支援資料預取與自動更新之使用者導向代理伺服器
A User-Oriented Proxy Server Supporting Data Prefetching and Refreshing

蔡尚榮
Shang-Rong Tsai
tsai@turtle.ee.ncku.edu.t

董仲愷
Chung-Kie Tung
tung@turtle.ee.ncku.edu.t

王宏銘
Hung-Ming Wang
whm@turtle.ee.ncku.edu.t

Distributed System Laboratory,Department of Electrical Engineering
National Cheng-Kung University, Taiwan, R.O.C.

摘要

本論文中提出一支援資料預取與自動更新之使用者導向代理伺服器，讓使用者能以訂閱方式將網頁快取至代理伺服器上，並由代理伺服器負責自動更新，以減少使用者存取網頁所需的時間，同時讓使用者能由代理伺服器上取到最新的資料.

Abstract

*Most studies of the proxy server focus on increasing the cache hit rate of the proxy server by caching the most popular pages. But the most popular pages are not what users need sometimes, especially for the professional. In this paper *, we propose a user oriented proxy system supporting prefetching and refreshing. The users could subscribe the URLs he want s and the documents of the subscribed URLs will be cached and automatically refreshed by the proxy system. A private space is reserved to store the subscribed pages to avoid being flushed out in the cache replacement. Thus, whenever the subscribed pages are requested, they can be got directly from the proxy system. The proxy systen can be used as an ordinary proxy server in addition to supporting prefetching and refreshing. To evaluate the service capacity of our proxy system, we collected statistic data from four running proxy servers to estimate the storage requirement of a typical subscription.*

Keyword: Proxy, Prefetch, Web Latancy

1. Introduction

The World Wide Web has become the most popular application on the Internet. The increasing number of web servers and browsers makes the web traffic overflow the Internet bandwidth. Most users of the web experience significant latency when they are using the web. How to reduce the web latency efficiently has become an important issue in today's Internet environment.

There are three sources of the web latency: the network latency, the speed of the hardware and the HTTP protocol[1,2,3,4]. The network latency is because of the lack of enough bandwidth and the network propagation delay. The network propagation is dependent on the distance between the server and the client and it could not be eliminated. The computer hardware and the network equipment would have influence on the network latency too. The HTTP protocol 1.0 [1] makes one connection for each object. A web page containing many objects would need many connections to complete, this costs a lot of TCP initial delay. The newer HTTP protocol 1.1[2] makes use of persistent connection and pipelining to reduce the network delay caused b the TCP connections[3,4].

Since the propagation delay only depends on the distance between the client and the server, the solution is using cache or mirroring to make the data to be retrieved near by the client. Mirroring is usually used to speed ftp access. Mirroring a server must consider the legality of copyright. For web access, cache approach is generall used. The cache approach can be categorized into three types: the server side, the client side and the proxy server. The server side cache is build on the web server, which eliminates the processing time on the web server to locate frequently accessed objects. It won't reduce the propagation delay. The client side cache is build on most of the modern web browsers, it could efficiently reduce the requests for the same object from the same client. The client side cache sometimes appears in the form of 'server push'. The users of the client subscribe the material s they want, and the server pushes the data back to the client in background. This is normally done by 'client-poll' approach. [7] In case the users want to browse the data, the data is immediatel available on the clients. lthough the client side cache effectively reduces the request for same object on one client, another client can not enjoy the benefit he wants to access the same object. The proxy server is the solution to this problem. It usually locates on the same local area network as the client does. When different clients access the web throu gh the same proxy server, they share the cache. If the cache hit, the access would get much lower latency compared to directl connecting to the web server. Due to the limited space of the proxy server and the number of users on the same proxy, a busy proxy server would do cache replacement frequently. Some studies show that the best hit ratio on a proxy server is about 30% to 50%[6]. Statistics show that the cache hit ratio of a proxy server is normally less than 30%.

Various researches[12,13,15] have focused on how to increase the hit ratio of a proxy server, and the prefetch mechanism is the most usual way. The prefetcher on the proxy server can gather information from different clients and the proxy server. With this information, the prefetcher makes prediction of what will be needed in the near future and prefetches them into to the cache. When the clients makes requests to the proxy server, the requested data is already in the cache. This makes the web access much faster when browsing the web pages.

Increasing the hit ratio of the proxy cache is not appropriate sometimes. Because the most frequently accessed data ma

* This project is funded by National Science Council, Project Number: NSC89-2219-E006-003

not fill the needs for some users. For example, when a doctor tries to fetch some medical data through the proxy, most of the time the cache will miss because the medical data is not frequently accessed. The doctor has to wait for the data to be fetched in to the cache. Even after the medical data enters the cache, it won't stay in the cache for a long time because other frequently used data may kick this medical data out of the cache.

In this paper, we will propose a user oriented proxy system supporting prefetching and refreshing, which can solve the problems described above. With our proxy system, users can subscribe the web objects they are interested in. The proxy system can prefetch the subscribed objects and refresh them automatically. A group of users with similar interests can thus share the web information through the proxy system and enjoy very fast access. In section 2, we will show several researches related to our work. In section 3, we will describe our proxy system in detail. In section 4, we will show some statistics to evaluate the service capacity of our system. In section 5, the result of our system is described. Conclusion is given in section 6.

2. Related Work

In this section, we will show some researches on prefetching.

2.1 Statistical Prefetching

Statistical prefetching[10] tries to analyze the log on the proxy server and prefetch the most frequently accessed pages by a threshold. It is based on the assumption that the most frequently accessed pages in the past will be also accessed frequently in the future. Inappropriate threshold could make statistical prefetching waste a lot of bandwidth. Here is a tradeoff between bandwidth and latency.Further more, the retrieving delay for non-prefetched data may actually increase due to the extra traffic of the prefetching.

2.2 Deterministic Prefetching

Unlike the statistical prefetching which may fetch unwanted data by analyzing the old access log, the deterministic prefetching[11] will only prefetch the pages specified by the user. Because the user knows what pages he will need in the near future, the deterministic prefetch costs almost no extra bandwidth overhead. The disadvantage of the design is that the prefetcher and the user preferred link database is located on the client side, the user's computer needs to keep powered on when doing prefetching. Since the user preferred link database is located in client side, it is not convenient when a user uses more than one computer.

2.3 Interactive Prefetching

The basic idea of interactive prefetching[13] is the links contained in a page is very possible to be viewed in the next. The proxy server parses the content of the requested page and find all the linked pages in it. When the user on the client is viewing the requested page, the prefetcher on the proxy server fetches the objects in those linked pages. This approach could raise the hit ratio of proxy server cache up to 60% but also make the traffic 4.12 times larger than a normal proxy server. The disadvantage is the extra traffic caused by prefetching. Because the interactive prefetcher only considers the links in the requested page and doesn't make use of the user access history log, it may prefetch many unwanted pages. Moreover, the parse of the page contents also induces overhead to the proxy server.

2.4 Top-10 Approach

The Top-10 approach [14] has 2 assumptions: Only the most popular pages are worth to be prefetched, and the amount of pages needed to be prefetched from a server is not equal for all clients. It should depend on how many pages have been requested by the client in a past period. The Top-10 approach needs the cooperation of the client, the proxy and the server. The Top-10 daemon on the server processes the access log to get the Top10, the most popular pages on the server. The prefetching agent on the client analyzes the access history to see if the request page count for any server is over a thresh old. Then the agent fetches the Top-10 pages from that server. The study shows that more than 40% requests could be cached with no more than 10% traffic. The problem of this design is the client, the proxy and the server need to follow the same policy.

2.5 Predictive Prefetching

There are two main components in this predictive prefetching[12]: the prediction daemon and the prefetching engine. The prediction daemon on the server generates the dependency graph of the client access. It will inform the prefetch engine on the client to do prefetching if any page is found to be highly dependent on current requested page.

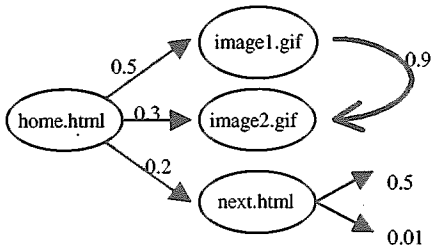

Fig 2-1: An example of the dependency graph
The image1.gif has 50% chance to be accessed when is browsed

By making use of the dependency graph containing the portability information, the prediction engine could make pretty good predictions. The disadvantage of this design is that it needs to modify the HTTP protocol.

2.6 Hybrid Prefetching

When the client requests a page, the hybrid prefetcher [15] tries to find the candidate to prefetch from 3 sources: the links contained in the current requested page, the dependenc graph build from the "referer" HTTP[1] header and the pages located in the same directories as current requested page is. The hybrid prefetcher will prefetch those candidates under 2 constraints: the popularity and the portability of the candidate. The studies shows that the proxy server cache hit rate could be more than 70% with no more than 40% extra network traffic.

2.7 The Squid Web Cache Proxy

The Squid proxy[8] is derived from the ARPA-funded Harvest project. Squid is a high-performance proxy system for web clients and supports the caching of FTP, gopher, and

HTTP data objects. The Squid uses the ICP[9] to do the inter proxy server communication. WithICP, the Squid servers could be arranged to form a hierarchy structure to share the objects in their cache. Due to the great characteristics of the Squid proxy, our User Oriented Proxy system is based on the Squid proxy.

3. The User Oriented Proxy System

3.1 Motivation

A proxy server in general serves many people. To make the use of the cache space efficientl , the proxy server tends to cache those most popular pages. In such circumstance, a professional people trying to retrieve some specific information ma probably get cache miss since the information is not popularly interested. Even more, after the information is cached into the proxy sever, it will be probably flushed out very soon due to the cache replacement.

The World Wide Web has become a very helpful tool in teaching and learning. More and more people use the documents on the web as their reference materials for teachin in their classes. To avoid violating the copyright, the teacher should not copy those data into his own web site. Consequently, when far-away web pages are referenced for teaching on the class, the people would wait for a long time.

The off line browsers are usually used by users to grab the contents of a web site. Using this kind of tools has several disadvantages. The data could not be shared with others. The bandwidth will be waste if man y people fetch the same data and the pages fetched back could not be viewed correctl sometimes. Besides, the storage of a personal computer is generally limited.

All the problems listed above lead us to think about designing a user oriented proxy system which supports prefetching and could refresh the data automatically.

3.2 Design objectives

There are four objectives in the design of our system.

1. The system should be compatible with existing prox servers. Thus the user can use our proxy system with n o modification to their browsers. Because squid is the most popular proxy server in the Internet, we decide to use squid as the base component of our system.
2. The system should have a friendly interface. Users use this interface to subscribe pages by specifying the parameters for prefetching, including the URL needed to be prefetched, the depth of internal links *and external links to be prefetched, the refresh frequency, etc.
3. The system would be used as a data-collecting tool. The system prefetches the data for the user and keeps it up to date automatically. Teachers can use this system to prefetch the referenced materials without violating the copyright. Students can browse the referenced material with very low latency delay by setting this system as thei r proxy.
4. The system would be easily extensible. Since one server has the limitation in disk space and processing power, multiple server configuration is applied. The service capacity will be easily extended by adding more servers to the system.

3.3 System Architecture

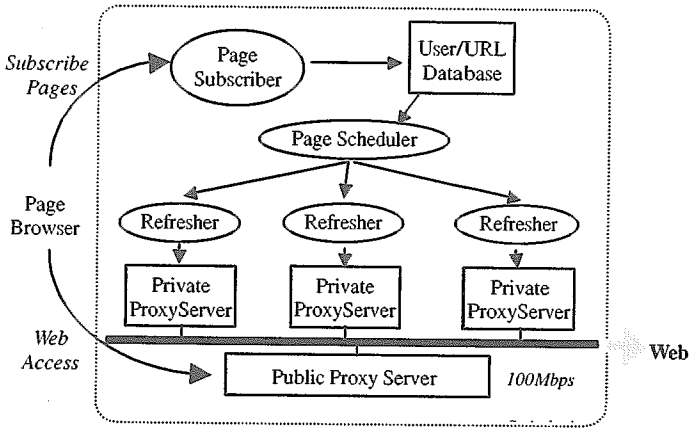

Fig 3-1: System Architecture of the User Oriented Proxy System

The system architecture is shown in figure 3-1 and the main components of our system are described below.

1. Public Proxy: This is the proxy server directly accessed b the client (the browser). It use ICP[9] to communicate with other private proxy servers.
2. Private Proxy: The private proxy servers provide the space to cache the pages subscribed by users. They are not directly accessible to users. They are responsible for passing the fetch requests to the prefetch engine to cache the pages subscribed by users. The private servers are configured as sibling servers of the public proxy. It will answer the ICP query from the public server but won't fetch data for the public proxy.
3. Page Subscriber: It provides the user interface for users to add/delete the pages to be subscribed. The user also uses this interface to check the status of the subscribed pages.(complete, not complete or updated)
4. User/URL Database: It is used to store the user account data, including the password, the subscribed URL and the status of the subscribed URL.
5. Page Scheduler: This is the most important component in our system. The scheduler looks for the URL to be prefetched or refreshed in the database. It then passes the refreshing job to the refreshers on the private proxy server. The scheduler monitors the operating status of each refresher and distributes the job to those active refreshers using a hash value based on the URL string. In case the refresher or the private proxy server crashes, the system can still operate correctly. This gives our system a limited fault tolerant capability.
6. Page Refresher: Each private proxy server in our system has a refresher. The refresher accepts the URL passed b the page scheduler, it then forks a prefetch engine to do the prefetching job. After the job is complete, the refresher updates the prefetch status for the URL in the database.
7. Prefetch Engine: This is the process doing the real prefetching job. It is forked by the page refresher and will report the status back to the refresher.

* The page of a URL and all pages linked from that URL form a URL tree. The links from the same directory of the URL or the sub directories of the URL are internal links. Others are external links.

Proxy Server	Total size of all objects	Number of objects	Number of HTML files	Number of Internal links in all HTML files	Number of external links in all HTML files
Turtle .ee.ncku.edu.tw	3180000	172837	28512	148639	111872
Proxy.ncku.edu.tw	4002100	199711	42911	217649	177959
Proxy2.ncku.edu.tw	15134000	820160	147611	720764	651339
Gate2.ncku.edu.tw	5538700	252267	50531	257327	239908
Server Total	27854800	144, 4975	269565	1344379	1181078

Table 4-1 : statistics on 4 proxy servers in ncku.edu.tw

Proxy server	The mean size of an object (unit:k)	The mean size of a page (unit:k)	The average count of internal links	The average count of external links
Turtle.ee.ncku.edu.tw	18.39884	111.532	5.213208	3.923681
Proxy.ncku.edu.tw	20.03946	93.26513	5.072103	4.147165
Proxy2.ncku.edu.tw	18.4525	102.5262	4.882861	4.412537
Gate2.ncku.edu.tw	21.95571	109.6099	5.092458	4.747739
Pag Average	19.7	104	5.06	4.3

Table 4-2 : Statistics on 4 proxy servers in ncku.edu.tw

3.4 Implementation

We use the squid software for the public and private prox servers. The prefetch engine is written in perl. The page scheduler and page refresher are java applications. The page subscriber is made with CGI form and Java servlet. The database is MySQL. It is accessed through the JDBC interface. There are some issues in our implementation.

1. Prefetching Polic
The system must engage a policy to decide the priority of each URL object to be fetched. In our system, the priority is dynamically changed. Each URL in the database has a weight value. The initial value assigned by the system administrator is depend on the user group. The scheduler will increase the weight of all URLs everyday, then it passes those URL whose weight >=100 to the refreshers on the private proxy servers. The weight value of the new added URLs will be set to 100 to make them be prefetched as soon as possible.

2. Load distribution on multiple private proxy server
In order to have sufficient cache space to store the subscribed pages, multiple private proxy servers are set up and cooperate in our system. An algorithm is needed to distribute a document object to one of the private proxy servers. The distribution formula is

Proxy_Server_Number = Hash_Number mod N

Where
N is the total numbers of the private servers
Hash_Number is a hash value calculated based on the URL string
Proxy_Server_Number is the private proxy server where the prefetch job will be

Because the Hash_Number is calculated based on a URL, each page will has a fixed hash value. If the number of the private proxy servers doesn't change, a page will be cached to a fixed server. If the number of the private proxy servers changes, the page would be cached to another server, and the page cached on the original server will be finall expired and deleted automatically by the squid system.

3. Load control on each private server
The page prefetching and refreshing is actually done in each private proxy server. The refresher on each private prox server forks mulitiple child precesses to execute the prefetch engine. The refresher will monitor the system load. If the system load is too high, it won't fork new child process until the load is down.

4. Prefetching size control
Unlike other web grabbing tools which typically use only one parameter to control the depth of the URL tree to fetch, our prefetch engine has the depth control for both internal and external links. Thus the user can specify the space of the web documents to be prefetched more precisely. To control the size of one prefetching, the system limits the maximum value of the depth in prefretching. In order to minimize the impact of the extra traffic caused by the prefetching, The prefetching can be set to start when network is idle.

4. Evaluation of storage requirements

The capacity of this service is important. We want to make sure how many subscriptions are affordable in practice. Because the system is used to cache the subscribed URLs, we define the service capacity as the number of URL trees that can be cached when the total cache size is fixed. In other words, we want to figure out what is the size of a typical URL tree. We define the expression of a URL tree as follow

Tree: [URL, depth_of_internal_links, depth_of_external_links]

In the following, we evaluate how much space is needed if a user subscribe a URL tree [URL,5,1]. The evaluations have been done in three cases.

Case 1: The URL tree is a full tree

We have made statistics on 4 proxy servers in our university. There are total 1444975 objects in the caches and the total object size is 27 GB.

In table 4-1, 4-2, it shows that

Proxy Server	Number of HTML files	Number of internal HTML links	Number of internal non-HTML links	Number of external HTML links	Number of external non-HTML links
turtle.ee.ncku.edu.tw	24919	110486	23762	63893	30381
proxy.ncku.edu.tw	42235	138589	36294	135214	33417
proxy2.ncku.edu.tw	130205	446607	96079	410781	91286
gate2.ncku.edu.tw	47958	177451	34677	168867	30589

Table 4-3 : statistics on 4 proxy servers in ncku.edu.tw

Proxy Server	The average count of internal HTML links	The average count of internal non-HTML links	The average count of external HTML links	The average count of external non-HTML links
Turtle.ee.ncku.edu.tw	4.433806	0.95357	2.564027	1.21919
Proxy.ncku.edu.tw	3.210347	0.859335	3.201468	0.791216
Proxy2.ncku.edu.tw	3.43003	0.737906	3.154879	0.701094
Gate2.ncku.edu.tw	3.700133	0.72307	3.521144	0.637829
Pag Average	I_h =3.693579	I_n = 0.81847	E_h = 3.110379	E_n =0.837332

Table 4-4 : statistics on 4 proxy servers in ncku.edu.tw

Where I_h is the number of internal HTML links inside a page
I_n is the number of internal non-HTML links inside a page (the number of internal leaf links)
E_h is the number of external HTML links insid e a page
E_n is the number of external non-HTML links inside a page (the number of external leaf links)

Proxy server name	The average count of internal HTML links	The average count of internal non-HTML links	The average count of external HTML links	The average count of external Non-HTML links
Turtle.ee.ncku.edu.tw	2.480435	0.795922	1.434412	1.017629
proxy.ncku.edu.tw	1.974842	0.765683	1.926749	0.704988
Proxy2.ncku.edu.tw	1.690504	0.634419	1.554895	0.60277
gate2.ncku.edu.tw	1.973332	0.630034	1.877874	0.555761
Pag Average	I_h =2.029778	I_n = 0.706515	E_h = 1.698483	E_n =0.720287

Table 4-5 : statistics on 4 proxy servers in ncku.edu.tw (duplication removed)

1. The mean object size in the cache is about 20 K. Other research[5] shows the same result.
2. The mean page size is about 100 K.
3. Each page contains 5 internal links and 4 external links.

An I-way full tree with depth = L will has $^{L+1}$-1 nodes. If each node has E external links, then there will be E*($^{L+1}$-1) external links. For a Tree:[URL,L,1], th ere will be

$(I^{L+1}-1) + (I^{L+1}-1)*E = (I^{L+1}-1) * (1+E)$ HTML pages. [formula 4-1]

From table 4-2, it shows each HTML page has 5 internal links and 4 external links. The count of pages in Tree:[URL,5,1] will be

$5^{5+1} * (1+4)$=78120 pages.

From table 4-2, it shows the mean page size is 100k. The size of Tree:[URL,5,1] will be

78210*100k=7.45GB

This result shows the maximum size of a tree:[URL,5,1]. It is different from the size of a normal tree:[URL,5,1], The reason is most URL trees are not full trees. The links inside a page may link to a non-HTML object. There will no child nodes of these non-HTML links.

Case 2: The URL tree is not a full tree.

The node with no further links available is call ed a leaf. The mean size of a leaf is 20k . We make the statistics on the same four proxy servers again to find the number of internal leaf links and external leaf links.

The size of a URL tree Tree:[URL,5,1] is

(number of internal_html_links+number of external_html_links)* 100k +
(number of internal_non-html_links+number of external_non_html_links) * 20k =
$(I_h^{5+1}-1 + (I_h^{5+1}-1)*E_h)*100k + ((I_h^5-1)*I_n + (I_h^{5+1}-1)*E_h)*20k$ [formula 4-2]

In table 4-4, it shows I_h=3.7, I_n=0.8, E_h=3.1, E_n=0.84. With formula 4-2, the size of Tree:[URL,5,1] is 894M bytes.

This size is still much larger than a normal URL tree because we do not take the duplication of the links into account.

Case 3: URL tree is not a full tree with duplication between links.

The links contained in the pages in a URL tree may point to the same HTML files or objects.(ex: an image) We need to consider of the duplication in calculating the size of a UR tree.

We redo the statistics on the same 4 proxy servers with the duplication of links removed.

From table 4-5, it shows $_h$=2, I_n=0.7, E_h=1.69, E_n=0.72. With formula 4-2, the size of Tree:[URL,5,1] will be 19.81MB

For URL trees of 20MB size, a proxy system with 10 GB cache space will be able store 500 trees. If each user subscribes 10 URL trees, the proxy system will be able to serve 50 people. Thus our proxy system is suitable for a laboratory or an office as a data collecting tool.

5. Experimental Result

In our running system, there are 10 subscribed URLs with different fetch depth setting for each, the total size of the UR trees is about 250MB. Each tree occupies about 25 MB. This result is closed to our evaluation in section 4.

The hit rate of our proxy system depends on how often the user accesses the subscribed pages.
For example, if the hit rate an ordinary proxy is 20%, and 10% of the user access is on the subscribe pages, the hit rate of our system will be 0.1*100%+0.9*25%=28%.

6. Conclusion

Various researches have focused on how to increase the cache hit rate of the proxy server by caching the most popular pages. But the most popular pages do not fill the need for all people, especially for the professional. The professional information will be probably flushed out when competing with those popular pages in the proxy cache. In this paper, we propose a user oriented proxy system which supports prefetching and refreshing and give the estimation of the service capacity of our system. When users subscribe pages with our prox system, these subscribed objects will be stored in the private proxy cache and won't be flushed out by ordinary data. In summary, our system has the following characteristics:

1. It is compatible with the existing proxy server, the user could use our system without any difficulty.
2. After the user makes a subscription in our system, the system will prefetch the page and refresh the content periodically to keep the information up to date.
3. This system is suitable for data collecting without violating the copyright because the data is accessed in its original URLs.
4. This system is easy to extend. The cache space could be increased by adding more private proxy servers to get higher service capacity.
5. This system has limited fault tolerant ability in case any of the private proxy server is crash.
6. Due to the large disks pace needed to cache a URL tree, this system is intend to be used for a small group of users as a data collecting tool.

Much future work remains. The control of disk size used b one user could be done by making the system more adaptive when interpreting the link fetching depth. The impact of extra traffic caused by the system needs to be further studied though it could be reduced by prefetching at the time the network traffic is low. Interactive prefetching can also be adopted in case the page subscribed by user is not in the private cache.

Reference

[1] T. Berners-Lee, RFielding & H. Frystyk., *"Hypertext Transfer Protocol -- HTTP/1.0,"* RFC 1945, UC Irvine, May 1996.

[2] R. Fielding, J. GettysJ. Mogul, H. Frystyk, T. Berners-Lee, *"Hypertext Transfer Protocol -- HTTP/1.1,"* RFC 2068, UC Irvine, January 1997.

[3] World Wide Web Consortium. *"Initial HTTP/1.1 Performance Tests,"* http://www.w3.org/pub/WWW/Protocols/HTTP/Performance/Pipeline.html

[4] Binzhang Liu, Ghaleb Abdulla,Tommy Johnson and Edward A.Fox, *"Web Response Time and Proxy Caching"*, Virginia Polytechnic Institute and State University. http://www.cs.vt.edu/~nrg/pubs.html

[5] Ghaleb Abdulla, Edward A.Fox and Marc Abrams, *"Shared User Behavior on the Eorld Wide Web"*. In Association for the Advancement of Computing in Education, 1997. http://www.cs.vt.edu/~chitra/docs/97webnet

[6] Marc Abrams et al. *"Caching Proxies: Limitations and Potentials,"* http://ei.cs.vt.edu/~succeed/WWW4/WWW4.html

[7] "Seminar Paper Survey of World Wide Web Caching", University of British Columbia, Mar 1997

[8] *"Squid Web Proxy Cache"*, the National Laborator for Applied Network Research, http://squid.nlanr.net/Squid/

[9] Duane Wessels, *"Internet Cache Protocol Version 2,"* RFC 2186.

[10] Azer Bestavros. *"Using speculation to reduce server load and service time on the WWW"*, in Proc of CIKM95: The Fourth ACM International Conference on Information and Knowledge Management, Baltimore, Maryland, Nov 1995

[11] Zheng Wang and Jon Crowcroft *"Prefetching in World Wide Web"*, Department of Computer Science, University College London Gower Street , Lodon WC1E 6BT, United Kingdom http://www.cs.ucl.ac.uk/staff/zwang/papers/prefetch.ps.Z

[12] Radhika Malpani, Jacob Lorch, and David Berger, Venkata N. Padmanabhan and Jeffrey C. Mogul, *"Using Predictive Prefetching to Improve WorldWide Web Latency,"* ACM SIGCOMM Computer Communication Review, July 1996, http://daedalus.cs.berkeley.edu/publications/ccr-july96.ps.gz

[13] Ken-ichi Chinen and Suguru Yamaguchi, *"An Interactive Prefetching Proxy Server for Improvement of WWW Latency"*, Nara Institute of Science and Technology, Japan.

[14] Evangelos P. Marktos and Catherine E. Chronaki, "A Top-10 Approach to Prefetching on the Web", In Proc of INET'98, July 1998, http://130.75.2.13/inet98_proc/li/li_2.htm

[15] Yui-Wen Horng, Wen-Jou Lin and Hsing Mei, *"Hybrid Prefetching for WWW Proxy Servers"*, IEEE 1998. http://dlib.computer.org/conferen/icpads/8603/pdf/86030541.pdf