order to retain user browsers’ privacy, the message passing from the source client to the requesting client should be anonymous to each other. The second implementation alternative is to make the proxy server provide the data by loading the data object from the source client and then storing it to the requesting client.

In order to implement the browsers-aware concept in a proxy server, we create a *browser index file* in the proxy server. This index file records a directory of cached file objects in each client machine. Each item of the index file includes the ID number of a client machine, the URL including the full path name of the cached file object, and, if any, a time stamp of the file or the TTL (Time To Live) provided by the data source. Since the dynamic changes in browser caches are only partially visible to the proxy server (when a file object is sent from the proxy cache to the browser), the browser index file will be updated periodically by each browser cache. Here is another alternative. After a file object is sent from the proxy server to a client’s browser cache, its index item is added to the browser index file. Whenever this file object is replaced or deleted from the browser cache, the client sends an invalidation message to the proxy server. After then, the proxy deletes the corresponding index item.

Figure 1 presents the organization of the browsers-aware proxy server by an example. A group of client machines is connected by a local area network. For a given web service request with a specific URL in client machine i , the browser cache is first searched attempting to satisfy the request. After the request misses in the browser cache, client

i sends the request to the proxy server, where the proxy cache is searched for the same purpose. After the request misses again in the proxy cache, the browser index file is searched, where the URL is matched in client machine j . The proxy server informs client machine j to forward the cached file object to client i , or fetches the cached object from machine j and then forwards it to client i .

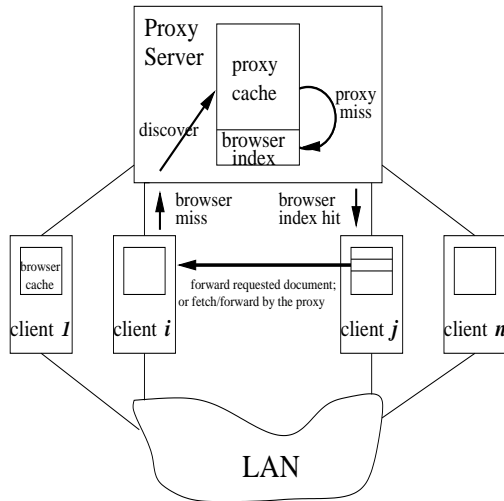


Figure 1. Organizations of the browsers-aware proxy server.

3. Simulation Environment

The browsers-aware proxy server and related performance issues are evaluated by trace-driven simulations.

3.1. Traces

Table 1 lists the web traces we have used for performance evaluation, where *infinite* cache size is the total size storing all the unique requested documents.

1. NLANR traces: NLANR (National Lab of Applied Network Research) provides sanitized cache access logs for the past seven days in the public domain [11]. NLANR takes special steps to protect the privacy of those participating in their cache mesh. Client IP addresses are randomized from day-to-day, but consistent within a single log file. Client IP addresses are very important in our study, so we use traces based on one day's log file. NLANR provides about ten proxies' traces. We have used one day's trace of July 14, 2000 from the "uc" proxy and one day's trace of August 29, 2000 from the "bo1" proxy, which are denoted as NLANR-uc and NLANR-bo1.

2. BU traces: Boston University collected traces from the similar computing facility and user population in 1995 and 1998, which can be found in [2]. We select the traces in a period of two months of the two years, which are denoted as BU-95 and BU-98, respectively.
3. CA*netII traces: The CA*netII (Canada's coast to coast broadband research network) parent cache provides sanitized log files in [3]. The client IDs are consistent from day to day, so we concatenate two days' logs together as our trace. The two logs we used are the traces collected on September 19, 1999 and September 20, 1999, which are the most recent traces in this site.

3.2. A browsers-proxy caching environment

We have built a simulator to construct a system with a group of clustered clients connecting to a proxy server. The cache replacement algorithm used in our simulator is LRU. All the traces have the size of a document for each request. If a user request hits on a document whose size has been changed, we count it as a cache miss. We have implemented and compared the following five web caching organizations using the trace-driven simulations:

1. *Proxy-cache-only*: each client does not have a browser cache. Every client request is sent directly to the proxy cache server.
2. *Local-browser-cache-only*: each client has a private browser cache, but there is no proxy cache server for client machines.
3. *Global-browsers-cache-only*: each client has a browser cache which is globally shared among all the clients by maintaining an index file in each client machine. The index file records a directory of cache documents of all clients. A browser does not cache documents fetched from another browser cache. If a request is a miss in its local browser, the client will check the index file to see if it is stored in other browser caches before sending the request to a web server. There is no proxy cache server for the group of client machines.
4. *Proxy-and-local-browser*: each client has a private browser cache, and there is a proxy cache server for the group of client machines. If a request misses in its local browser, it will be sent to the proxy to check if the requested document is cached there. If it misses again, the proxy will send the request to an upper level server.
5. *browsers-aware-proxy-server*: this is the enhanced proxy caching technique presented in section 2.

Traces	Period	# Requests	Total GB	<i>Infinite</i> Cache (GB)	# Clients	Max Hit Ratio	Max.Byte Hit Ratio
NLANR-uc	7/14/00	360806	4.36	3.72	95	19.11%	14.80%
NLANR-bo1	8/29/00	263942	1.71	1.22	115	21.32%	28.79%
BU-95	Jan.95-Feb.95	502424	1.31	0.90	591	64.14%	31.37%
BU-98	Apr.98-May 98	72626	0.45	0.29	306	40.62%	35.94 %
CA*netII	9/19-/9/20/99	745943	0.089	0.062	3	34.20%	29.84%

Table 1. Selected Web Traces.

We use two Performance metrics. *Hit ratio* is the ratio between the number of requests that hit in browser caches or in the proxy cache and the total number of requests. *Byte hit ratio* is the ratio between the number of bytes that hit in browser caches or in the proxy cache and the total number of bytes requested.

4. Performance Evaluation

Based on real-world proxy configurations reported in [14], we define a *minimum* browser cache size as $\frac{Cache_{proxy}}{m}$, where m is the number of clients, and $Cache_{proxy}$ is the size of the proxy cache responsible for the m clients. We also conservatively define an *average* browser cache size as $\frac{\beta Cache_{proxy}}{m}$, where β is in a range of 2 to 10.

4.1. How much is browser cache data sharable?

To answer this question, we have operated the five caching policies with different traces on a simulated web caching environment where the browser cache size of each client is set to *minimum*. Performance results of all the traces we have used are quite consistent. Due to the page limit, we only present the results of hit ratios and byte ratios from the NLANR-uc trace in Figure 2, where the size of the proxy cache is scaled from 0.5%, 5%, 10%, and to 20% of the *infinite* proxy cache size, the browser cache size is also scaled up accordingly.

Figure 2 shows that the hit ratios (left) and byte hit ratios (right) of the *browsers-aware-proxy-server* are the highest, particularly, the hit ratios are up to 5.94% higher and the byte hit ratios are 9.34% higher than those of the *proxy-and-local-browser*, even when the browser cache size is set to *minimum*. This means that sharable data locality does exist, even for a small browser cache size. The sharable data locality proportionally increases as browser cache size increases and as the number of unique file objects cached in browsers increases, both of which are the trends in web computing. In next subsection, We will show that significant proxy cache performance improvement can be achieved by the proposed browsers-aware proxy server to exploit sharable data locality.

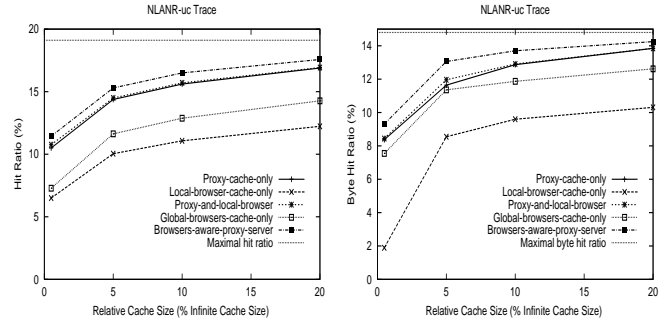


Figure 2. The hit ratios and byte hit ratios of the five caching policies using NLANR-uc trace, where the browser cache size is set *minimum*.

We also show that methods of *proxy-cache-only*, *local-browser-cache-only*, and *global-browsers-cache-only* are not as effective as the method of *proxy-and-local-browser*. *Local-browser-cache-only* had the lowest hit and byte hit ratios due to the minimum caching space. *proxy-and-local-browser* only slightly outperforms *proxy-cache-only*, which implies that performance gain from a local browser cache is limited. Another observation worth mentioning is that *proxy-and-local-browser* and *global-browsers-cache-only* had lower hit and byte hit ratios than *browsers-aware-proxy-server*. This observation confirms the existence two types of misses. First, there exist some documents which are already replaced in the proxy cache but still retained in one or more browser caches, because the request rates to the proxy and to browsers are different, causing the replacement in the proxy and browsers at a different pace. Second, there are some documents which are already replaced in browser caches but still retained in the proxy cache, because a browser cache is much smaller than the proxy cache. The *browsers-aware-proxy-server* effectively addresses these two types of misses.

Figure 3 presents the breakdowns of the hit ratios and the byte hit ratios of the *browsers-aware-proxy-server* using NLANR-uc trace. There are three types of hits: hits in the local browser cache, hits in the proxy cache, and hits in

remote browser caches. We show that the hit ratio and byte hit ratio in remote browser caches should not be neglected even when the browser cache size is very small.

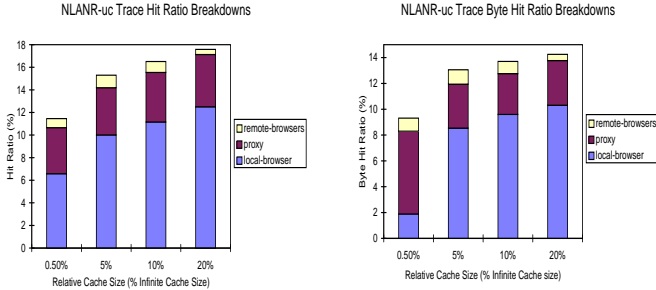


Figure 3. The breakdowns of the hit ratios and byte hit ratios of the browsers-aware proxy using NLANR-uc trace, where the browser cache size is set *minimum*.

The *browsers-aware-proxy-server* has another advantage over the *proxy-and-local-browser* policy in terms of “memory” byte hit ratios. In other words, for the same byte hit ratio, a higher percentage of requests will hit in the main memory of browser caches and the proxy cache provided by the *browsers-aware-proxy-server*. To quantitatively justify this claim, we have compared the memory byte hit ratios of the two policies for an equivalent byte hit ratio.

In our simulation, we set the memory cache size in the proxy as 1/150 of the proxy cache size based on the memory ratio reported in [14]. We also set the memory size of a browser cache as 1/150 of the browser cache size, which is not in favor of the *browsers-aware-proxy-server* because the memory cache portion in a browser can be much larger than that for the proxy cache in practice. We also conservatively assume that one memory access of one cache block of 16 Bytes spends 200 ns (the memory access time is lower than this in many advanced workstations), and one disk access of one page of 4 KBytes is 10 ms.

Figure 2 shows that the hit and byte hit ratios of the *browsers-aware-proxy-server* at 5% of the *infinite* cache size are very close to those of the *proxy-and-local-browser* policy at 10% of the *infinite* cache size (the hit ratio comparison is 15.3 v.s. 15.7, and byte hit ratio comparison is 13.06 v.s. 12.91). However, the memory byte hit ratios of the two schemes are quite different under the same condition, which are 3.5% for the *browsers-aware-proxy-server*, and 1.9% for the *proxy-and-local-browser* policy, respectively. The larger memory byte hit ratio of the *browsers-aware-proxy-server* in this case would reduce 15.2% of the total hit latency compared with the *proxy-and-local-browser*. The latency reduction due to the higher percentage memory ac-

cesses will be larger in practice because the memory cache size of each browser is much larger than the assumed size.

4.2. Performance of browsers-aware proxy server

We have evaluated and compared the performance of the *browsers-aware-proxy-server* and *proxy-and-local-browser* schemes using the NLANR-bo1 trace and two BU traces. For experiments of each trace, the proxy cache size is set to 0.5%, 5%, 10%, and 20% of the *infinite* proxy cache size. Accordingly, each browser cache is also set to 0.5%, 5%, 10%, and 20% of the average *infinite* browser cache size calculated from all the browsers. The *infinite* cache size of a browser is the total size of all uniquely requested documents in this client. The value of β calculated from each trace falls into the *average* range of 2 to 10.

Figures 4, 5, and 6 present the hit ratios (left) and byte hit ratios (right) of the two policies on NLANR-bo1 trace, BU-95 trace, and the BU-98 trace, respectively. Compared with the *proxy-and-local-browser* scheme, *browsers-aware-proxy-server* consistently and significantly increases both hit ratios and byte hit ratios on all the traces.

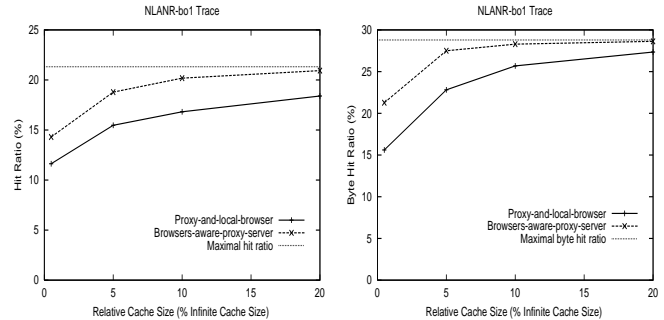


Figure 4. The hit rates and byte hit rates of the browsers-aware-proxy-server and proxy-and-local-browser scheme using NLANR-bo1 trace, where the browser cache size is set *average*.

The limit of the Browsers-Aware Proxy Server

When the number of clients is small, and their accumulated size of the browser caches is much smaller or not comparable to the proxy cache size, the cache locality inherent in browsers is low, so the performance gain from the browsers-aware proxy cache will also be insignificant. Figure 7 presents such an example, where the total number of clients of the CA*netII trace is only 3, the accumulated size of three browser caches is small. The increases of both average hit ratio and byte hit ratio of this trace by the *browsers-aware-proxy-cache* are below 1%, compared with the *proxy-and-local-browser* scheme.

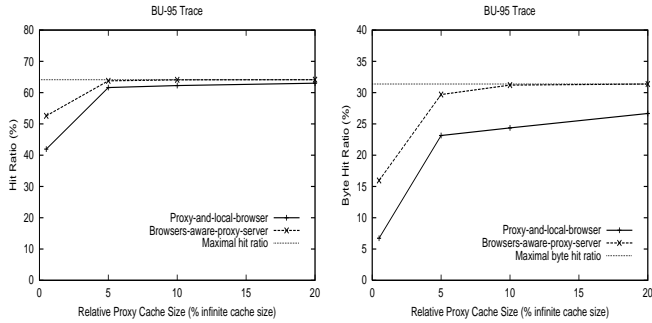


Figure 5. The hit rates and byte hit rates of the *browsers-aware-proxy-server* and the *proxy-and-local-browser* scheme using the BU-95 trace, where the browser cache size is set *average*.

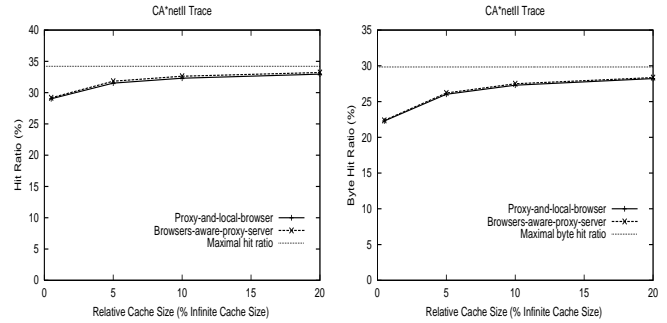


Figure 7. The hit ratios and byte hit ratios of the *browsers-aware-proxy-server* and *proxy-and-local-browser* scheme using the CA*netll trace.

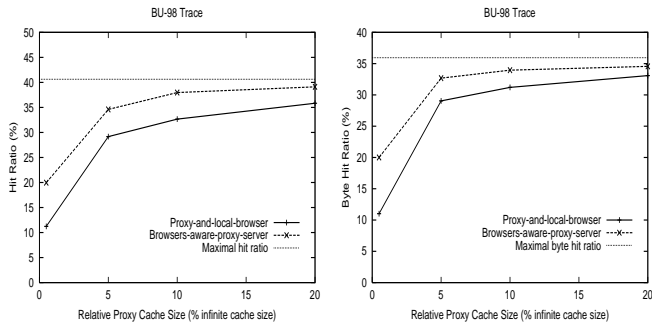


Figure 6. The hit rates and byte hit rates of the *browsers-aware-proxy-server* and the *proxy-and-local-browser* scheme using the BU-98 trace, where the browser cache size is set *average*.

4.3. Performance Impact of Scaling the Number of Clients

We have also evaluated the effects of scaling the number of clients to browsers-aware proxy servers. For each trace, we observe its hit ratio (or byte hit ratio) increment changes by increasing the number of clients from 25%, to 50%, to 75%, and to 100% of the total number of clients. We also call each percentage as a relative number of clients. For all relative numbers of clients of each trace, the proxy cache size is fixed to 10% of the *infinite* proxy cache size when the relative number of clients is 100%. The byte hit ratio increment or the hit ratio increment of the browsers-aware proxy server for a given trace is defined as

$$\frac{(\text{byte}) \text{ hit ratio of browser-aware} - (\text{byte}) \text{ hit ratio of proxy-and-local-browser}}{(\text{byte}) \text{ hit ratio of proxy-and-local-browser}}$$

Figure 8 presents the hit ratio increment curves (left figure) and the byte hit ratio increment curves (right figure) of the three traces as the relative number of clients changes from 25% to 100%. Our trace-driven simulation results show

that both hit ratio increment and byte hit ratio increment of the browsers-aware proxy server proportionally increases as the number of clients increases. For some traces, the increments are significant. For example, the hit ratio increment of BU-98 trace increases from 10.70% to 13.35%, to 16.87%, and to 19.35%, as the relative number of clients increases from 25% to 50%, to 75%, and to 100%, respectively. The byte hit ratio increment of BU-95 trace increases from 4.33% to 20.17%, to 24.82%, and to 28.08%.

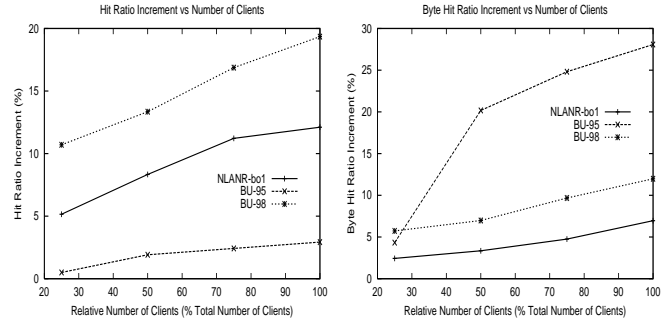


Figure 8. The hit ratio and byte hit ratio increments of the *browsers-aware-proxy-server* over the *proxy-and-local-browser*.

The performance results indicate that a browsers-aware proxy server is performance beneficial to client cluster scalability because it exploits more browser locality and utilizes more memory space as the number of clients increases.

5. Overhead Estimation

The additional overhead of the browsers-aware proxy cache comes from the data transferring time for the hits in remote browsers. The simulator estimates the data transferring time based on the number of remote browser hits and

their data sizes on a 10 Mbps Ethernet. Setting 0.01 second as the network connection time, we show that the amounts of data transferring time and the bus contention time spent for communication among browser caches of the browsers-aware proxy server on all the traces is very low. For example, the largest accumulated communication and network contention portion out of the total workload service time for all the traces is less than 1.25%. In addition, the contention time only contributes up to 0.12% of the total communication time, which implies that browsers-aware proxy server does not cause bursty hits to remote browser caches.

Another potential overhead is the update of the browser index file if the update is not conducted at a suitable time or conducted too frequently. There have been some solutions to address this concern. For example, the browser could send its update information when the path between the browser and the proxy is free to avoid contention. The study in [4] shows that the update of URL indices among cooperative caches can be delayed until a fixed percentage of cached documents are new. The delay threshold of 1% to 10% (which corresponds to an update frequency of roughly every 5 minutes to an hour in their experiments) results in a tolerable degradation of the cache hit ratios. In their experiments, the degradation is between 0.02% to 1.7% for the 1% choice. Our concerns should be less serious because the updates are only conducted between browsers and the proxy without broadcasting. Thus, the index file update overhead between browsers and proxy is very low.

The last potential concern is the space requirement of the proxy cache to store the browser index. We address this concern by an example. Each URL is represented by a 16-byte MD5 signature [9]. Assume there are 1000 clients connected to one proxy. Each client has a browser with a 8MB cache. We assume that an average document size is 8 KB. Each browser has about 1 K web pages. The proxy server only needs about $1000 \times (8MB/8KB) \times 16 = 16MB$ to store the whole browser index file for the 1000 browsers. If we apply the compression methods presented in [4] or [10], the browsers-aware proxy server requires even less space to store the index file. (e.g. a storage of 2 MB is sufficient for the 1000 browsers with a tolerant inaccuracy).

6. Reliability and Security

In order to make the browsers-aware proxy server feasible in practice, the reliability and security of the browser data must be seriously considered [16]. For example, the browser data files that have been modified by an owner client are not reliable for sharing among clients. In addition, the identities of requesting browser and hit browser, and the hit document should not be visible among clients to preserve the privacy of each client. These concerns can be addressed by ensuring data integrity and making anonymous communications between clients.

We have proposed protocols to enforce data integrity and communication anonymity [17]. Our study shows that the associated overheads are trivial. These protocols are based on symmetric and public key encryptions [9]. In a symmetric key system, two communicating parties share an identical secret, the symmetric key, used for encryption and decryption. DES (Data Encryption Standard) is such an example. In a public key system (e.g. RSA), such party has a public/private key pair. A public key can be accessed by everyone. A sender encrypts an outgoing message using a receiver's public key, and the receiver uses its private key to decrypt this ciphertext.

6.1. Data Integrity

To ensure that a document received by a client is tamper-proof, we need to find a way for a requesting browser to check whether the content it receives is intact. For this purpose, we use the proxy server to produce a digital water mark in the following manner: for a document f , the digital water mark is produced by first generating a message digest using MD5 [13], and then encrypt the message digest with the proxy server's private key. (We assume that the private key of the proxy is x , the corresponding public key is y , and the public keys of the browser caches are known to all peer clients. We use $K(M)$ to represent either (i) the message M being encrypted with the key K , or (ii) the ciphered message M being decrypted with decryption key K .)

Initially, when a client c_i , sends a request to the proxy for a document, the proxy obtains the requested document, denoted as f , either from the server or an upper level proxy. The proxy generates a MD5 message digest, $h(f)$, of the document. It then encrypts $h(f)$ with its private key x to produce $x(h(f))$. The message $\{f, x(h(f))\}$ is sent to the client c_i and stored in its local cache. If another client c_j requests the same document, and this document has been replaced in the proxy cache and is found to be in c_i 's cache, the proxy will instruct c_i to send the message $\{(h(f)), f\}$ to c_j . On receiving the message, c_j will produce a message digest of the document using MD5, and compare the message digest with $y(x(h(f)))$. No client can temper with the document f and produce a matching digital water mark, because no client but the proxy server knows the private key of the proxy server.

6.2. Communication Anonymity

Our browsers-aware proxy system hides the identities of both browser senders and receivers. This communication anonymity is ensured by having the proxy act as an anonymizing proxy. A client always sends a request to the proxy. The proxy contacts a targeted client and receives the content on behalf of the requesting client. The targeted

client does not know which client requests the document, and a requesting client does not know which client delivers the content. We have also developed several anonymity protocols that hide identities among peer browsers with no or limited centralized controls of the proxy [17].

7. Related Work

Paper [7] studies Web content sharing in a multi-level Internet storage hierarchy without considering client browsers. Designs and implementations of browsers have been studied in many papers, e.g. [8], [12]. The work in [18] attempts to transfer server's functions to clients. The results of above cited work are supportive to our work based on increasingly powerful browsers.

Paper [1] gives a comprehensive study on the changes in web client access patterns based on the traces collected from the same computing facility and the same nature of the user population separated by three years. Their experiments show that the hit ratios are significantly reduced compared with the data three years ago. One reason for this, we believe, is that the access variations have increased as more web servers are emerging. Thus, It will be more difficult to retain an optimal hit ratio by increasing proxy cache size and by improving cache replacement strategy. The browsers-aware proxy server has the potential to harvest the sharable data to adapt the trend of the increasing access variations.

8. Conclusion

We have proposed and evaluated a browsers-aware proxy server to provide a distributed P2P Web document sharing service. We have also quantitatively answered two questions: how much browser data is sharable? and how much proxy caching performance improvement can we gain by this P2P approach? Could the browsers-aware proxy server be scalable and reliable? Our study shows that the amount of sharable data is significant and should be utilized and the proxy caching performance can be significantly improved by the proposed browsers-aware structure that is scalable and reliable. We are currently implementing a browsers-aware proxy server along with data integrity and communication anonymity protocols.

Acknowledgments: We thank Bill Bynum for reading the paper and for his constructive comments. The comments from the anonymous referees are very constructive and helpful. This work is also a part of an independent research project sponsored by the National Science Foundation for program directors and visiting scientists.

References

- [1] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications", *World Wide Web Journal*, 2(1):15-28, January, 1999.
- [2] BU traces. <ftp://cs-ftp.bu.edu/techreports/1995-010-www.client-traces.tar.gz>
<ftp://cs-ftp.bu.edu/techreports/1999-011-usertrace-98.gz>
- [3] Canada's coast to coast broadband research network: <http://ardnoc41.canet2.net/>;
Sanitized log files:
<http://ardnoc41.canet2.net/cache/squid/rawlogs/>
- [4] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol", *Proceedings of 1998 SIGCOMM Conference*, pp. 254-265.
- [5] J. Fox, "Browser cache switch for Internet explorer", *Web-Developer Conference 2000*, San Francisco, California, September 2000.
- [6] Pat Gelsing, "Building the Peer-to-Peer Community", *Intel Developer Forum Conference, Spring 2001*, Keynote Presentations, February 2001. <http://developer.intel.com/idf>
- [7] S. Gadde, M. Rabinovich, J. Chase, "Reduce, Reuse, Recycle: An Approach to Building Large Internet Caches", *Proceedings of the sixth Workshop on Hot Topics in Operating Systems*, May, 1997.
- [8] T. S. Loon and V. Bharghavan, "Alleviating the Latency and Bandwidth Problems in WWW Browsing", *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems*, December 1997.
- [9] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, October 1996.
- [10] B. S. Michel, K. Nikoloudakis, P. Reiher, and L. Zhang, "URL Forwarding and Compression In Adaptive Web Caching", *Proceedings of IEEE INFOCOM 2000*, March, 2000.
- [11] National Lab of Applied Network Research:
<http://www.ircache.net/>
Sanitized access logs: <ftp://ircache.nlanr.net/Traces/>
Statistics: <http://www.ircache.net/Cache/Statistics/>
- [12] M Reddy and G. P. Fletcher, "An Adaptive Mechanism for Web Browser Cache Management", *IEEE Internet Computing*, 2(1), January 1998.
- [13] R. Rivest, "The MD5 Message-Digest Algorithm", *Internet RFC/STD/FYI/BCP Archives*, Request for Comments: 1321, April 1992. (<http://www.faqs.org/rfcs/rfc1321.html>).
- [14] A. Rousskov and V. Soloviev, "A Performance Study of the Squid Proxy on HTTP/1.0", *World Wide Web*, 2(1-2):47-67, January 1999.
- [15] Working Group on Peer-to-Peer Computing.
<http://www.peer-to-peerwg.org>
- [16] L. Xiao and X. Zhang, "Exploiting neglected data locality in browsers", *Proceedings of the 10th International World Wide Web Conference (WWW-10)*, Hong Kong, May 1-5, 2001 (an extended abstract).
- [17] Z. Xu, L. Xiao, and X. Zhang, "Data integrity and communication anonymity in peer-to-peer networks", Hewlett Packard Laboratories, Technical Report HPL-2001-204, August 2001.
- [18] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler, "Using Smart Clients to Build Scalable Services", *Proceedings of the USENIX 1997 Annual Technical Conference*, January, 1997.