An Efficient Video Prefix-Caching Scheme in Wide Area Networks

Hyotaek Lim[1,⋆], DaeHun Nyang[2], and David H.C. Du[3]

[1] Department of Computer Engineering, Dongseo University
Busan, 617-716, Korea
htlim@dongseo.ac.kr
[2] Graduate School of Information Technology and Telecommunication
Inha University, Inchon, 402-751 Korea
nyang@inha.ac.kr
[3] Department of Computer Science and Engineering, University of Minnesota
Minneapolis, MN 55455, U.S.A
du@cs.umn.edu

Abstract. Web proxy caching provides an effective way to reduce access latency and bandwidth requirement. In particular, prefix caching is considered as an alternative for improving video delivery over wide area networks because video objects are usually too large to be cached in their entirety. Many studies have pointed that the user-perceived latency is often not dominated by object transmission time, but rather by setup process such as TCP connection time that precedes it. We propose *pre-connecting* techniques which hide the setup process such as TCP connection time in prefix caching proxy and show that the pre-connection can be used efficiently in TCP splicing. Our analysis shows the pre-connection significantly reduces start-up latency and TCP connection time in simple analytical model and hierarchical caching model, respectively. The deployment of the proposed pre-connection does not require protocol modification or the cooperation of other entities.

1 Introduction

As the web continues to grow exponentially, streaming media delivery is gaining popularity as indicated by dramatically increased deployment of commercial products for playback of stored video and audio over the Internet [1]. However, the service of quality as perceived by the end user is still very poor because the support is lacking in the Internet to meet delay and jitter requirements for realtime traffic. In particular, start-up latency is the central performance problem of streaming media delivery in the Internet today.

Caching of web objects for improving end-to-end latency and reducing network load have been studied extensively starting with CERN httpd, followed by improvements in hierarchical caching and co-operative caching in the Harvest

⋆ The first author was partially supported by Dongseo University, "Dongseo Frontier Project" Research Fund of 2002.

H.-K. Kahng and S. Goto (Eds.): ICOIN 2004, LNCS 3090, pp. 679–688, 2004.

project and the Squid project, respectively [9]. Recently, some works have been presented on proxy caching architectures for improving video delivery over wide area networks [9, 10, 11]. Video objects are usually too large to be cached in their entirety. Video files in size are usually much larger than other web documents. HTML documents and embedded images are usually in the range of 1Kbyte to 10Kbyte, however, video data such as mpeg, AVI, QuickTime typically exceed 1Mbyte [10]. The large size of video data makes conventional proxy caching techniques inappropriate.

As an alternative for video caching proxy, a prefix-caching proxy can be used efficiently in case of the large streams such as video files. A prefix-caching proxy caches the initial portion of the stream at proxy and transmits the prefix from the proxy to the client. During the transmission of the prefix to client, the proxy requests the remainder from the server. Fortunately, most of streaming protocols, such as HTTP/1.1 and RTSP, allow the proxy to fetch the remainder of the stream by random access.

A key realization is that streaming latency is often not dominated by object transmission time, but rather by the setup process that precedes it [7]. The dominating latency factors of the user perceived latency include name-to-address resolution, TCP connection establishment time, HTTP request-response time, server processing and finally transmission time. Information exchange between a client and a proxy, a proxy and a server in the Web is performed using HTTP, which typically uses TCP as the underlying transport protocol. Hence HTTP request and response messages are carried on a TCP connection established between the two entities. When the requested object is of small size, it is often the case that only two round trips are required to fetch the object: one for connection-establishment and the other for the object transfer itself.

This paper proposes a pre-connecting scheme to hide TCP connection establishment time, which is significant relative to HTTP request-response times when the prefix-caching proxy requests the remainder to the server [11, 10]. It allows the proxy to save time to support additional functions such as transcoding. We also show that the proposed protocols can be used efficiently in TCP splicing. The analytical results exhibit that the proposed scheme substantially reduces the start-up latency in a simple model and the connection time in hierarchical caching model.

The remainder of this paper is organized as follows. Section 2 proposes the pre-connection for prefix-caching video proxy and describes the results of applying the pre-connection to TCP splicing. Section 3 analyzes the proposed pre-connection. The analysis is concentrated on start-up delay and connection time in simple analytical model. Finally, Section 4 summarizes and concludes.

2 Pre-connection for Prefix-Caching

2.1 Connection for Prefix-Caching

Web clients and servers use HTTP which in turn uses TCP as its underlying reliable transport protocol. A TCP connection needs to be established and

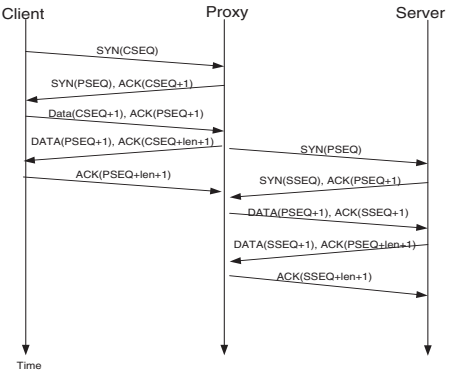

Fig. 1. Protocol Sequences in Normal Case

acknowledged prior to transporting HTTP messages. Fig. 1 illustrates the protocol timeline of TCP segments in conventional prefix caching when the proxy has cached the prefix of the video data. TCP connections are established with a 3-way handshaking: The client sends a SYN segment to the proxy or server, receiving the proxy or server's SYN segment with the ACK flag on , and then acknowledging the server's SYN. We assumed that a data segment(HTTP GET request) is sent on the 3rd segment of an initial 3-way handshaking to simplify the timing sequence. The 4rd segment contains the HTTP response to send the prefix of an object to client. CSEQ, PSEQ and SSEQ in the figure indicate client, proxy and server sequence number which is included in TCP segment, respectively. In most systems, the process to send the prefix to client and the process to request the remainder of a stream from the server are started concurrently. Nevertheless, as we can see in Fig.1, the proxy has to wait the TCP connection time before starting to fetch the remainder of a video stream from the server by invoking the range request specified in HTTP/1.1 [2, 13]. In real systems, the round-trip time for TCP connection between the proxy and the server may not be negligible because the proxy and server exist in WAN such as Internet. The round-trip time in hierarchical caching in which caches are placed at multiple levels of the networks degrades the quality of service significantly.

So, we propose the protocol sequence to hide the round-trip time for TCP connection as shown in Fig.2. It is possible for the prefix-caching proxy to pipeline the handshake by sending out the SYN segment to the server immediately after receiving the SYN segment from the client. The initial SYN segment contains the destination IP address and port number. The proxy has all of the necessary information for TCP connection setup. In the prefix-caching proxy, the sequence described in Fig.2 allows the proxy to have the time to support additional functions such as transcoding. This scheme is transparent to the client and server and follows a standard protocol. As the number of caching proxies increases, it significantly reduces the round-trip time for TCP connection.

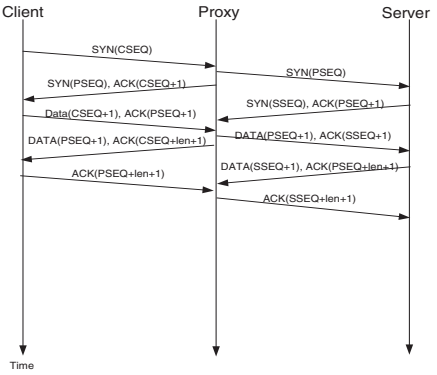

Fig. 2. Protocol Sequences in Proposed Case

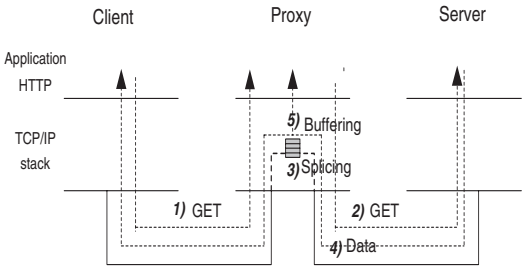

Fig. 3. Steps to process HTTP GET request in TCP-Splicing

2.2 Pre-connection in TCP-Splicing

The pre-connection proposed in the previous Section can be used in TCP splicing. TCP splicing is a technique to splice together two TCP connections that were independently set up with a proxy [3, 4, 12]. In conventional caching systems, received packets in the proxy are passed up through the protocol stack to application layer, where they are then passed back down again in order to be sent out to the requesting client. After the splice is created, however, these two connections should be one connection actually as shown in Fig. 3. The TCP splicing can improve the performance of the video prefix-caching proxy with the proposed protocols.

Fig. 3 shows all steps for the proposed prefix-caching proxy to process a cache miss in TCP splicing. Assuming that the prefix of the requested data is not in the proxy cache, the steps of the TCP splicing including the proposed protocols are as follows: (1) Proxy receives a GET from the client after TCP connection establishment, (2) The proxy sends a GET request immediately after receiving the 2nd segment for TCP connection from the server because the requested data is not in the cache, (3) The proxy splices the two connections, (4) The data

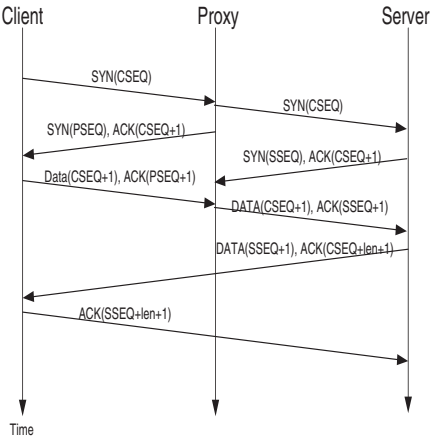

Fig. 4. Protocol Sequence in TCP-Splicing

flows through the splice from server to client, (5) Proxy application reads the data from tap buffer and spools into the proxy cache. With TCP splicing, the requested data can be sent from the server to the client without going up the application layer. During sending the data to client, at the same time, the proxy can cache the data from the server using an additional tap buffer in the proxy. The proposed pre-connection is used efficiently in the case of a cache miss or a system initialization.

Fig. 4 illustrates the TCP sequence which sends the data from the server to the client using the TCP splicing in the case of a cache miss. The sequence shows that the proxy sends out the SYN segment to the server immediately after receiving the SYN segment from the client as in the previous section. The proposed pre-connecting technique improves the performance of the TCP splicing by hiding the TCP connection time between the proxy and the server.

3 Analysis of Pre-connecting Prefix Caching

3.1 Analysis of Start-Up Delay

To analyze client start-up latency, we use a simple model as shown in Fig. 5. The model involves a caching proxy between the client and the server [6]. Client start-up latency can be defined as the time difference between sending a request and starting to play the media object at the client. We assume that the server send out data packet according to its playback rate r bytes/second, and each client keeps a buffer of K seconds. The client does not start playing the object until its buffer is filled. Also, we assume the delay between the server and the proxy is d_1, and between the proxy and the client is d_2. In Fig. 5, without a proxy the start-up latency, L_0 is $4(d_1 + d_2) + K$, where rational for the 4 is due to the

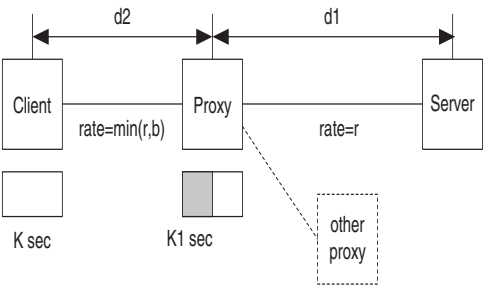

Fig. 5. Analysis Model

three-way handshaking of TCP connection and the delay to deliver an object. We assumed a HTTP request is sent on the 3rd segment to simplify the timing delay. Assuming the proxy has K_1 seconds of data in its cache, let's consider the start-up latency. At first, it takes $3d_2$ msec for the client request to arrive at the proxy because we assumed that a HTTP GET request is sent on the 3rd segment of the initial three-way handshaking. And then, the proxy starts two processes concurrently.

One process is to download the existing K_1 seconds of data to the client. Assuming the rate between the client and the proxy is b bytes/second, it takes $(K_1 \cdot r)/b$ seconds. The other process is to request $(K - K_1)$ seconds of data from other sources such as other proxy or the server. This process takes $4d_1$ because of the three-way handshaking of TCP connection. However, the proposed pre-connection takes $2d_1$ assuming that d_1 is equal to d_2 to simplify timing problem. It is because the proposed pre-connection hides the TCP connection time between the proxy and the server somewhat. Thus the time for both processes to finish is $max(K_1 \cdot r/b, 4d_1)$ and $max(K_1 \cdot r/b, 2d_1)$ in normal and proposed prefix caching, respectively. And then, the time for the remaining part of data to arrive at the client is $d_2 + (K - K_1) \cdot r/min(r, b)$. During this step, the buffer at the proxy is filled with rate r and drained with rate b. In order to avoid buffer underflow the actual draining rate for the buffer is set to $min(r, b)$.

The resulting start-up latency, L_1 is $4d_2 + max(K_1 \cdot r/b, 4d_1) + (K - K_1) \cdot r/min(r, b)$ and $4d_2 + max(K_1 \cdot r/b, 2d_1) + (K - K_1) \cdot r/min(r, b)$ in normal and proposed case, respectively. Fig. 6 shows the start-up latency for K_1 and b/r in normal case. As shown in the figure, the start-up latency is decreases as K_1 increases, and the start-up latency is decreases as b/r increases. Fig. 7 compares the start-up latency of normal and pre-connecting prefix caching. The figure shows that the start-up latency in proposed prefix caching indicates lower latency than in normal prefix caching as b/r increases.

3.2 Connection Time in Hierarchical Caching

With hierarchical caching, caches are placed at multiple levels of the network. We shall make the reasonable assumption that the cache levels consist of three

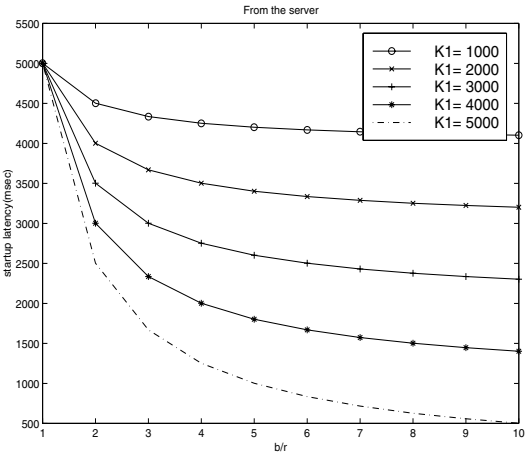

Fig. 6. Start-up Delay from Server

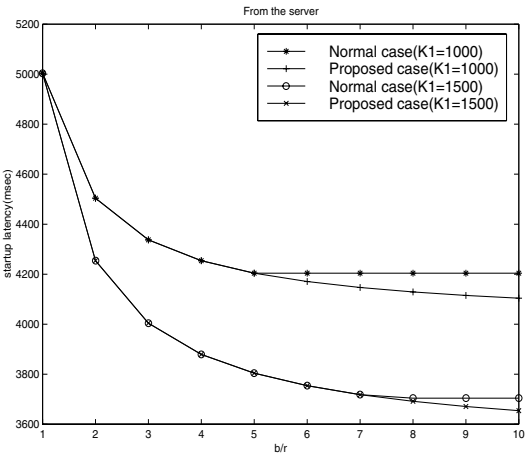

Fig. 7. Comparison of Normal and Pre-connecting Prefix Caching

levels of institutional, regional, national caches [8]. All the clients are connected to the institutional networks which contain their caches. When a request is not satisfied by the client cache, the request is redirected to the institutional cache. If the object is not found at the institutional level, the request is then forwarded to the regional level cache which in turn forwards unsatisfied requests to the national level cache. If the object is not found at any cache level, the national level cache contacts directly the original server, it travels down the hierarchy, leaving a copy at each of the intermediate caches along its path.

We model the network topology as a full O-ary tree, as shown in Fig. 8 [5]. The following notation is used in the model.

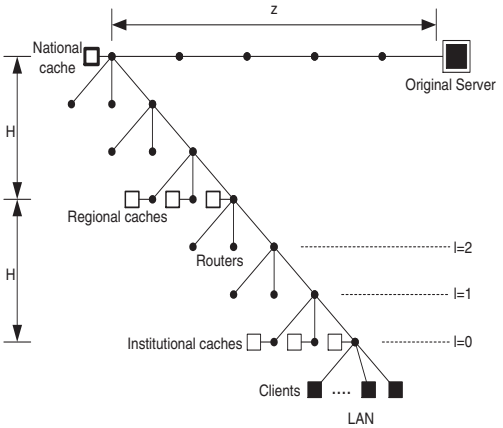

Fig. 8. Caching Hierarchy

- O: nodal outdegree of the tree
- H: number of network link between the root node of a national network and that of a regional network
- z: number of links between a origin server and root node
- l: level of the tree, $0 \leq l \leq 2H + z$
- d: per-hop propagation delay
- L: number of links that a request travels to hit a object in the caching hierarchy
- T_c: Connection time

Actually, the connection time depends on the number of links from the client to the cache containing the desired object. The connection time in normal caching can be formulated as follows [5]:

$$E[T_c] = 4d \sum_{l=\{0,H,2H,2H+z\}} P(L = l)(l + 1) \tag{1}$$

In the above equation, the factor of 4d, which is due to the three-way handshaking of TCP connection, is the dominant factor of the overall connection time. The proposed pre-connection as described in the previous Section can be used efficiently in hierarchical caching as well. It reduces the round-trip delay of the TCP connection time by connection hiding or pipelining. As the level of the hierarchical caching increases, the proposed pre-connection reduces the TCP connection time much more.

The connection time in the proposed prefix caching is given by

$$E[T_c^*] = 4d \cdot P(L = 0) + 2d \sum_{l} P(L = l)(l + 1) \tag{2}$$

where $l = \{H, 2H, 2H + z\}$.

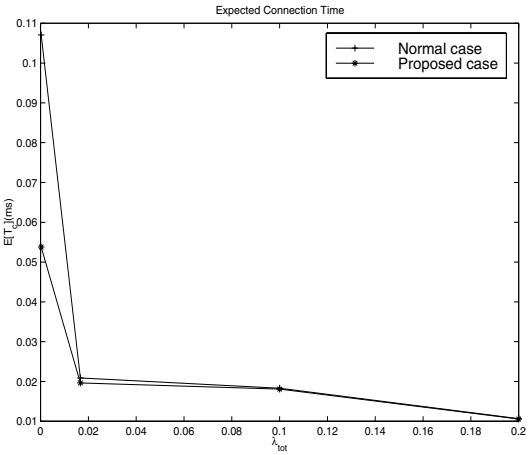

Fig. 9. Expected Connection Time

The first part of the above equation is the connection time to request a prefix in the institutional cache and the second part is the connection time to request the remainder in the caches of higher levels than institutional cache. The factor of $2d$ is due to pre-connecting for TCP connection. To calculate $P(L = l)$ we use $P(L = l) = P(L \geq l) - P(L \geq l + 1)$ as described in [5].

Fig. 9 shows the connection time of the normal and proposed case for different object's popularity. Assuming that requests for document i are uniformly distributed between all O^{2H} institutional caches, there are $\lambda_{tot} = \lambda_{I,i} \cdot O^{2H}$ total requests for document i. We observe that for unpopular object(small λ_{tot}) both normal and proposed case experience high connection times because the request needs to travel to the origin server. As the number of requests for an object increases(large λ_{tot}), the average connection time decreases since there is a higher probability to hit an object at closer caches than the origin server. For all the objects which λ_{tot} is smaller than 0.1, the proposed prefix caching gives shorter connection times than the normal prefix caching.

4 Conclusion

Prefix caching offers an effective way for improving video delivery over wide area networks. The prefix caching can be supported by the range request specified in HTTP/1.1 and RTSP which allows the proxy to fetch the remainder or suffix of the stream by random access. However, many studies have pointed that the user-perceived latency is often not dominated by object transmission time, but rather by setup process such as TCP connection time that precedes it. We have proposed a simple pre-connecting technique to hide the TCP connection time in prefix caching proxy and showed that the proposed technique can be used efficiently in TCP splicing. Our analysis has also showed the pre-connection

significantly reduces start-up latency and TCP connection time in simple model and hierarchical caching model in which caches are placed at multiple levels of the networks, respectively. The deployment of the proposed pre-connection does not require protocol modification or the cooperation of other entities. As part of our ongoing part, we are pursuing a good design and implementation of the video prefix-caching proxy system including the proposed technique. In addition, we are investigating how to combine the proposed technique with connection caching to reduce the latency of servicing user request.

References

[1] S. Gruber, J. Rexford, A. Basso, "Protocol considerations for a prefix-caching proxy for multimedia streams," *Computer Networks*, Vol. 33, 2000, pp.657-668. 679

[2] J. Jung, D. Lee, K. Chon, "Proactive web caching with cumulative prefetching for large multimedia data," *Computer Networks*, Vol. 33, 2000, pp.645-655. 681

[3] D. A. Maltz, P. Bhagwat, "Improving HTTP caching proxy performance with TCP tap", *Proc. of the Fourth International Workshop on High Performance Protocol Architectures (HIPPARCH'98)*, June 1998, pp.98-103. 682

[4] G. Apostolopoulos, D. Aubespin, V. Peris, P. Pradhan, D. Saha, "Design, implementation and performance of a content-based switch," *Proc. of the IEEE Infocom*, Mar. 2000. 682

[5] P. Rodriguez, C. Spanner, E. W. Biersack, "Web Caching Architectures: Hierarchical and Distributed Caching", *Proc. of the 4th International Caching Workshop*, San Diego, California. Apr, 1999. 685, 686, 687

[6] E. Bommaiah, K. Guo, M. Hofmann, S. Paul, "Design and Implementation of a Caching System for Streaming Media over the Internet", *IEEE Real-Time Technology and Applications Symposium (RTAS)*, Washington D. C., USA, May 31-June 2, 2000. 683

[7] E. Cohen, H. Kaplan, "Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency", *Proc. of the IEEE Infocom*, Mar. 2000. 680

[8] Jia Wang, "A Survey of Web Caching Schemes for the Internet", *ACM CCR* Vol. 29, Nov. 1999. 685

[9] Markus Hofmann, T. S. Eugene Ng, Katherine Guo, "Caching Techniques for Streaming Multimedia over the Internet", *Bell Labs Technical Memorandum*, April, 1999. 680

[10] Soam Acharya, "Techniques for Improving Multimedia Communication Over Wide Area Networks", Ph.D. thesis, Cornell University, Dept. of Computer Science, 1999. 680

[11] S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams", *Proc. of the IEEE Infocom*, Mar. 1999. 680

[12] Cohen, A., S. Rangarajan, and H. Slye. On the Performance of TCP Splicing for URL-aware Redirection. *Proc.of the USENIX Symposium on Internet Technologies and Systems*, pp. 117-125, October 1999. 682

[13] Hyotaek Lim, David H. C. Du, "Protocol Considerations for Video Prefix-Caching Proxy in Wide Area Networks", To appear, *IEE Electronics Letters*, 2001. 681