Vol. 39 No. 6 Transactions of Information Processing Society of Japan June 1998

*Regular Paper*

# An Evaluation of Connection Caching on the World Wide Web

NORITAKA OSAWA,† TARO TAKEI,† KAZUHISA YANAGAWA† and TOSHITSUGU YUBA†

This paper proposes and evaluates connection caching to reduce the overhead of accessing pages on the World Wide Web (WWW). Connection caching means that a WWW server or a WWW proxy does not release its connection with a client or a peer but retains it, and uses it again. The retained connection is cached and it is used for future access. Connection caching reduces network traffic, server load and access latency. Connection caching in this study allows multiple connections to be established for the same host in order not to increase access latency, and it is global caching which does not distinguish requesting hosts. This paper shows that the hit rate of connection caching for WWW accesses is high and connection caching is effective on the WWW. For example, the caching of 16 connections gives hit rates of more than 81% at both a proxy and a server. This evaluation is based on actual logs from a server and a proxy at a university. The logs are records of 350 days. The log from the proxy includes more than 14 million accesses. Our analysis also shows that more than 29% of requests from clients to the proxy share cached connections to servers in the global 16-connection cache. Global connection caching is effective at the proxy.

## 1. Introduction

The World Wide Web (WWW) on the Internet is widely used. A large part of network traffic on the Internet is occupied by WWW accesses. User access latency and the server load have become problem areas. Sometimes users experience long access latency. The main reasons are network congestion and server overload.

A caching technique is effective in solving these problems. Data caching has been used in servers, proxies and clients. By caching the frequently accessed data at sites near a client, network traffic and access latency decrease. Moreover, the WWW server load also decreases because the amount of accesses to the server is reduced by caching. A lot of research on data caching has been performed [1),8),9),14),18)]. We also proposed generational caching schemes for proxy server caches [15),16)].

Not only data caching but also connection caching is possible [13)]. Connection caching means that a WWW server or a WWW proxy does not release its connection with a client or a peer but retains it and uses it again after the transmission is completed. The retained connection is cached and it is used for future access. Since the establishment of a connection creates some network traffic and needs server response, connection caching reduces network traffic, server load and access latency.

This paper proposes and evaluates global connection caching for the WWW servers and proxies. In this study, we will evaluate and discuss the hit rates of the connection cache in a proxy and a server, using access logs from the Information Processing Center (IPC) of the University of Electro-communications (UEC) in Japan. Our study of connection caching is based on logs of 350 days.

We have reported an analysis of connection caching for the WWW [13)]. The analysis assumes that only one connection can be established for each host (server) at most. When multiple accesses to a server are requested simultaneously, access latency may increase if only one connection to the server has to be used. On the other hand, this paper will present an evaluation on condition that multiple connections to a server are permitted.

Our evaluation shows that the caching of 16 connections on a WWW server and a proxy server gives hit rates of more than 81%. This is fairly high. Furthermore, more than 29% of requests from clients to the proxy share connections to servers. Global connection caching allows clients to share connections to servers and enhances the hit rate.

The rest of this paper is organized as follows. Section 2 explains our evaluation method. Section 3 shows the characteristics of logs which are used to evaluate connection caching, and that

† Graduate School of Information Systems, the University of Electro-Communications

the logs are not special but as usual as logs of other studies. Section 4 explains our connection caching scheme. Section 5 discusses our analysis and connection caching. Section 6 describes the differences of this paper from related work. Lastly, concluding remarks summarize our work.

## 2. An Evaluation Method Based on Logs

Our evaluation of connection caching is based on trace-driven simulation. Logs are gathered at a proxy and a server, and they are used as input data to a connection cache simulator. The connection cache simulator takes a cache size as a parameter and simulates the connection cache and outputs a hit rate.

The entity that a Universal Resource Locator (URL)[5] refers to is called a *page* in this paper. In analysis of connection caching, the host part of URL is used to identify the peer. A proxy and a server where logs were gathered and the logs will be explained.

### 2.1 A Proxy and a Server

We describe the proxy, its users and the server at IPC of UEC.

Workstations (WS's) for education at IPC can not communicate directly with sites outside the university. Therefore users of the WS's have to use a proxy. IPC operated the CERN httpd[12] as its proxy when the logs were collected. The proxy is believed to be used by all users of the educational WS's at IPC. The proxy at IPC is also used as a cache by departments and laboratories that do not have their own proxy servers.

NCSA Mosaic[2] was the WWW client on educational WS's at IPC. Mosaic used at IPC did not hold cached data beyond one session. On computers other than the WS's at IPC, other WWW clients are also used in addition to Mosaic. Some clients, such as Netscape Navigator and Internet Explorer, hold cached data beyond one session.

The WWW server☆ at UEC is accessed from inside and outside UEC. It contains an introduction about UEC and has links to departments of UEC.

The relationship among clients, proxies and servers is shown in **Fig. 1**. Client hosts access WWW pages on servers through proxies or directly. Logs from our proxy are used to identify both client host addresses and server host addresses. Logs from our WWW server are used to identify addresses of hosts which access pages on the server.

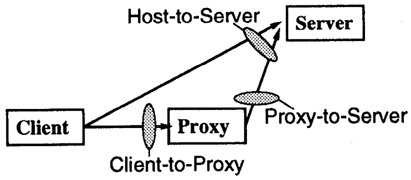



**Fig. 1** Relationship among clients, proxies and servers.

**Table 1** Fields in common logfile format.

| | |
|---|---|
| `remotehost` | The remote host name |
| `rfc931` | The remote logname of the user |
| `authuser` | The authenticated user name |
| `date` | The date and time of the request |
| `request` | The request line exactly as it came from the client |
| `status` | The HTTP status code returned to the client |
| `bytes` | The content-length of the document transferred |

### 2.2 Log Data

An example of one record in a log is shown below.

```
gold.yuba.is.uec.ac.jp - -
 [31/Oct/1995:13:31:32 +0900]
  "GET http://www.yuba.is.uec.ac.jp/~osawa
   HTTP/1.0" 200 1620
```

The common logfile format is as follows. Fields in the format are explained briefly in **Table 1**.

```
remotehost rfc931 authuser [date]
"request" status bytes
```

In a proxy log, `remotehost`☆☆ identifies a client host, and the host part☆☆☆ of URL in `request` specifies a server host. In a server log, `remotehost` identifies a remote client.

`date` represents a timestamp when processing of the request is started. The order of lines represents the order of the completion time of the request. Therefore we can identify the starting order and the finishing order of requests from the logs. The starting order is specified by timestamps. The finishing order is identified by the order of lines because in the above format lines are recorded in a log file. This order information enables us to evaluate connection caching where multiple connections to a server

☆ http://www.uec.ac.jp/

☆☆ `gold.yuba.is.uec.ac.jp` in the example

☆☆☆ `www.yuba.is.uec.ac.jp` in the example are permitted.

Unfortunately, transfer time of pages is not recorded in the logs. Thus precise evaluation of timing or latency is impossible on the basis of our logs alone. However, our logs are sufficient to evaluate the hit rates and the effectiveness of connection caching.

## 3. Characteristics of Log Data

We analyze accesses through Hypertext Transfer Protocol (HTTP) [4]. Not only successful accesses (with a `status` code of 200) but also unsuccessful accesses are used because unsuccessful accesses also need connections. Log data was gathered between 1 PM on October 24, 1995 and 1 PM on October 8, 1996. The length of log period is 350 days. The total number of accesses to the proxy was 14,601,900. That is an average of 41,719.7 access/day. The total number of accesses to the WWW server was 1,371,193. That is an average of 3,917.7 access/day.

**Figures 2** and **3** show ratios of access amounts by hour and by days of the week at the proxy respectively. A peak exists between 3 PM and 4 PM. Weekdays have more accesses than weekends. Distribution of accesses in Figs. 2 and 3 is as expected. There is nothing special about our logs.

Glassman [8] states that the access frequencies of *pages* in a WWW server follow Zipf's law[11]. If Zipf's law holds, there is locality of access. We investigated the access frequencies of *hosts* and the frequencies of *intervals* between accesses to the same host. To our knowledge, this has never been reported before.

Let the number of entities whose access frequency is $f$ be $P(f)$. If Zipf's law holds, $P(f) = M/f^k$ where $M$ and $k$ are constants. $P(f)$ can be plotted as a line between top-left and bottom-right in a log-log graph.

The number of hosts that have the same access frequency is shown in **Fig. 4**. The X-axis is the access frequency of a host. The Y-axis is the number of hosts whose access frequencies are the same.

In Fig. 4, Zipf's law is applicable to the range of lower access frequencies to server hosts from the proxy (`Proxy-to-Server`), from remote hosts to the server (`Host-to-Server`), and from client hosts to the proxy (`Client-to-Proxy`) although there are some differences between the law and the log data.

The host that has the highest access frequency in `Proxy-to-Server` is the educational WWW server host at IPC which has home pages related to the educational computers. Mosaic on an educational WS accesses the home pages at startup time because it does not hold cached data beyond one session.

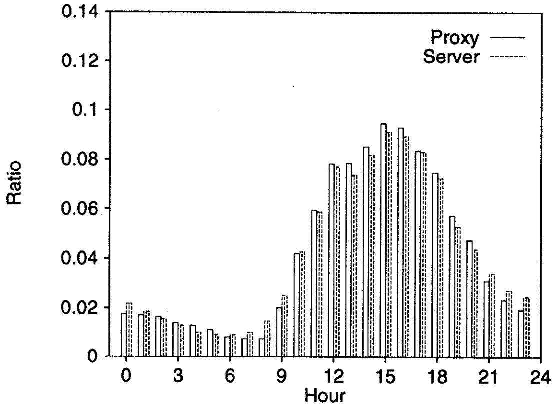



**Fig. 2** Normalized access frequencies by hour at the proxy. Access frequencies are normalized by total number of accesses. `Proxy` and `Server` represent accesses to the proxy and accesses to the server respectively.

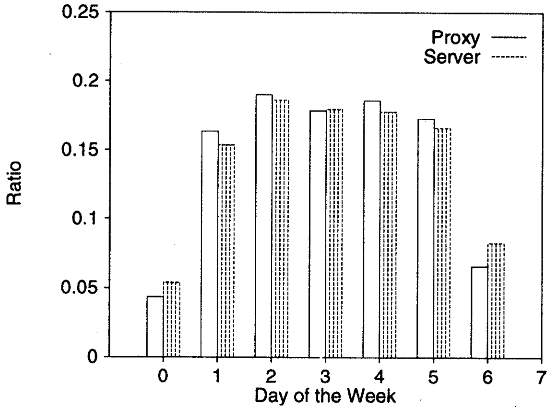



**Fig. 3** Normalized access frequencies by days of the week at the proxy. Access frequencies are normalized by total number of accesses. Zero and 6 in X-axis represent Sunday and Saturday respectively. `Proxy` and `Server` represent accesses to the proxy and accesses to the server respectively.

**Figure 5** shows the distribution of access intervals. It is a log-log graph. The X-axis is the interval between accesses to the same host. An access interval is the interval between an access to a host and the next access to the same host. An access interval represents locality of access. High frequencies of short access intervals exhibit high locality of access. High locality improves the hit rate of a cache. The high hit rate reduces the overhead of accesses. Figure 5 shows the high locality of access to hosts, thus connection caching would be effective. For example, a sum of frequencies of access intervals of between 1 and 10 is 11,115,473 (76.4%) and a sum of frequencies of access intervals that are greater than 10 is 3,438,592 (23.6%) in `Proxy-to-Server`. The total number of frequencies of all access intervals is 14,554,065 and it is less than the total number of accesses because the first occurrences are not counted as valid intervals. In Fig. 5, Zipf's law is applicable to the range of smaller intervals between accesses to the same host. We will show quantitative evaluation on hit rates of connection caching later.

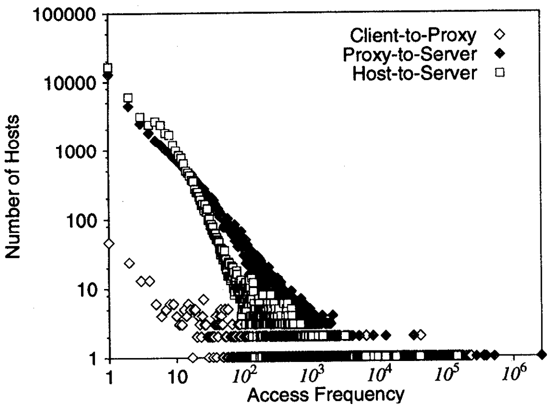



**Fig. 4** The number of hosts that have the same access frequency. `Client-to-Proxy`, `Proxy-to-Server` and `Host-to-Server` represent accesses from clients to the proxy, accesses to servers from the proxy, and accesses from remote hosts to the server, respectively.

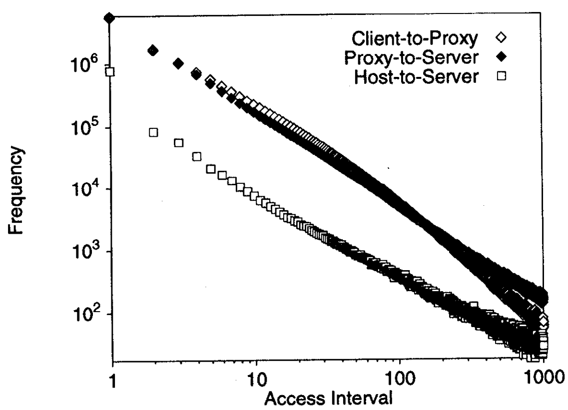



**Fig. 5** Distribution of access intervals between accesses to the same host. `Client-to-Proxy`, `Proxy-to-Server` and `Host-to-Server` represent accesses from clients to the proxy, accesses to servers from the proxy, and accesses from remote hosts to the server, respectively. This shows high locality of access.

## 4. Connection Caching

The evaluation of the caching of connections from clients to the proxy, from the proxy to servers, from clients or proxies to the WWW server is shown. This evaluation is based on logs from the proxy server and the WWW server at IPC in UEC.

**Figures 6** and **7** show internal structures of a connection cache and a connection cache scheme which is assumed in this study, respectively.

In a server and a proxy, all connection requests are accepted. When the processing of a request using a connection is finished, the connection is not released immediately but it is put into a connection cache. Least Recently Used (LRU) algorithm is used as the replacement algorithm of the connection cache. LRU algorithm replaces the least recently used entry with a new entry. LRU algorithm and simplified LRU algorithms are widely used in cache replacement. A connection overflowed from the cache by replacement is released as shown in Fig. 6.

When a client requests a page through the proxy, the proxy searches for a connection to the server of the page as shown in Figs. 6 and

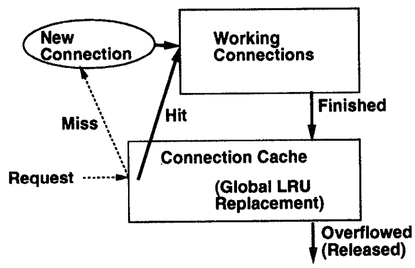



**Fig. 6** Connection cache.

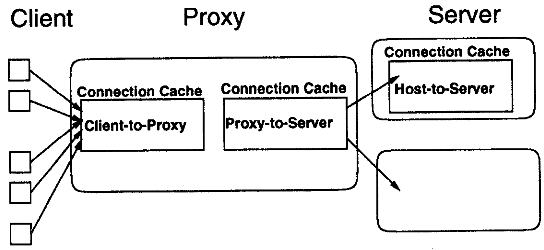



**Fig. 7** Usage scheme of connection caches.

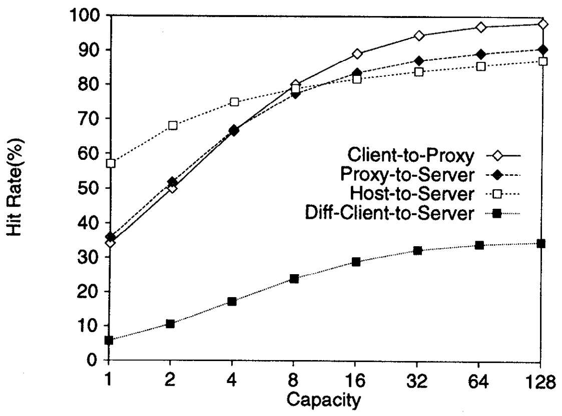



**Fig. 8** Hit rates of connection caching. `Client-to-Proxy`, `Proxy-to-Server` and `Host-to-Server` represent accesses from clients to the proxy, accesses to servers from the proxy, and accesses from remote hosts to the server, respectively. `Diff-Client-to-Server` represents different client accesses to the same server.

7. If the proxy finds an unused connection to the server in the connection cache, the proxy uses it to communicate with the server. Otherwise, the proxy establishes a new connection to the server in order not to increase the communication latency.

Multiple connections to the same host can exist and they will enter the connection cache after they are finished. That is different from a usual cache. Duplicated entries are not allowed in a usual cache, however, multiple connections are permitted in the connection cache.

Hit rates of connection caching are shown in **Fig. 8**. For example, the caching of 16 connections gives hit rates of more than 81% in all cases of accesses from clients to the proxy (`Client-to-Proxy`), accesses to servers from the proxy (`Proxy-to-Server`), and accesses from remote hosts to the server (`Host-to-Server`). It also shows that 128-connection caching gives hit rates of more than 86% in all cases. These results confirm that connection caching is effective.

The used connection caching is global, that is, different clients share cached connections and the replacement algorithm does not choose a victim only among connections accessed by the requesting client but considers all connections in the cache. A connection which is established or used by client `A` can be used by a different client `B`. The use of the connection by client `B` is regarded as a different client access. Incidentally, the use of a newly established connection is never counted as a different client access. `Diff-Client-to-Server` in Fig. 8 represents the ratio of different client accesses to connections. It shows that a connection in the cache is utilized by not only a client but also multiple clients.

This sharing of connections may be influenced by the fact that all clients of the proxy belong to one organization. However, this type of usage of a proxy is common. Thus a global connection cache is usually effective.

## 5. Discussion

A WWW page access usually establishes a connection to a server using TCP/IP protocol and the connection is released after the page data is transmitted. This style of transmission may increase the access latency and the remote server load. However, TCP/IP is used because the protocol is reliable and it passes network fire walls. Transmission error recovery is not a problem when using TCP/IP on WWW systems because TCP/IP retransmits lost and error packets automatically. Network fire walls are widely installed to enhance network security. They usually disallow UDP/IP and unknown protocols but allow TCP/IP. Therefore the use of connection-based TCP/IP is practical as it is difficult to use connection-less protocols presently. It is important to investigate the reduction of overhead where a connection-based protocol is used.

HTTP/1.1 [7] introduced a persistent connection, which lives until the connection is explicitly closed. With this facility, a connection caching scheme is more easily implemented than under the HTTP/1.0 environment, which has no standard for persistent connections. For example, a connection cache can be implemented simply using an array of persistent connections. The connection is closed when replacement occurs. The release of connections should be handled appropriately to operate the connection caching as an effective mechanism. Therefore a definite protocol should be investigated.

The proposed connection caching needs cooperation of peers, that is, clients, proxies and servers. All of them should support connection caching in order to obtain the merits. This is the same as the deployment of the persistent connection facility. If most hosts support connection caching with a reasonable cache size, large parts of connection requests are omit-

**Table 2** Basic statistics of logs: `UEC-IPC` represents the log of the Web server at IPC of UEC. `Usask`, `NASA` and `ClarkNet` represent a log of a campus-wide Web server at the University of Saskatchewan, a log of the Web server at NASA's Kennedy Space Center and a log of the Web server from ClarkNet, a commercial Internet provider in Baltimore, Washington D.C. region respectively. Maximum hit rate means the hit rate where the cache size is unlimited.

| Log | UEC-IPC | Usask | NASA | ClarkNet |
|---|---|---|---|---|
| Amount (request) | 1371193 | 2408625 | 3461612 | 3328632 |
| Period (day) | 350 | 214 | 62 | 14 |
| Average (req./day) | 3917.7 | 11255 | 56748 | 237756 |
| Max. hit rate (%) | 96.5 | 94.0 | 96.5 | 95.5 |

ted. This omission reduces access latency to remote nodes, especially to far nodes, for example, nodes at the opposite side on the earth. It also reduces the traffic of the whole network since packets of connection requests decrease.

Connection caching does not increase the overhead of establishment and release of connections but adds to the overhead of connection management in hosts. Let us assume that the maximum number of connections to be managed in a host (that is, the size of a connection cache) is less than about 1000. The size of connection cache data structures, which are a hash table and LRU list links, must be less than 1M bytes. Since the size is sufficiently small, all of the structures can be managed efficiently in primary memory of current personal computers and workstations. Thus the connection management would not be a performance bottleneck.

Data caching reduces data requests to servers but cannot eliminate the requests. For example, when a client accesses pages in a server which have not ever been accessed, the client must communicate with the server and usually accesses multiple pages from the server. When images and applets embedded in a page are loaded and links are traversed, accesses from the host to the server are performed with locality. Therefore, regardless of data caching, connection caching reduces the latency of connection establishment and release, and the network traffic. In other words, connection caching can reduce the latency and traffic which data caching cannot reduce. Therefore both data caching and connection caching should be employed in order to decrease the access latency and network traffic.

Hit rates of connection caching in various servers are plotted in **Fig. 9**. From the Internet Traffic Archives☆, three logs are used. They are labeled as `Usask`, `NASA` and `ClarkNet` in

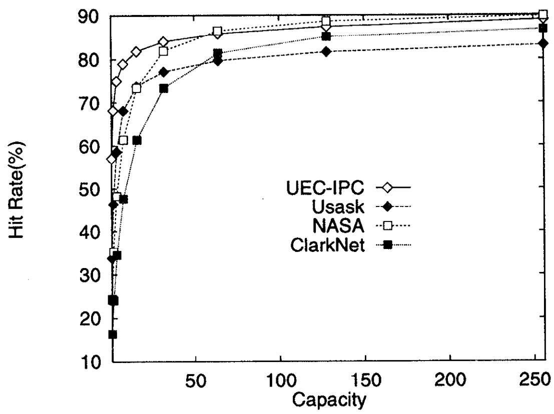



**Fig. 9** Hit rates of connection caching in four servers: `UEC-IPC` represents the log of the Web server at IPC of UEC. `Usask`, `NASA` and `ClarkNet` represent a 7-month log of a campus-wide Web server at the University of Saskatchewan, a two-month log of the Web server at NASA's Kennedy Space Center, and a two-week log of the Web server from ClarkNet, a commercial Internet provider in Baltimore, Washington D.C. region, respectively, from the Internet Traffic Archives.

Fig. 9. `Usask`, `NASA` and `ClarkNet` represent a 7-month log of a campus-wide Web server at the University of Saskatchewan, a two-month log of the Web server at NASA's Kennedy Space Center and a two-week log of the Web server from ClarkNet, a commercial Internet provider in Baltimore, Washington D.C. region, respectively. `UEC-IPC` represents the log of the Web server at IPC of UEC. **Table 2** shows basic statistics of the logs. Please refer to the paper of Arlitt, et al.[3] for more details of the logs. Figure 9 and Table 2 show that 128-connection caching achieves more than 84% of the maximum hit rates in all cases. In brief, connection caching is effective in all of the servers.

☆ http://www.acm.org/sigcomm/ITA (as of December 1997)

## 6. Related Work

A protocol that transmits multiple pages in one connection was proposed in Ref. 17). It uses long-lived connections. Pages that are referenced by the page that the user has accessed are transmitted together. That proposal aims at reducing access latency. It focuses on connections between a client and a server. In other words, it uses a local connection caching technique where a client can use only the retained connections that the client has established. On the other hand, our study uses a global caching technique where any clients can use retained connections no matter which clients established the connections. Our analysis shows that more than 29% of the requests from clients to the proxy share connections in a global 16-connection cache. Moreover our study evaluated the effectiveness of connection caching from clients to the proxy, from the proxy to servers, and from clients to servers quantitatively. Our study is different from Ref. 17).

Jomer [10] describes an algorithm which calculates expiration of a connection by taking into account the current traffic load, the connection cost, and the maintenance cost where OSI connectionless network protocol (CLNP) is supported over a X.25 network. However, multiple connections are not considered in that paper.

Yang, et al. [20],[21] analyze connection caching and its performance using simulation when host grouping is used in a network. They use synthesized traffic and they do not mention the WWW accesses. On the other hand, our study is based on actual logs of the WWW.

Dittia, et al. [6] use connection caching in an ATM Host-Network Interface Chip. The effectiveness of connection caching is not shown quantitatively.

## 7. Concluding Remarks

We have proposed and analyzed connection caching for WWW severs and proxies on the basis of actual logs of 350 days. Our study showed that the connection caching is effective. For example, the caching of 16 connections on a WWW server and a proxy server gives hit rates of more than 81%. Connection caching utilizes the locality of access. Sufficiently large effectiveness is expected when using connection caching. Furthermore, global connection caching is more effective than local connection caching as a means to enhance hit rates and the performance because more than 29% of requests from clients to the proxy share connections to servers in our analysis. We have a plan to investigate and evaluate a definite protocol that uses connection caching. We will also investigate the workload of WWW accesses and model it in order to make the synthetic workload generation possible, like synthetic traces for trace-driven simulation of cache memories[19].

**Acknowledgments**

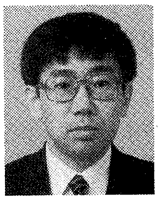
We wish to acknowledge Prof. Shuichi Itoh, Dr. Katushi Kobayashi and other members of the Information Processing Center at the University of Electro-communications for helping us acquire the WWW access logs. We would like to thank Dr. Jim Dumoulin of the Kennedy Space Center, Dr. Stephen Balbach of ClarkNet, Drs. Earl Fogel, Martin Arlitt and Carey Williamson of the University of Saskatchewan, and the Internet Traffic Archives for placing their logs publicly on the Internet Traffic Archives and letting us use the logs.

## References

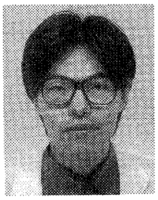
1) Abrams, M., Standridge, C.R., Abdulla, G., Williams, S. and Fox, E.A.: Caching Proxies: Limitations and Potentials, *Proc. Fourth Int'l World Wide Web Conference*, pp.119–133 (1995).
2) Andreessen, M.: NCSA Mosaic Technical Summary, Technical Report, National Center for Supercomputing Applications (1993).
3) Arlitt, M.F. and Williamson, C.L.: Web Server Workload Characterization: The Search for Invariants, *Proc. 1996 ACM SIGMETRICS Conf. on the Measurement and Modeling of Computer Systems*, pp.126–137 (1996).
4) Berners-Lee, T., Fielding, R. and Frystyk, H.: Hypertext Transfer Protocol – HTTP/1.0, Technical Report, HTTP Working Group Internet-Draft (1996).
5) Berners-Lee, T., Masinter, L. and McCahill, M.: Uniform Resource Locators, RFC 1738 (1994).
6) Dittia, Z.D., Jerome, R. Cox, J. and Parulkar, G.M.: Design of the APIC: A High Performance ATM Host-Network Interface Chip, *IEEE INFOCOM '95*, Vol.1, pp.179–187 (1995).
7) Fielding, R., Gettys, J., Mogul, J., Frystyk, H. and Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1, RFC 2068 (1997).
8) Glassman, S.: A Caching relay for the World Wide Web, *Proc. 1st Int'l Conf. on WWW* (also appeared in *Computer Networks and ISDN Systems*), Geneva, Vol.27, pp.165–173

(1994).
9) Ichii, S. and Nakayama, M.: Effectiveness of WWW caching in a campus network, Technical Report, SIG Notes DSM-9505033, IPSJ (1995).
10) Jomer, P.: Connection Caching of Traffic Adaptive Dynamic Virtual Circuits, *Proc. SIGCOMM 89 Symp.: Communications, Architectures and Protocols*, pp.13–24 (1989).
11) Knuth, D.: *Sorting and Searching*, The Art of Computer Programming, Vol.3, Addison-Wesley (1973).
12) Luotonen, A. and Altis, K.: World-Wide Web Proxies, *Proc. 1st Int'l Conf. on WWW* (also appeared in *Computer Networks and ISDN Systems*), Geneva, Vol.27, pp.147–154 (1994).
13) Osawa, N., Yanagawa, K. and Yuba, T.: Connection Caching for WWW Servers and Proxies, *Proc. World Conference of the WWW, Internet, and Intranet* (*WebNet97*), pp.364–369 (1997).
14) Osawa, N., Hayano, F., Yuba, T. and Hakozaki, K.: Evaluation of Replacement Policies in WWW Cache Based on Logs from Proxy Server, Technical Report, SIG Notes DPS 74-33, IPSJ (1996).
15) Osawa, N., Yuba, T. and Hakozaki, K.: Generational Caching Schemes for a WWW Caching Proxy Server, *Proc. High-Performance Computing and Networking* (*HPCN97 Europe*), pp.940–949 (1997).
16) Osawa, N., Yuba, T. and Hakozaki, K.: An Evaluation of Generational Caching Schemes Based on WWW Caching Proxy Server Logs, *Trans. IPSJ*, Vol.39, No.3, pp.760–768 (1998).
17) Padmanabhan, V. and Mogul, J.: Improving HTTP Latency, *Proc. 2nd Int'l WWW Conf.*, pp.995–1005 (1994).
18) Pitkow, J.E. and Recker, M.M.: A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns, *Proc. 2nd Int'l WWW Conf.*, pp.1039–1046 (1994).
19) Thiebaut, D., Wolf, J.L. and Stone, H.S.: Synthetic traces for trace-driven simulation of cache memories, *IEEE Trans. Comput.*, Vol.41, No.4, pp.388–410 (1992).
20) Yang, S.-M. and Cho, Y.: A Performance Study of A Connection Caching Technique, *Proc. IEEE WESCANEX Communications, Power, and Computing Conference - Part 1*, pp.90–94 (1995).
21) Yang, S.-M. and Cho, Y.: Connection Caching Technique with Host Grouping, *Proc. 3rd Int'l Conf. on High Performance Computing*, pp.374–378 (1996).

(Received October 1, 1997)
(Accepted March 6, 1998)

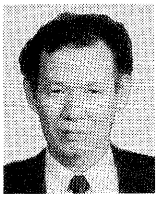


**Noritaka Osawa** received his B.S., M.S. and Ph.D. degrees in information science from the University of Tokyo in 1983, 1985 and 1988, respectively. Since 1993, he has been at the University of Electro-Communications, as a research associate. His current research interests focus on parallel and distributed system software. He is a member of IPSJ, ACM and IEEE-CS.



**Taro Takei** received his B.S. and M.S. degrees from Kanagawa University in 1995 and 1997, respectively. His research interests include high-speed networks and protocols. He is a student member of IPSJ.

**Kazuhisa Yanagawa** received his B.E and M.E. degrees from the University of Electro-Communications in 1996 and 1998, respectively. His research interest is object-oriented programming.



**Toshitsugu Yuba** was born in 1941. He received the B.E. and M.E. degrees in electrical engineering from Kobe University in 1964 and 1966, respectively, and the Ph.D. degree from the University of Tokyo in 1982. In 1966 he joined the Nomura Research Institute. From 1967 to 1993, he was with the Electrotechnical Laboratory (ETL) of the Agency of Industrial Science and Technology, the Ministry of International Trade and Industry. His final position at ETL was the director of Computer Science Division. He moved from ETL to the University of Electro-Communications in 1993. He is currently a professor of the Graduate School of Information Systems. His current interest is parallel and distributed computing in computer science; parallel computation models, parallel/distributed operating systems, parallel compilers, parallel architectures and parallel applications such as parallel discrete event simulation. Dr. Yuba is a member of IEICE, IPSJ, the Japan Society for Software Science and Technology, the Robotics Society of Japan, ACM, and IEEE-CS.