Proceedings of International Conference
on Networking and Multimedia

Approaches to Reducing Object Retrieval Latency in the World Wide Web*

Yo-Feng Weng and Wen-Shyen E. Chen **
Institute of Computer Science
National Chung-Hsing University
Taichung, Taiwan, 40227 ROC
email: echen@cs.nchu.edu.tw

Abstract

The rapid growth of the World Wide Web (WWW) brings along several problems. Among the problems, the most serious the user experienced is the increasing amount of time he/she has to wait for the retrieval of WWW objects due to increased network bandwidth usage and server overload.

In this paper, we propose some approaches to reducing the latency for retrieving WWW objects: persistent connections and a new protocol to access the proxy server. A prototype of the proposed approaches has been implemented and the performance measurement results show significant improvements in terms of network latency and robustness over the original schemes.

1 Introduction

The past few years have witnessed the accelerated growth of the Internet and the proliferation of the information available in the World Wide Web (WWW) [1, 2]. The Hypertext Transfer Protocol (HTTP) [4] is the transport protocol for the WWW. It is designed to be fast, stateless, extensible, and easy to implement for both clients and servers. However, HTTP has some deficiencies: it does not interact well with the TCP, the underlying protocol for the Internet. As a result, HTTP incurs much unnecessary overhead and leaves some room for improvements. In addition, the explosive increase in the number of clients accessing the Web servers and the multimedia information available to the users result in server overload and saturation of the network bandwidth usage. Consequently, the amount of time needed to retrieve WWW objects becomes unbearable.

The HTTP/1.0 protocol was designed to work with different transport protocols. However, when works with the TCP protocol [5], the HTTP protocol will establish a separate TCP connection every time it retrieves a WWW object. Since, the average size of WWW objects is small[6], the lifetime of the TCP connections established by the HTTP protocol is quite short. With the Slow Start mechanism [7] employed by the TCP protocol, the short-lived connection is destined to be inefficient. Persistent connections can be used to better utilize the bandwidth and to reduce the latency incurred.

Another mechanism to reduce the latency is the use of proxy cache [8, 9]. With a proxy, the internet client will contact a proxy server for the requested objects. If the objects can be found in the proxy, no connections need to be established to the remote WWW server. However, the proxy also introduces a single point of failure and a performance bottleneck. Hierarchical proxy servers structure [10] has been proposed to address the problem of overloaded server but failed to resolve the robustness problem. We devise a proxy-query protocol to address the fault-tolerant and load-balancing issues of the structure. Combined with the persistent connection approach, we think the mechanisms can effectively reduce the latency in retrieving the WWW objects.

The rest of this paper is organized as follows. An overview of the problems is introduced in Section 2. The persistent-connection enhancement to the HTTP/1.0 is discussed in Section 3. The Proxy Query Protocol is illustrated in Section 4. Section 5 shows some performance measurement results of our approaches. Conclusion remarks and future research directions are given in Section 6.

* This research was supported in part by the National Science Council of the Republic of China under contract No. NSC85-2213-E-005-012.

** Corresponding Author.

2 Overview

2. 1 HyperText Transfer Protocol

HyperText Transfer Protocol (HTTP) [4] is a transfer protocol used by the WWW to retrieve distributed hypermedia objects. HTTP has the advantages of being fast, stateless, extensible, and easy to implement for both clients and servers. It provides the user a consistent view of the information available in the network and hides the complexity of communication.

The relationship between the HTTP and the TCP is illustrated in Fig. 1. (Note that we use "internet client" and "browser" interchangeably in this paper.) However, from our initial observation and reports in the literature [12], certain design features of HTTP interact badly with TCP, causing. performance degradation: Latency problems are caused by opening a single connection per request, through connection setup and Slow-Start costs. Further avoidable latency is incurred due to the fact that the protocol only returns a single object per request.

As can be seen in Fig. 2, for each WWW object the HTTP client retrieves, a TCP connection has to be established. Since the three-way handshaking is used [7], the connection setup time takes a round-trip time (RTT) between the client and the server. In the WWW environment, the RTT is significant when compared to the time spent in transmitting the WWW object, which is normally small.

In addition, TCP employs a Slow Start congestion control mechanism [7]: Associated with the sender, there is a congestion windows which indicates the packets that are sent to but have not been acknowledged by the server. The congestion window size is initialized to one (1) when the connection is first established. It would then be incremented by one every time an acknowledgment is received, until a packet is lost (acknowledgment not received) or a preset value is reached. This mechanism has little effect for transferring long files but has a strong impact for the bandwidth utilization for transferring small files, such as WWW objects, since the congestion windows have not fully opened before the connection is terminated.

2.2 Proxy and Proxy Cache

The proxy [8, 9] was originally designed for the communication between the users inside a firewall and the outside world. A typical proxy locates at the firewall host and accepts the requests from the users, delivers the requests to the servers that provide the services. After receiving the responses, the proxy converts the responses into proper formats and sends them to the requesting clients. The function of a proxy is illustrated in Fig. 3.

By adding some disk spaces to the proxy, the objects retrieved from the remote servers can be stored in the disk "cache". The future requests from other clients for the same objects can then be satisfied by the proxy server if the objects are available in the proxy cache, as shown in Fig. 4.

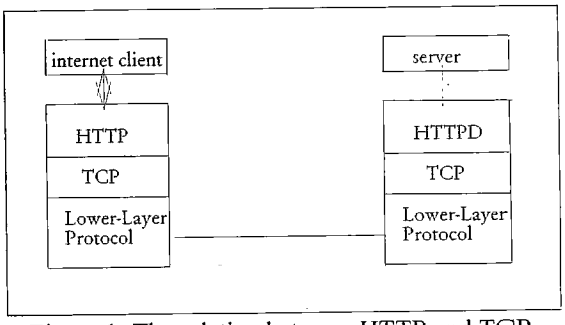

Figure 1: The relation between HTTP and TCP.

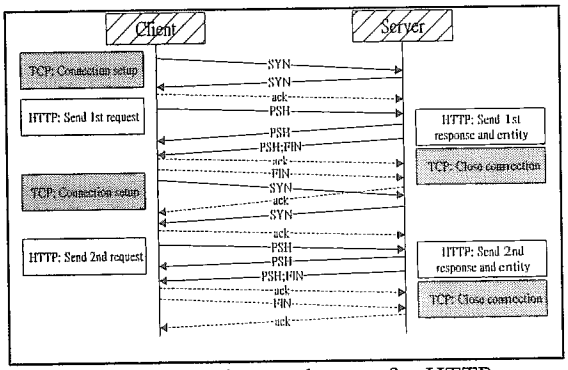

Figure 2: The TCP data exchanges for HTTP request and responses.

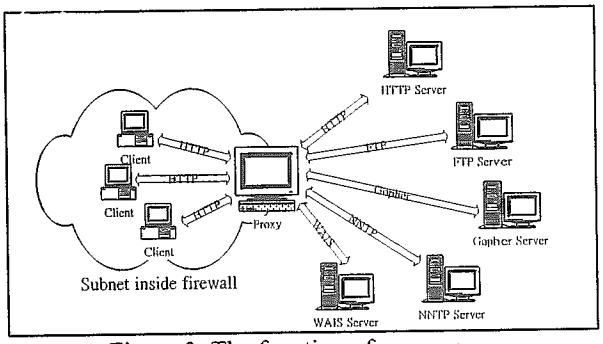

Figure 3. The function of a proxy.

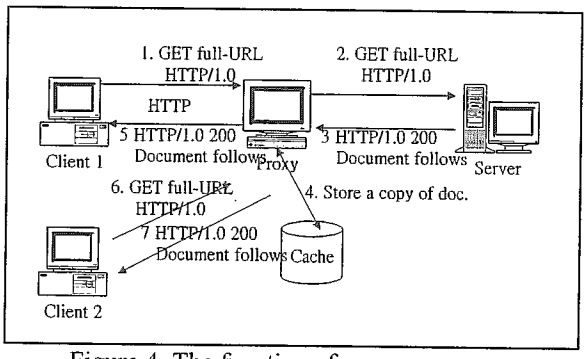

Figure 4. The function of a proxy server.

3 Persistent Connection Enhancement to the HTTP/1.0

In this section, we will present the persistent connection enhancement to the HTTP/1.0 and discuss the problems and solutions while this approach is implmented. As the bases for implementing our approach, we use NCSA Mosaic Version 2.6 as the WWW client, NCSA httpd Version 1.42 as the WWW server, and CERN httpd Version 3.0 as the proxy server. The selections was based on their acceptance and availability of the source code.

3.1 Persistent HTTP Connection

As discussed in Section 2, one of the major reasons that cause the inefficiency of the HTTP protocol is the need to establish a TCP connection for each of the WWW objects to be retrieved, even though the objects are located at the same WWW server. An obvious solution would be to maintain a persistent connection between the client and the server for a reasonable long period of time until it is no longer needed. As a result, when the internet client requests additional WWW objects, such as inline images, from the same server, no further connections need to be established. The persistent connection can be used for transferring other HTML files requested from the server and thus can reduce the effects of slow-start mechanism of the underlying TCP protocol and the waste of the system resources by TCP connections being put into the TIME_WAIT state [7]. Fig. 5 shows the TCP packets exchanged for HTTP with persistent connections. Note that the shaded boxes indicate either a TCP connection set-up or termination. As can been seen, the time to terminate the first connection and that to establish the second connection to the same server can be saved. The saving can be significant if the time to transmit the requested WWW object is small.

A related work about persistent connection that was developed independently the work in the paper can be found in [13, 14].

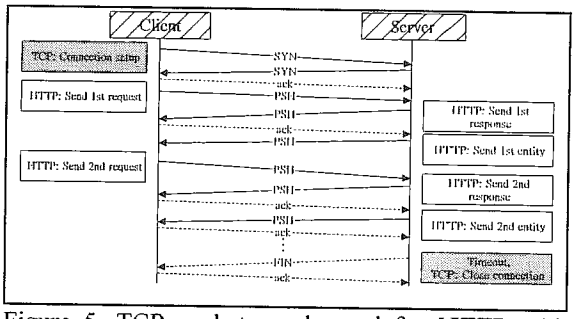

Figure 5. TCP packets exchanged for HTTP with persistent connection.

3.2 Implementation Issues

Client Side Modification

On the internet client side, we use a linked list to record the WWW server addresses and the port numbers that the client has connections with. When a request for a WWW object arrives, the client first check the linked list to see if a connection has been set up to that server. If yes, then use the corresponding socket number to communicate with the server; if not, then set up a new connection to that server and record the WWW server address and the socket number of the connection in the linked list. After the request is fulfilled, the connection is not terminated and can be reused for the future requests.

Server Side Modification

The WWW server we selected (NCSA httpd 1.4.2) has two execution modes. In the first execution mode, when receive a request, the master process will fork a child process for processing the request. In this case, the persistent connection can be maintained by not terminating the connection and the child process created. Thus the requests from the same client can be processed without incurring more overhead. The second operation mode of the NCSA httpd is created to reduce the clients' waiting time by "preforking" several child processes. When the master process receives a request, it passes the socket number of the corresponding connection to a selected child process via interprocess

communication, and thus saves the time to "fork" a new child process. After fulfilling the request, the child process goes back to the preforked child process pool. In this case, our modification is to let the child process keep accepting the requests from the same connection after finishing serving the previous request, until a predetermined idle period is reached.

A major concern for these modifications is that the number of connections a server has to maintain will become unmanageable. In our approach, we have an "idle timer" associated with each connection. The idle timer will be activated after a request receives its services. It will be reset every time a new request is received. The server will terminate the connection only when its corresponding idle timer expires.

3.3 Backward Compatibility with the Original Internet Client/Server

A "Connection" header is added to the HTTP entity header as a token to indicate if the entity supports persistent connections. When an internet client sends a request for persistent connection (by adding the Connection header to the Entity header) to the modified WWW server, the server also adds the Connection header in its response, notifying the client that the persistent connection will be provided. The original server does not understand the newly added header and will ignore it as specified in the HTTP protocol. If the client does not find the Connection header in the response, it knows that the server does not support persistent connection and will behave accordingly. Therefore, the backward compatibility can be achieved.

However, if a unmodified proxy server sits between the internet client and server, the aforementioned mechanism cannot function properly. This is because according to the specification, the proxy will deliver the unrecognized header to the server. Consequently, the newly added Connection header will be delivered between the client and server, without any change. Since the client and server have no way of knowing if a proxy server is located in between them, they would falsely assume that the persistent connection is supported, while in fact the proxy server simply delivers whatever it receives. In this case, the proxy server will wait indefinitely for the server to terminate the connection, which will not happen, or until the timer expires. During this period, the proxy server will not be able to accept new requests from the client.

4 Proxy Query Protocol

4.1 Overview

To relieve the problem of server overload and network congestion, considerable effort has been spent to investigate the method of providing cache for WWW objects. Our work only involves client side solution, which uses caching proxy that accepts requests from one or more clients and caches objects on the clients' behalf.

Many client side caching server have been developed recently. The most widely used in CERN's httpd 3.0. All clients using the proxy can have the benefit of a shared cache. Although it can effectively reduce the server overload problem, it also introduces single point of failure and becomes a performance bottleneck. The structure of such a proxy server is illustrated in Fig. 6, where the proxy servers P1 and P1 serve C1 and C2, C3 and C4, respectively.

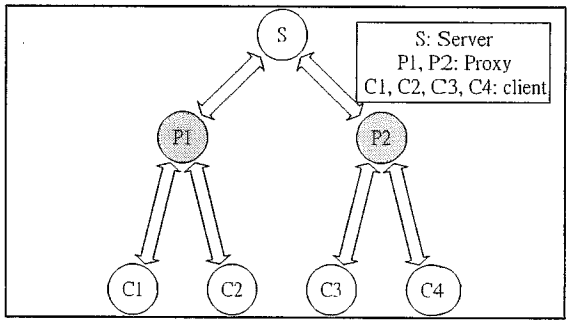

Figure 6. Single level of proxy server.

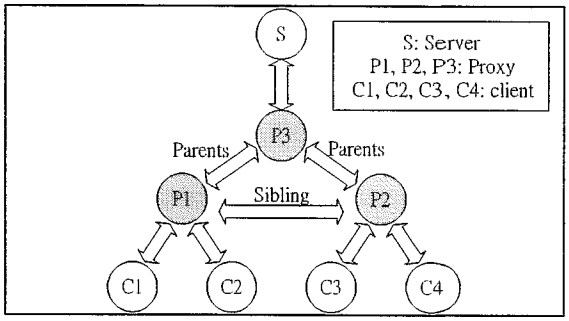

Figure 7. Hierarchical proxy servers.

Another caching server system developed independently with the work done for this paper is the Harvest cache [10]. Developed at the University of Colorado and the University of Southern California, the Harvest cache is a proxy designed to operate in concert with other instances

of itself. These servers are typically configured as a tree, with each server considering a certain set of other servers as parents and certain others as siblings, as shown in Fig. 7. When a server receives a request that it cannot fulfill, it will consult with its parents and siblings to see if they have the requested object cached. One disadvantage of the Harvest cache approach is that it uses unicast to communicate with these parents and sibling servers and the hierarchy does not provide robustness to the proxy structure.

Related Work

Another caching server system, developed concurrently with the work done in this paper is the Cooperating Cache Server scheme in [19]. In this work, a set of cooperating proxy servers are available for clients to use. For each request, the client randomly picks a proxy server ("master" proxy) form the list and sends its request to it. If the master has the requested objects, it returns to object to the client. Otherwise, it multicasts the requests to other proxy servers in the set. If no reply is received within a predetermined time interval, it contacts the server specified in the URL, as it normally does. If any of the other cache servers has the object, the proxy server informs the master proxy, which in turn redirect the clients to the specific proxy server. The client then makes a new request for the object to the proxy server. The protocol is illustrated in Fig. 8.

However, the approach requires that the proxy server supports IP multicast, which is not a reasonable assumption. In addition, the randomly selected proxy server (the master server) might be busy or not functioning properly. In this case, the request cannot be handled efficiently. Furthermore, it requires real TCP connection to look up the requested object in the proxy servers. The latency incurred might not be justified.

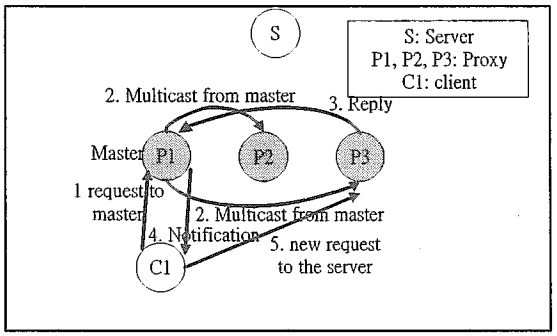

Figure 8. Cooperating Proxy Servers.

4.2 Proxy Query Protocol

To address the robustness and load-balancing issues in the aforementioned proxy servers, we propose to use a "Proxy Query Protocol" to allow the client to use multiple proxy servers. With the protocol, the client will issue a query to each of the proxy servers specified in a configuration file to see if the requested HTML object is cached in one of them. (No query will be sent for the inline images, as will be explained in the following.) The query will be delivered to the proxy servers through UDP, which does not set up connections to transmit data as does in TCP.

A timer will be set when the query is sent. If no response is received before all the timers expire, the client will bypass the proxy and contact the WWW server specified in the URL directly. This is to increase the robustness seen by the client when the proxy servers are all overloaded or are not working at all. If the answers all indicate that the proxy servers do not have the requested objects, the first proxy responded to the query will be selected as the *target proxy* and the normal proxy service will be provided by that proxy server to the client. The first server that responds a "hit" will become the *target proxy* and a regular request will be sent to that proxy. Note that the proxy query protocol needs to be applied for the HTML object only. The client will send the requests for the inline images included in the HTML object to the target proxy to ensure that the related WWW objects will reside in the same proxy server. In addition, the load-balancing of the proxy servers can be achieved as the proxy servers with lower loads will response faster to the client. The protocol is illustrated in Fig. 9.

An alternative to the above mentioned protocol is to use IP multicast to allow the client to multicast the queries to the proxy servers. This approach was rejected because it might not be acceptable to require IP multicast in all TCP/IP protocol stack implementations used by the client.

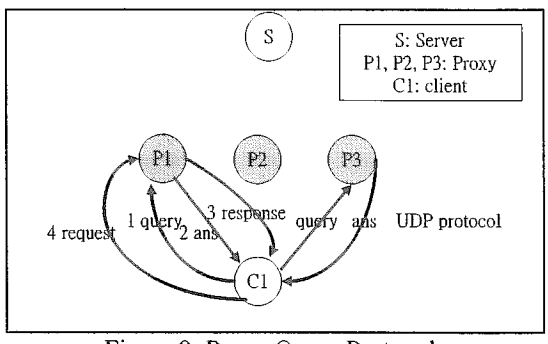

Figure 9. Proxy Query Protocol.

4.3 Implementation Issues

To implement the proxy query protocol, a configuration file called proxy.conf is added on the client side to specify the proxy servers to use. UDP protocol is used in delivering the queries and answers to avoid the high overhead of TCP connections. On the proxy server side, a process is created to handle queries from the clients. The data exchanged between the client and the server has the formats as shown in Figure 10. The format for the client query has two field: A timestamp for the time that the query was issued, and a URL filed indicating the URL it needs. The ranswer to the query also has two fields: The timestamp that was sent in the original query so that the client can correlate the response to the query it sent, and a response indicating "hit" or "miss".

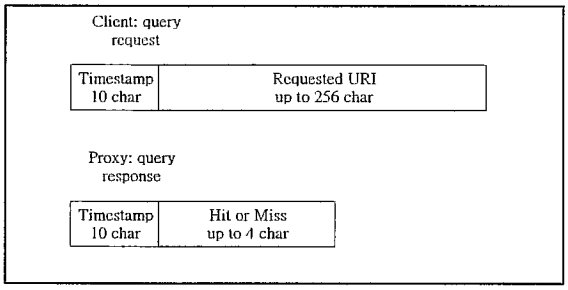

Figure 10. The formats of the query and answer.

We use UDP unicast in delivering the queries and answers. This is because it is unreasonable to require all the protocol stacks in the clients and proxy servers to support muticast. In addition, to reduce the time in collecting all the answers from the proxy servers, we also introduce a limit to the number of the proxy servers that a client can send the query to.

5 Performance Analysis

In this section, we will use a prototype implementation for performance measurements of the aforementioned persistent-connection mechanism and proxy query protocol. The experiments were carried our between 4:00AM and 7:00AM, when the network traffic volume was shown to be the lowest. In addition, the time measured is for data transmission only and does not include the time for displaying the retrieved objects.

5.1 Measurements for Persistent Connection

In the experiments conducted for persistent connection, the workstations for the modified client is a SUN Sparc 20 running Solaris 2.4 OS. The workstation for the modified WWW server is a SUN Sparc 10 running Sun OS 4.1.3 OS. The client is located at the Institute of Computer Science, National Chung-Hsing University, in Taichung, Taiwan. The servers are located in Department of Computer Science and Information Engineering, National Taiwan University, in Taipei, Taiwan. The two universities are the network centers for Taiwan Academic Network (TANET).

A "sample" HTML file with 10 inline images is also constructed for the experiment. The size of the WWW objects has a big impact to the possible performance improvement of the persistent connection mechanism: If the size of the WWW object is large, then the connection set-up/ termination time is small, compared to the total time in retrieving the object. Therefore, selecting the "right" size for the sample documents and objects us important. We analyze the files in the WWW proxy server cache located at the Computer Center of the National Chung-Hsing University and obtain the statistic about the files as listed in Table 1.

We select the median of the distribution, 2.8Kbytes as the standard WWW object size for the experiments. This size is larger than the 2.0Kbyte object size as mentioned in [6]. However, in [6], Bray also notices the trend of increasing size for the WWW objects. As a result, we think the size we select is in accordance to the result presented in [6].

Table 1: The statistics of the files in the proxy cache of NCHU. (Size in Byte)

Number of Files	8,358
Total size of all files	107,608,200
number for files with size 0	8
max. file size	2,930,811
min. file size	183
Average file size (excluding size 0)	12,887
Median of the file size (excluding size 0)	2,818

The measurement results are shown in Fig. 11. As illustrated in the figure, the network latency incurred by the modified client and server is substantially smaller (about 55% less) than that of the unmodified counterparts. The results indicates that persistent connection can effectively reduce the latency in retrieving WWW objects.

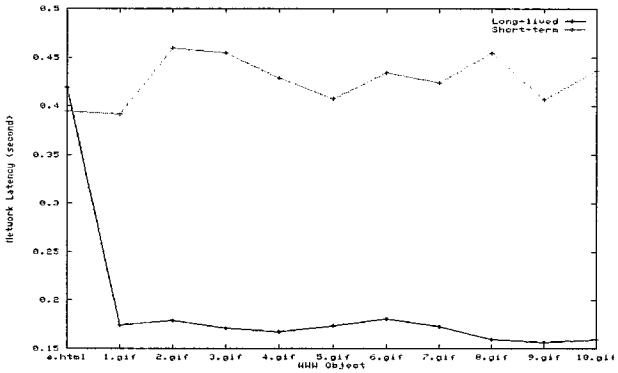

Figure 11. The network latencies comparison of the persistent connection and unmodified versions.

5.2 Measurements for Proxy Query Protocol

In the experiments for Proxy Query Protocol, the workstations we used for the modified clients are SUN Sparc 20 running Solaris 2.4 OS. The workstations for the modified proxy servers are SUN Sparc 2 running Sun OS 4.1.3 OS. The client and proxy servers are connected by a 10Mbps Ethernet. The scenarios we are interested in are single level proxy, two-level hierarchical proxy, and proxies using proxy query protocol. (The configurations for the three scenarios are as shown in Figs. 6, 7, and 9, respectively.) The eleven sample WWW objects used for this experiment are listed in Table 2. For the sake of simplicity, we also have the clients retrieve objects in a sequential order (C1, C2, C3, and then C4). In addition, a client will wait for an object to be retrieved in full before it issues its request for the same object..

Table 2. The sample WWW objects to retrieve.

URLs
http://fakeindy.linkease.com.tw/twnet/www.html
http://www.ntu.edu.tw
http://www.ntnu.edu.tw
http://www.nccu.edu.tw
http://sparc14.ncu.edu.tw
http://www.nthu.edu.tw
http://www.nctu.edu.tw
http://www.nchu.edu.tw
http://www.ccu.edu.tw
http://www.ncku.edu.tw
http://www.nsysu.edu.tw

Fig. 12 shows the performance measurement results of the scenario of a single level proxy. With the experiments we conduct, Clients 1 and 3's requests result in "misses' in the proxy servers and the proxies have to retrieve the objects from the servers. Clients 2 and 4's requests can be fulfilled by the proxy servers. The average retrieval times of the eleven objects for the clients 1, 2, 3, and 4 are 4.53, 1.57, 4.8, and 1.64 seconds, respectively.

Fig. 13 shows the performance measurement result of the scenario of a two-level hierarchical proxy server. In this scenario, Clients 1 and 3 have to go through 2 proxies to retrieve the objects, while requests from Client 2 and 4 can be fulfilled by the first-level proxies. The average retrieval times for Clients 1, 2, 3, and 4 are 6.15, 1.66, 3.81 and 1.49 seconds, respectively.

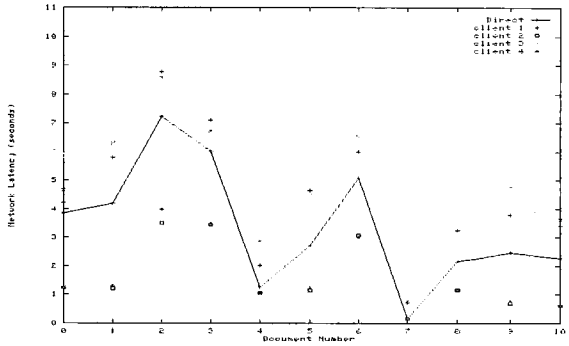

Figure 12. Latency of the single-level proxy server.

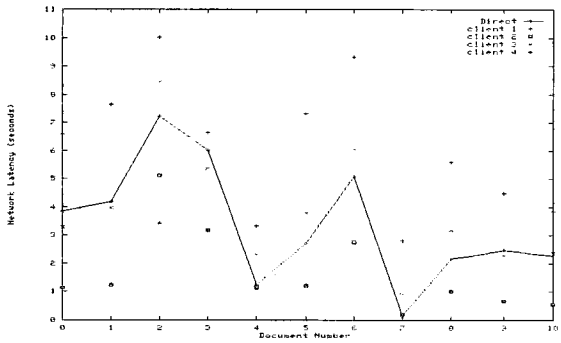

Figure 13. Latency of the two-level hierarchical proxy servers.

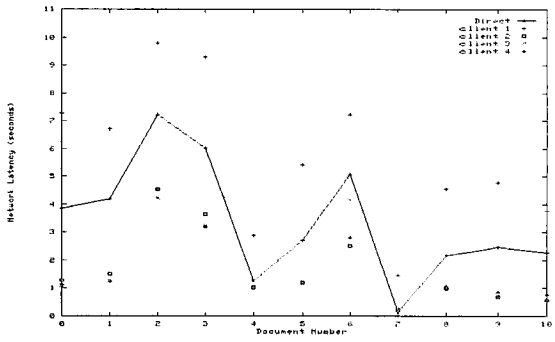

Figure 14. Latency of the proxies with proxy query protocol.

With the proxy query protocol, Clients 2, 3, and 4 all issues their queries to proxies and will

receive a cache hit answer as the objects have been retrieved by Client 1 before. As a result, only client 1 has a higher average retrieval time for the objects. The measurement results are shown in Fig. 14.

6 Conclusion

As a result of growing user population and traffic volume, the WWW users have experienced increased latency while retrieving WWW objects. This is caused by deficiency in the interaction between the HTTP 1.0 protocol and the underlying TCP protocol, server overload, and network congestion. In this paper, we have proposed to use persistent connection for better interactions between HTTP and TCP, and a proxy query protocol to enhance the performance and robustness of proxy servers to reduce WWW server overload and the traffic volume flowing on the Internet. We have also implemented the approaches and performed measurements for the modified client, proxy and WWW server. The results show that the approaches can effectively reduce the latency of WWW object retrievals.

However, the combined effect of the persistent connection and the proxy query protocol remains to be analyzed. In addition, a theoretical analysis of the latency incurred by different proxy caches can provide a better insight about the selections of the parameters (the number of proxy servers in the server set, the orders for accessing the proxies, etc.) for the proxy query protocol. Another possible enhancement might be to incorporate the proxy query protocol into a hierarchical proxy structure (by extending the single proxy in a level of the hierarchy to distributed, shared proxies). These can be the topics for future researches.

References

[1] T. Berners-Lee, R. Cailliuau, J.-F. Groff, and B. Pollermann, "World-Wide Web: The Information Universe," *Electronic Networking: Research, Applications and Policy,* Vol. 1, No. 2, Meckler, Westport, CT., Spring 1992.

[2] R. J. Vetter , C. Spell, and C. Ward, "Mosaic and the World-Wide Web," *IEEE Computer,* pp. 49-57, October 1994.

[3] "NSFNET Traffic Distribution Highlights," Available from Anonymous FTP nic.merit.edu, December 1994.

[4] T. Berners-Lee, R. Fielding and H. Frystyk, "Hypertext Transfer Protocol - HTTP/1.0," Internet Draft, IETF, August 1995.

[5] J. Postel, "Transmission Control Protocol," RFC 793, Network Information Center, SRI International, September 1981.

[6] T. Bray, "Measuring the Web," In *Proc.of the Fifth International World Wide Web Conference*, Paris, May 1996.

[7] W. R. .Stevens, *TCP/IP Illustrated, Volume 1*, Addison-Wesley, 1994.

[8] A. Luotonen and K. Altis, "World-Wide Web Proxies," April 1994, http://www.w3.org/hypertext/WWW/Proxies.

[9] S. Glassman, "A Caching Relay for the World Wide Web," In *Proc. of the First International World Wide Web Conference*, pp. 69-76, Geneva, Switzerland, May 1994.

[10] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A Hierarchical Internet Object Cache," Technical Report, USC/UCB, 1995.

[11] T. Berners-Lee, "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as Used in the World-Wide Web," RFC 1630, CERN, June 1994.

[12] Simon E. Spero, "Analysis of HTTP Performance Problems," July 1994. http://sunsite.unc.edu/mdma-realease/http-prob.html.

[13] J. C. Mogul, "The Case for Persistent-Connection HTTP," In *Proc. SIGCOMM '95*, pp. 299-313, Cambridge, Mass., August 1995.

[14] V. N. Padmanabhan and J. C. Mogul. "Improving HTTP Latency," In *Proc. of the Second International World Wide Web Conference*, pp. 995-1005, Chicago, IL, October 1994.

[15] S. Spero, "Next Generation Hypertext Transfer Protocol," Internet Draft, IETF, March 1995.

[16] NCSA, "Common Gateway Interface," 1994, http://hoohoo.ncsa.uiuc.edu/docs/cgi/overview.html.

[17] NCSA, "Server Side Includes (SSI)," 1994, http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html.

[18] R. Malpani, J. Lorch, and D. Berger, " Making World Wide Web Caching Servers Cooperate," in *Proc. of the Fourth International World Wide Web Conference*, Boston, December 1995.