American Journal of Operations Research, 2012, 2, 339-347
http://dx.doi.org/10.4236/ajor.2012.23041 Published Online September 2012 (http://www.SciRP.org/journal/ajor)

Web Acceleration by Prefetching in Extremely Large Latency Network

Fumiaki Nagase[1], Takefumi Hiraguri[2], Kentaro Nishimori[1], Hideo Makino[1]
[1]Graduate School of Science & Technology, Niigata University, Niigata, Japan
[2]Nippon Institute of Technology, Minamisaitama-gun, Japan
Email: nishimori@ie.niigata-u.ac.jp

Received May 29, 2012; revised June 28, 2012; accepted July 11, 2012

ABSTRACT

A scheme for high-speed data transfer via the Internet for Web service in an extremely large delay environment is proposed. With the wide-spread use of Internet services in recent years, WLAN Internet service in high-speed trains has commenced. The system for this is composed of a satellite communication system between the train and the ground station, which is characterized by extremely large latency of several hundred milliseconds due to long propagation latency. High-speed web access is not available to users in a train in such an extremely large latency network system. Thus, a prefetch scheme for performance acceleration of Web services in this environment is proposed. A test-bed system that implements the proposed scheme is implemented and is its performance in this test-bed is evaluated. The proposed scheme is verified to enable high-speed Web access in the extremely large delay environment compared to conventional schemes.

Keywords: Extremely-Large-Latency Network; Satellite Communication; HTTP; Web Prefetching; Prefetching Proxy Server; Information Storage Server

1. Introduction

The World Wide Web (WWW), also known as Web service, is standardized by the World Wide Web Consortium (W3C). WWW is the most popular service among Internet applications. W3C regulates HyperText Markup Language (HTML) [1] and Extensible HyperText Markup Language (XHTML), which are used to achieve Web services. A Web server provides Web contents composed from hypertexts that include multimedia contents, such as pictures, moving videos, and music. Web contents are referred to in a Uniform Resource Identifier (URI). Users (clients) send a request for Web contents to the Web server, and the Web server replies to the request. The request and the Web contents are transferred by the Hypertext Transfer Protocol (HTTP) [2,3] which is an application protocol.

Web service is the most popular Internet service. Wide-spread use of broadband services can enable high-speed communication for Web service. Web service with HTTP [4,5] often uses the Transmission Control Protocol (TCP) as the transmission protocol. A Web service is composed of several tens or hundreds of web contents, each of which is approximately 10 kBytes. Due to the reliability of TCP, extremely large latency between the clients and the Web servers restricts the throughput of a Web service regardless of the network bandwidth. Therefore, under unusual environments or system condition, the extremely large latency has a major influence on the throughput of a Web service, though it doesn't influence usually. As examples of unusual environment, the configuration of high-speed trains is shown in **Figure 1**.

In this system for high-speed trains with satellite communications [6,7], since Clients in the trains connects to WLAN (Wireless Local Area Network) [8] equipment located at in-vehicle, the latency of between Clients and WLAN will not be almost influenced. However, the latency between the train and the ground station becomes much larger than usual since the propagation distance is long.

Clients in a high-speed train can access the Internet via the satellite link between the train and the ground station. The long propagation distance via the satellite link causes

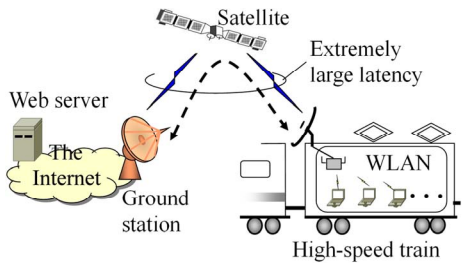

Figure 1. Configuration of target WLAN system.

extremely large latency of about 500 ms between the clients in the train and the Web server. The through- put of a Web service with HTTP is restricted in such an extremely large latency environment [9]. To improve these issues, several TCP acceleration schemes and protocol conversion schemes have been studied to boost Web service performance. These schemes may accelerate throughput by enhancement of TCP, which improves congestion control algorithms.

However, since, the Web service transfer a large amount of small-sized data, these schemes will not achieve high-speed Web service. Moreover, the load of the server may degrade its performance, because Web server is installed complex algorithm. Details of these issues is described in Section 2. Thus, a scheme is proposed that can enable a high throughput Web service in the extremely large latency environment. The effectiveness of the proposed scheme implemented in a test-bed sys- tem in a laboratory test is shown.

This paper is organized as follows. Section 2 describes conventional schemes and issues, and theoretical values are analyzed and derived. Section 3 presents our proposed prefetching scheme for a large latency network, and implementation of the scheme on a test-bed system is described. The evaluation test that was carried out using the test-bed system is described in Section 4. Section 5 concludes this paper.

2. Conventional Schemes and Issues

This section shows the signal sequence of HTTP/TCP that is usually used in Web services. A Web service is composed of many Web contents on Web servers (Web Serv.). In a Web service, a client requests Web contents on a Web server, and the server replies to the request by sending an HTML file. The client receiving the file analyzes it and issues the request written in the file. Thus, one content is delivered from the Web server to the client in reply to one request from the client to the Web server. After the content is transferred, the next request for other content is sent. Many contents are transferred repeatedly as described. HTTP/TCP sequence and its performance with normal access is explained hereafter. After that, the conventional prefetching scheme and its issues are described.

2.1. HTTP/TCP Sequence

This section introduces the throughput of Web contents with HTTP/TCP. A general overview of the transmission sequence of HTTP is shown in **Figure 2(a)**. The HTTP sequence begins after a TCP connection is established through a three-way handshake procedure [10]. The TCP connection is established once, and the session continues until the transfer of all contents ceases. Many contents

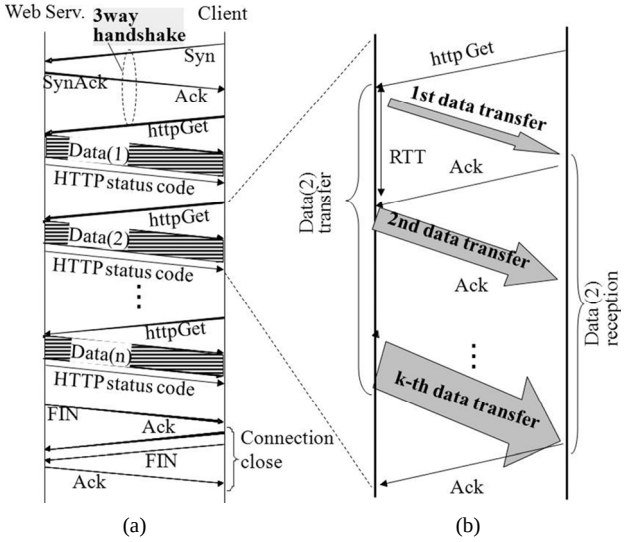

Figure 2. Sequence of HTTP (a) and data transfer procedure (b) of TCP.

are transferred through the TCP session. One content continues to be sent until an HTTP status-code signal is finally sent.

One content is transferred through the TCP session as shown in **Figure 2(b)**. This environment has the latency of round trip time (RTT). In the figure, the content is transferred through a number (k) of data transfers. In data transfer with TCP, window base control and congestion control [11,12] are basically used to achieve reliable data transfer. When receiving an acknowledge (ACK) signal, the Web server (Web Serv.) sends data one of k times, the size of which is expressed as $Ds(k)$, with a smaller window base control value (*Rwin*) and congestion window size at the k-th congestion window size ($cwnd(k)$), as in the equation denoted below.

$$Ds(k) = \min\left(cwnd(k), Rwin\right). \tag{1}$$

Here, *Rwin* is the receive window size advertised from the client to the Web server. The sum of $Ds(k)$ is the size of the content, which is expressed as "*Cs*" in this paper. The congestion window size at the k-th TCP transmission ($cwnd(k)$) is expressed as

$$\begin{aligned} cwnd(k) &= 2^{(k-1)} \times cwnd(1) \\ &\quad (cwnd(k) < ssthresh), \\ cwnd(k) &= 2^{(k-1)} \times cwnd(k-1) + \mathrm{MSS} \\ &\quad (cwnd(k) \geq ssthresh), \end{aligned} \tag{2}$$

where *ssthresh* means the slow start threshold of congestion control by TCP. *MSS* is the maximum segment size, which is 1460~Bytes in Ethernet, for example.

The congestion window algorithm begins in the exponential growth phase initially with a congestion window size. Since the client sends an ACK, this behavior effect-

tively doubles the window size each round trip of the network. Once the *cwnd* reaches the *ssthresh*, TCP goes into congestion avoidance mode, where each ACK increases the *cwnd* by 1 *MSS* for each ACK received. This results in linear increase of the *cwnd*. This behavior continues until the *cwnd* reaches the size of the client's advertised window (*Rwin*). The TCP/IP standard allows for *Rwin* up to 65,535 bytes (= 64 kBytes) in size, which is the maximum value that can be specified in the 16-bit TCP window size field. The maximum value of *Rwin* is accurately expressed as $65{,}535 = 2^{16} - 1$. The maximum TCP throughput (Max*S*) is obtained as follows:

$$\begin{aligned}\mathrm{Max}S &= Rwin/\mathrm{RTT} \\ &= 64\ \mathrm{kBytes} \times 8/0.5 = 1024\left[\mathrm{kbps}\right]\end{aligned} \tag{3}$$

Hence, the TCP throughput is restricted to 1024 kbps in a high latency system, regardless of the maximum *Rwin* value.

2.2. HTTP Performance with Normal Access

This section shows the time for all contents in a Web service to be downloaded (Web load time) and HTTP throughput in a large latency network. Web services are composed of several tens or hundreds of Web contents, which are text, pictures, moving pictures, and so on. The total size of a Web service is generally about several tens of kilobytes to several megabytes. This signifies that the size of one web content is very small, at about 10 kBytes. From the Data(1) transfer to Data(*n*) transfer, shown in **Figure 2(a)**, which expresses each web content as small size. One web service is composed of such a small number n of contents. One web content is transferred by *k* number of data transfers, shown in **Figure 2(b)**. The client issues the request (HTTP get request) for acquiring the content. Since the data transfer *Ds*(*k*) in Equation (1) does not increase to *Rwin*, the server sends small-sized data (*Ds*(*k*)) *k* times in the sending of one content. Thus, the time for one small-sized content transfer increases to RTT × *k*, meaning clients cannot obtain high throughput. One content is transferred in the summed time of *Ds*(*k*). Since the throughput (*S*) is the value of the total transfer data size (*Cs*) divided by the total transfer time (RTT × *k*), *S* can be expressed as follows.

$$S = \frac{Cs}{\mathrm{RTT} \times k} = \frac{1}{\mathrm{RTT} \times k} \times k \sum_{n=1}^{k} Ds(n) \tag{4}$$

The throughput for various transfer data sizes can be derived as shown in **Figure 3**. RTT is assumed to be 500 ms in **Figure 3**. The throughput for 10-kByte content is restricted to less than 50 kbps regardless of the 64-kByte *Rwin*. This value is about 20 times smaller than 1024 kbps, which was obtained in Equation (3). This indicates that high throughput cannot be obtained when sending small sized data with the maximum *Rwin* in a large latency network.

HTTP can transfer contents simultaneously in one TCP session. The time for transferring all contents in the Web service, called Web load time, is expressed as follows.

$$Tl = \frac{\sum_{n=1}^{k} wc(n)}{sm} \tag{5}$$

Here, *Tl* is the web load time, *wc*(*n*) is the transfer time of the *n*-th content, *k* is the total number of Web contents in the Web service, and *sm* is the number of simultaneous connections for transfer. The size of *wc*(*n*) is *Cs*, as denoted in Equation (4).

Since HTTP throughput (*Sh*) is derived from the value of the size of all contents divided by the Web load time, *Sh* is expressed as follows.

$$Sh = \frac{\sum_{n=1}^{5} Cs(n)}{Tl} \tag{6}$$

Here, *Cs*(*n*) is the size of the *n*-th content in the Web service. HTTP throughput derived from Equation (6) is shown in **Figure 4**. In **Figure 4**, it is assumed that the

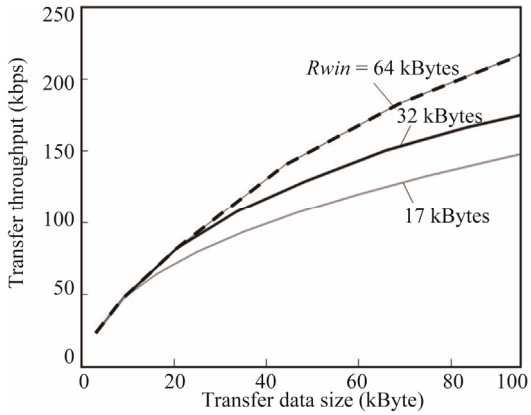

Figure 3. Transfer throughput for various data sizes and different *Rwin*.

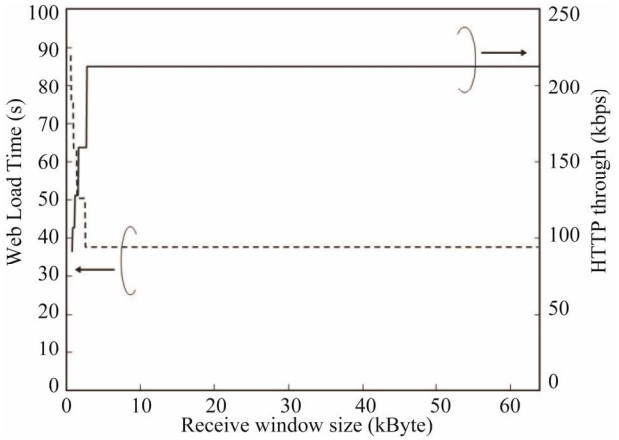

Figure 4. HTTP throughput for receive window size.

content size is 10 kByte, the total number of contents is 100, the number of simultaneous connections is 4, and RTT is 500 ms. As can be seen, HTTP throughput is restricted to 212 kbps and the Web load time is larger than 37.6 s when *Rwin* is only 4 kByte. Since a Web load time within 8 s is commonly known to be demanded for user satisfaction [13,14], a Web load time of 30 s by normal access in an extremely large latency environment will not satisfy user demands.

2.3. Conventional Prefetching Scheme

Several TCP acceleration schemes and protocol conversion schemes have been studied to boost Web service performance [15]. The schemes can accelerate TCP throughput by expanding *Rwin*, which improves congestion control algorithms. The Web service characteristic that a large amount of small-sized data is transferred makes achieving high-speed service difficult with such schemes.

Increasing the number of simultaneous connections can increase HTTP throughput, as shown by Equation (5). However, increasing the number of simultaneous connections means the Web server must manage a large number of sessions. Increasing the load of the server degrades its performance [16]. Hence, it is recommended by RFC2616 so that the number of simultaneous connections is restricted. A pipeline [17] scheme has also been studied, but few Web servers support this scheme.

A Web prefetching scheme [18-20] has been studied to improve HTTP performance. The prefetching server (Pf Serv.) in a wide area network (WAN), shown in **Figure 5**, enables high-speed HTTP throughput in a conventional scheme without expanding *Rwin* size or increasing the number of simultaneous connections. The prefetching server has two functions: prefetching, and forwarding the stored prefetched information. The sequence of the conventional prefetching scheme is shown in **Figure 6**.

The prefetching server captures the HTML file (data 1), which is the data sent in reply to the request from the client to the server. The prefetching server analyzes the file to issue the prefetching request (Pget 2) and stores the prefetched contents (data 2). After receiving the HTML file, the client may issue a request (get 2) for the same contents acquired by the prefetching server. The prefetching server replies to the request and sends the stored content (data 2).

The conventional prefetching scheme effectively doubles the throughput compared with normal access when the prefetching server is located halfway between the Web server and the client. However, the conventional scheme is not as effective for achieving the maximum performance in the target system, which would incorporate the prefetching server in the train or the ground station.

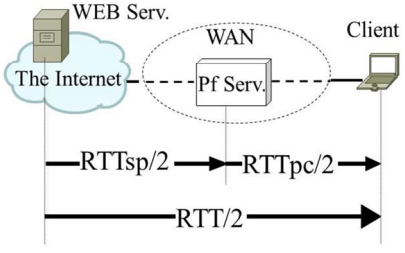

Figure 5. Configuration of conventional prefetching scheme.

3. Proposed Scheme

3.1. Operation of Proposed Prefetching Scheme

In this subsection, the proposed prefetching scheme is described. The proposed scheme is characterized by its division of the two functions base on the conventional prefetching scheme. Each function is installed so that a WAN (satellite link) may be inserted as shown in **Figure 7**. Therefore, a WAN with the extremely large latency is sandwiched by two servers with different function. The prefetching server, which prefetches for the Web server and forwards the prefetched contents to the information storage server, is installed in the network near the Web server. The information storage server, which stores the prefetched contents and forwards them to the client, is installed in the network near the client. The sequence of the proposed prefetching scheme is shown in **Figure 8**. The prefetching server captures the HTML file (data 1), analyzes it to issue a prefetching request (Pget 2), and

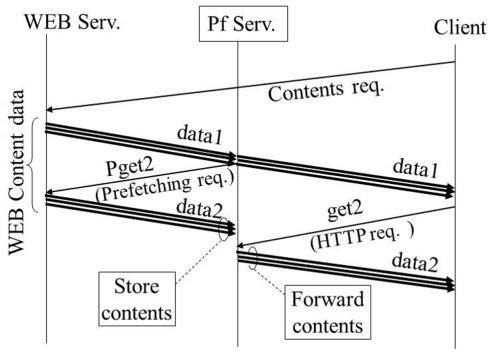

Figure 6. Signal sequence of conventional prefetching scheme.

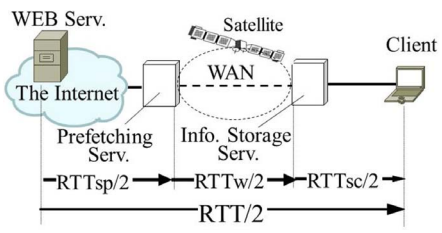

Figure 7. Configuration of proposed prefetching scheme.

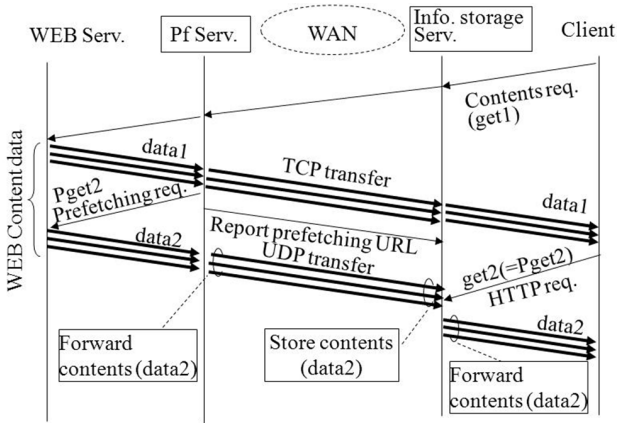

Figure 8. Signal sequence of proposed prefetching scheme.

acquires the content (data 2) noted in the HTML file.

Before sending data 2 to the information storage server, the prefetching server reports the prefetching URL to the information storage server, which replies with the prefetched content for the request (get 2) from the client.

Reliable communication that can be achieved by TCP with congestion control, retransmission control, and packet reordering may restrict the throughput. The User Datagram Protocol (UDP) is characterized by having no data transfer control such as implicit handshaking dialogues for providing reliability, receiving ordering, or data integrity. Because of this, UDP communication can provide high-speed communication in a large latency network when packet loss is negligible. This transmission characteristic was why we decided to use UDP in the proposed scheme to transfer prefetched contents.

Low cost development of the proposed scheme can be achieved by utilization the functions of the conventional scheme and transferring the prefetching data with high speed UDP.

The time taken to transfer prefetched contents is then only the contents size divided into the transmission speed and the time of propagation latency through WAN, so high-speed performance could be obtained with the proposed scheme. The satellite link of long distance accrues packet error by propagation loss. This system environment is significant problem in order to use UDP, since UDP does not have the function for packets error recovery. Scheme to solve this problem will be described in subsection 3.4.

3.2. Theoretical Evaluation of Web Load Time

To confirm the effectiveness of the proposed scheme, the Web load time was derived for normal access, *i.e.* without any prefetching, scheme, access with the conventional prefetching scheme, and access with the proposed prefetching scheme by theoretical analysis. The evaluation parameters are shown in **Table 1**.

Table 1. Evaluation parameters of web load time.

Size of one web content (*Sw*)	10 kBytes
No. of web contents	N
HTTP version	1.0
No. of simultaneous HTTP connections	1
Rwin	64 kBytes
RTT of client and Web server	510 ms
WAN latency	500 ms

Considering that the TCP uses window base control and congestion control as mentioned in Equations (1) and (2), the first transfer sends 1.46 kBytes (=1 *MSS*) of the content, the second transfer sends 2.92 kBytes (=2 *MSS*), and the third transfer sends the remaining 5.62 kbytes (=3.84 *MSS*, which is within 4 *MSS*). Since 10-kBytes content can be transferred in 3 transfers, it takes 3 × RTT to load one content. The web load time (*Tl*) can then be derived as follows.

1) Normal access

Each of the contents takes three RTTs to load. Since the Web service is composed of *N* number of contents, the Web load time can be expressed as follows.

$$Tl = 3 \times \mathrm{RTT} \times N \ . \tag{7}$$

2) Conventional prefetching scheme

Each of the contents is loaded to the prefetching server in three RTTs between the Web server and the prefetching server ($\mathrm{RTT_{sp}}$). Each of the prefetched contents in the prefetching server is loaded to the client in three RTTs between the prefetching server and the client ($\mathrm{RTT_{pc}}$). Therefore, the web load time is three times the larger value out of the $\mathrm{RTT_{sp}}$ and $\mathrm{RTT_{pc}}$ as follows.

$$Tl = 3 \times \max\left(\mathrm{RTT_{sp}}, \mathrm{RTT_{pc}}\right) \times N \tag{8}$$

Here, the sum of $\mathrm{RTT_{sp}}$ and $\mathrm{RTT_{pc}}$ is the RTT of the Web server and the client shown in **Figure 5**.

3) Proposed prefetching scheme

Each of the contents is loaded to the prefetching server in three RTTs between the Web server and the prefetching server ($\mathrm{RTT_{sp}}$). Each of the prefetched contents in the prefetching server is forwarded to the information storage server as content size (*Sw*) divided by the UDP transfer speed (*Su*). Each of the contents stored in the information storage server is loaded to the client in three RTTs between the information storage server and the client. Therefore, the web load time can be expressed as follows when it is assumed that no packet is lost in the WAN link.

$$Tl = 3 \times \max\left(\mathrm{RTT_{sp}},\ \mathrm{RTT_{pc}}\right) \times N + \left(Sw/Su\right) \times N \ . \tag{9}$$

The results of the load times derived from Equations (7), (8), and (9) are shown in **Figure 9**. As can be seen, the

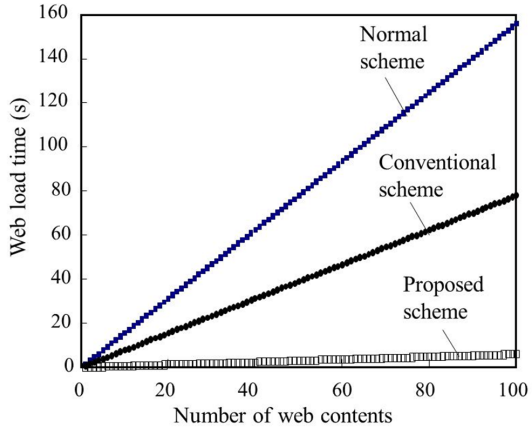

Figure 9. Web load time by different types of access for various numbers of web contents.

proposed scheme can significantly shorten the Web load time.

3.3. Test-Bed System of Proposed Scheme

A test-bed system was created to verify the effectiveness of the proposed scheme. The configuration of the prefetching server and the configuration of the information storage server are shown in **Figure 10** and **Figure 11** respectively. Apache Httpd 2.2 and Apache Tomcat 6.0.29 are installed on Red Hat Enterprise Linux 5.4. Each server operates as an HTTP server within a hierarchical structure. The servers perform in a so-called parent and child proxy configuration: the prefetching server is the parent proxy, and the information storage server is the child proxy. As a parent proxy, the prefetching server prefetches contents and forwards them by UDP to the information storage server. As a child proxy, the information storage server stores the prefetched contents and replies to requests from the client.

Considering practical utilization of the developed test-bed system, these servers must perform as an intercepting proxy, also known as a forced proxy or transparent proxy. Therefore, the information storage server is needed as a proxy, which intercepts normal communication, with clients not needing any special configuration to use the proxy. Clients do not need to even be aware of the existence of the proxy. In an intercepting proxy, utilization of the IP tables function installed in Linux detects the request packets from the client to the Web server to forward the packets to the port opened by the squid function. The squid function accepts the information storage server as the proxy server system.

Precise prefetching, *i.e.*, acquiring and storing the same content as the client's request, enables high-speed Web service. This section introduces a system for prefetching requests from an HTML file. The test-bed system targets HTML 4.01. The prefetching requests are

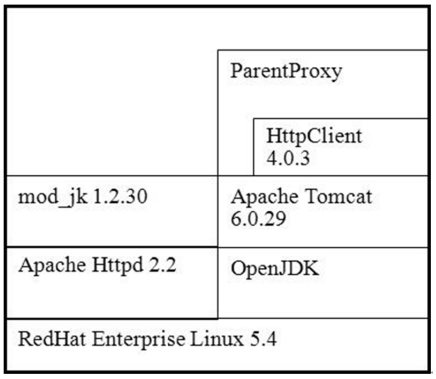

Figure 10. Configuration of prefetching server.

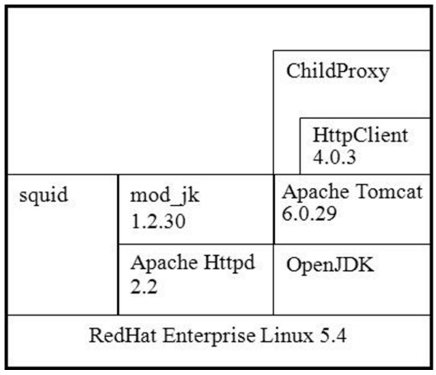

Figure 11. Configuration of information storage server.

extracted only when the "Content-Type" header value [21] of an HTTP reply includes "text/html", and the system is designed to extract the following elements in the HTML file.

- Pictures of src attribute in <img> tags.
- JavaScript of src attribute in <script> tags.
- "Style sheet" of href attribute in <link> tags.
- Pictures of href attribute of <link> tags.

3.4. UDP Packet Loss Recovery Function

Regarding the WAN link in this paper, packet loss may occur when the link is blocked by an obstacle or when packets spill over the network interface card of the client. Moreover, the satellite link of long distance has significant propagation loss compared to a wired link. If the prefetched contents are not transferred, the quality of the Web service utilizing this prefetching scheme may be degraded. Hence, the function to recover the packet loss is required in this system, since UDP does not have the function for packets error recovery.

To compensate for packet loss, a retransmission function (loss recovery function) was imposed in the test-bed system. The fundamental operation for recovering UDP packet loss is shown in **Figure 12**. The prefetching server divides a prefetched content into n number of UDP packets. The size of each packet is fixed to *L* Bytes.

The packets are added in ascending sequence number from 1 to *n*, and the *n*-th packet is indicated as the last packet. They are transferred for every fixed time. The fixed size and the fixed transmission interval can control the transmission speed so that it stays within the link speed of the WAN. The information storage server sends a retransmission request that includes the sequence number of lost packets to the prefetching server when the last packet is received. After receiving the request from the information storage server, the prefetching server retransmits the lost packet. Hence, all of the packets are confirmed to be transferred without any loss in the test-bed system.

4. Evaluation of Proposed Scheme

An evaluation test was performed to clarify the effectiveness of the proposed scheme by using the test-bed system. The configuration of the evaluation test is shown in **Figure 13**. The network emulator made a WAN environment that had latency of 250 ms in one-way trip time, *i.e.*, a 500-ms RTT in order to produce the satellite link, and the bandwidth was set to 100 Mbps. The rest of the network was linked with a 100-Mbps Ethernet LAN. The UDP size was fixed to 1414 Bytes of Segment Size, and the transfer interval was fixed to 1 ms. The client accessed the contents in the Web server with HTTP 1.0, and the number of simultaneous connections was 1. The contents, 28 in total, were 1 HTML file, 10 GIF files, 4 JPG files, and 13 PNG files: the total size of the contents was 1849.285 kBytes. Each of the contents was transferred in each HTTP session. The content packets were

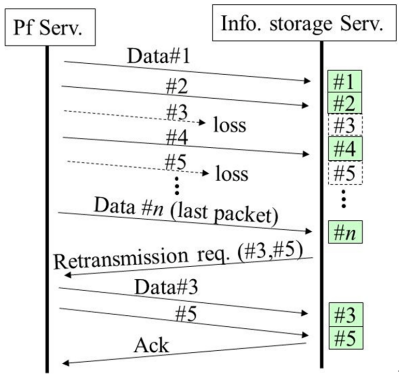

Figure 12. UDP packet loss recovery function.

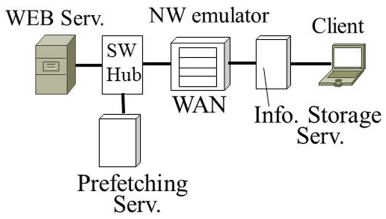

Figure 13. Configuration of evaluation test in laboratory.

captured in the client, and the final time of each HTTP session was observed. The test results and the theoretical values from Equation (6) are shown in **Figure 14**. Web load times of 47.4 s with normal access and 4.3 s with the proposed prefetching scheme were observed. The derived times using Equation (5) were 89.2 s with normal access and 4.2 s with the proposed prefetched scheme.

Anextensive improvement of about 21-times higher-speed web service was obtained, and the values obtained by the test agreed with the theoretical ones.

Next, the throughput variation at every one second with the proposed scheme and normal access are shown in **Figure 15**. The UDP packets are generated 1414 bytes with the interval of 1 ms. In the proposed scheme, a maximum throughput of about 14 Mbps was obtained in the elapsed time of 3 s, where 1.792-MBytes data were transferred from 2.302 to 2.952 s. This shows that high-speed Web service can be achieved with the proposed scheme. The other hand, Normal access was low throughput and long elapsed time.

The UDP packet loss recovery function was finally evaluated. As a methodology for evaluation test, UDP

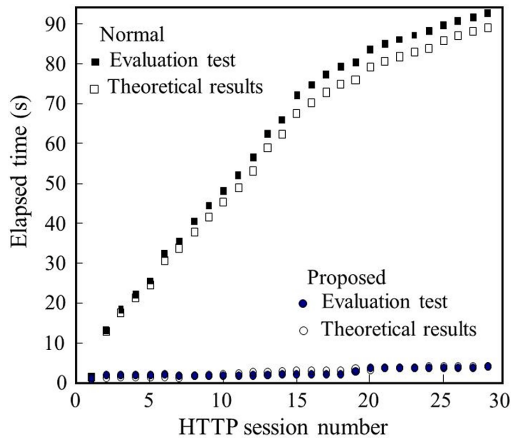

Figure 14. Evaluation test of the Web load times.

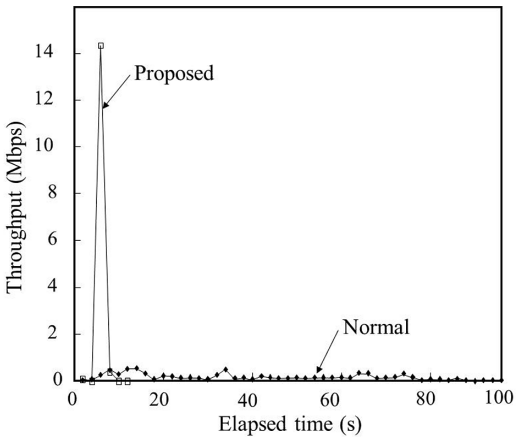

Figure 15. Throughput variation with proposed and normal access.

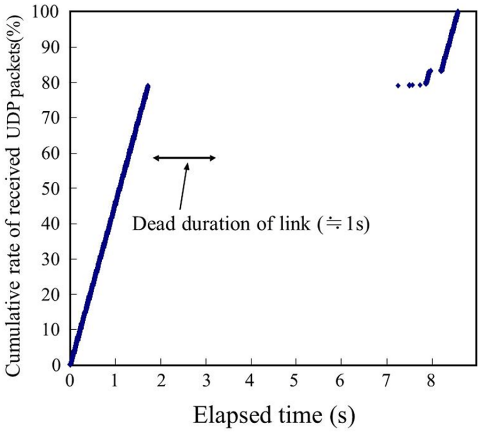

Figure 16. Evaluation of recovery function.

retransmission test was carried out by cutting the network cable off for about 1 s to cause packet loss.

The UDP packets were captured in the information storage server, and the retransmission operation was observed. The cumulative rate of successfully received in the information storage server is shown in **Figure 16**. In this test, the network cable was cut off for about 2 to 3 s of the elapsed time. The results show that 100 % of the UDP packets were perfectly transferred with the scheme.

The information storage server sends a retransmission request when packets are not received after it has waited for a certain time from receiving one packet that is not the last packet, or when packets replying to a retransmission request are not received. In this test, the waiting time was fixed at 5 s, and retransmission occurred after 7 s had elapsed. It is verified that reliable communication without any packet loss can reach with the installed scheme in the test-bed.

5. Conclusion

We investigated boosting web service performance over an extremely large latency network. We derived the theoretical performance and proposed a prefetching scheme that uses a prefetching server and an information storage server. The proposed scheme was implemented on a test-bed system. We carried out an evaluation test using the system and verified that the proposed scheme can enable high-speed Web service over an extremely large latency network. Reliable communication without any packet loss was also verified.

6. Acknowledgements

This work was supported in part by KAKENHI, Grant-in-Aid for Young Scientist (B) 22760272.

REFERENCES

[1] D. Raggett, A. L. Hors and I. Jacobs, "HTML 4.01 Specification," W3C Recommendation, 24 December 1999.

[2] T. Berners-Lee, R. Fielding and H. Frystyk, "Hypertext Transfer Protocol, HTTP/1.0.," RFC 1945 MIT/LCS, UC Irvine, May 1996.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol, HTTP/1.1," Internet Draft Standard RFC 2616, June 1999.

[4] T. Berners-Lee, R. Fielding and H. Frystyk, "Hypertext Transfer Protocol, HTTP/1.0.," RFC 1945 MIT/LCS, UC Irvine, May 1996.

[5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol, HTTP/1.1," Internet Draft Standard RFC 2616, June 1999.

[6] X. Liang, F. L. C. Ong, P. M. L. Chan and R. E. Sheriff and P. Conforto, "Mobile Internet Access for High-Speed Trains via Heterogeneous Networks," *The* 14*th IEEE* 2003 *International Symposium on Personal, Indoor and Mobile Radio Communication Proceedings*, Vol. 1, 7-10 September 2003, pp. 177-181.

[7] V. Schena and F. Ceprani, "FIFTH Project Solutions Demonstrating New Satellite Broadband Communication System for High Speed Train," 2004 *IEEE* 59*th Conference on Vehicular Technology*, Vol. 5, 17-19 May 2004, pp. 2831-2835.

[8] IEEE Std. 802.11, Part 11, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," ISO/IEC 8802-11, 2008.

[9] H. Obata, K. Ishida, J. Funasaka and K. Amano, "Data Transfer Time by HTTP1.0/1.1 on Asymmetric Networks Composed of Satellite and Terrestrial Links," *IEICE Transactions on Communications*, Vol. E85-B, No. 12, 2002, pp. 2895-2903.

[10] G. Hasegawa, M. Murata and H. Miyahara, "Performance Evaluation of HTTP/TCP on Asymmetric Networks," *International Journal of Communication Systems*, Vol. 12, No. 4, 1999, pp. 281-296.

[11] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control," RFC 2581, April 1999.

[12] W. R. Stevens, "The Protocols," In: *TCP/IP Illustrated*, Vol. 1, Addison-Wesley, Reading, 1994.

[13] Z. Reserch, "The Need for Speed II," *Zona Market Bulletin*, No. 05, April 2001.

[14] T. Hyashi, Y. Muto and I. Nakajima, "Subjective Quality Evaluation on Response Time in Web Service," *IEICE General Conference*, B-11-23, Ritsumeikan University, March 2011.

[15] T. Yoshino, Y. Sugawara, K. Inagami, J. Tamatsukuri, M. Inaba and K. Hiraki, "Performance Optimization of TCP/IP over 10 Gigabit Ethernet by Precise Instrumentation," *International Conference for High Performance Computing, Networking, Storage and Analysis* (*SC* 2008), 15-21 November 2008, pp. 1-12. doi:10.1109/SC.2008.5215913

[16] S. Souders, "High Performance Web Sites: Essential Knowledge for Frontend Engineers," O'Reilly Media,

September 2007.

[17] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie and C. Lilley, "Network Performance Effects of HTTP/1.1, CSS1, and PNG," *Proceedings of ACM SIGCOMM*'97 *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Vol. 27, No. 4, 1997, pp. 155-166.

[18] K. Chinen and S. Yamaguchi, "An Interactive Prefetching Proxy Server for Improvement of WWW Latency," *Proceedings of the Seventh Annual Conference of the Internet Society* (*INEt*'97), Kuala Lumpur, June 1997.

[19] V. N. Padmanabhan and J. C. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *ACM SIGCOMM Computer Communication Review*, Vol. 26, No. 3, 1996, pp. 22-36. doi:10.1145/235160.235164

[20] R. Shinkuma, M. Okada and S. Komaki, "Adaptive Transmission Scheme for Web Prefetching in Wireless Environment," *IEICE Transaction on Electronics*, Vol. E85-C, No. 3, 2002, pp. 485-491.

[21] "MIME Media Types". http://www.iana.org/assignments/media-types/index.html

