PROXY PREFIX CACHING for MULTIMEDIA STREAMS

S. SEN, J. REXFORD
and D. TOWSLEY

CMPSCI Technical Report 98-27

July 1998

Proxy Prefix Caching for Multimedia Streams

Subhabrata Sen†*, Jennifer Rexford‡, and Don Towsley†

†Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
{sen,towsley}@cs.umass.edu

‡Networking & Distributed Systems
AT&T Labs – Research
Florham Park, NJ 07932
jrex@research.att.com

CMPSCI Technical Report 98-27
University of Massachusetts

Abstract

Proxies are emerging as an important way to reduce user-perceived latency and network resource requirements in the Internet. While relaying traffic between servers and clients, a proxy can cache resources in the hope of satisfying future client requests directly at the proxy. However, existing techniques for caching text and images are not appropriate for the rapidly growing number of continuous media streams. In addition, high latency and loss rates in the Internet make it difficult to stream audio and video without introducing a large playback delay. To address these problems, we propose that, instead of caching entire audio or video streams (which may be quite large), the proxy should store a *prefix* consisting of the initial frames of each clip. Upon receiving a request for the stream, the proxy immediately initiates transmission to the client, while simultaneously requesting the remaining frames from the server. In addition to hiding the latency between the server and the proxy, storing the prefix of the stream aids the proxy in performing *workahead smoothing* into the client playback buffer. By transmitting large frames in advance of each burst, workahead smoothing substantially reduces the peak and variability of the network resource requirements along the path from the proxy to the client. We describe how to construct a smooth transmission schedule, based on the size of the prefix, smoothing, and playback buffers, without increasing client playback delay. Through experiments with MPEG-1 and MPEG-2 traces, we show how a few megabytes of buffer space at the proxy can offer substantial reductions in the bandwidth requirements of variable-bit-rate video. Drawing on these results, we present guidelines for allocating buffer space for each stream, and how to effectively share buffer and bandwidth resources among multiple clients and streams.

Keywords: multimedia streaming, proxy caching, workahead smoothing, variable-bit-rate video, resource allocation

1 Introduction

The dramatic growth of the World Wide Web in the past few years has led to significant increases in user-perceived latency and network congestion for Internet applications. Service providers can reduce response time,

*The work of this author was performed while visiting AT&T Labs – Research.

server load, and network traffic by deploying proxy caches. A proxy cache stores recently accessed resources in the hope of satisfying future client requests without contacting the server [1–3]. However, existing techniques for caching text and image resources are not appropriate for the rapidly growing number of continuous media streams in the Internet. Storing the entire contents of several long streams would exhaust the capacity of a conventional proxy cache. Instead, any scalable caching solution should store just a portion of each stream. In particular, we propose that proxy caches should store a fixed set of frames at the *beginning* of each popular video (in effect a prefix), instead of storing the entire resource, as is typically the case in text and image caching.

Storing the initial frames of each continuous media stream is motivated by the observation that audio and video applications typically experience poor performance, due to the unpredictable delay, throughput, and loss properties of the Internet. Consider an environment where multiple clients request continuous media from a collection of servers. In the absence of a proxy, each client request proceeds directly to a server, which streams a sequence of frames. Before initiating playback, the client must wait for the round-trip delay to the server, and accumulate enough frames to tolerate jitter, or to allow time for retransmission of lost or corrupted packets. Even if the client's service provider has quality-of-service support for multimedia streams, the provider does not have control over the entire path between the two sites. As a result, the application may have to tolerate high and/or variable communication delays, particularly if the rest of the network provides best-effort service or at most coarse-grain traffic differentiation. In the absence of support inside the network, the application must either increase playback delay or experience degraded quality.

We therefore propose that service providers can deploy multimedia proxies along the path from the server to the client, as shown in Figure 1. Similar to traditional caching of text and image data, storing (part of) each audio and video stream enables the proxy to reduce client delay, without sacrificing quality. Upon receiving a client request, the proxy immediately initiates transmission to the client from the prefix cache, while simultaneously requesting the remainder of the frames from the server. In forwarding the stream, the proxy can capitalize on any quality-of-service guarantees (such as bounds on throughput and delay) along the path to the client. Section 2 describes the operation of the proxy in greater detail, and highlights how storing the prefix of the stream can improve the delivery of continuous media in the presence of network delay and loss. We also discuss how the prefix proxy can operate without requiring changes to the server, by using standard primitives in HTTP 1.1 [4] and RTSP (Real-Time Streaming Protocol) [5].

Although proxy prefix caching applies to both constant-bit-rate and variable-bit-rate streams, the presence of a prefix buffer offers additional advantages in transmitting variable-bit-rate video. High-quality video

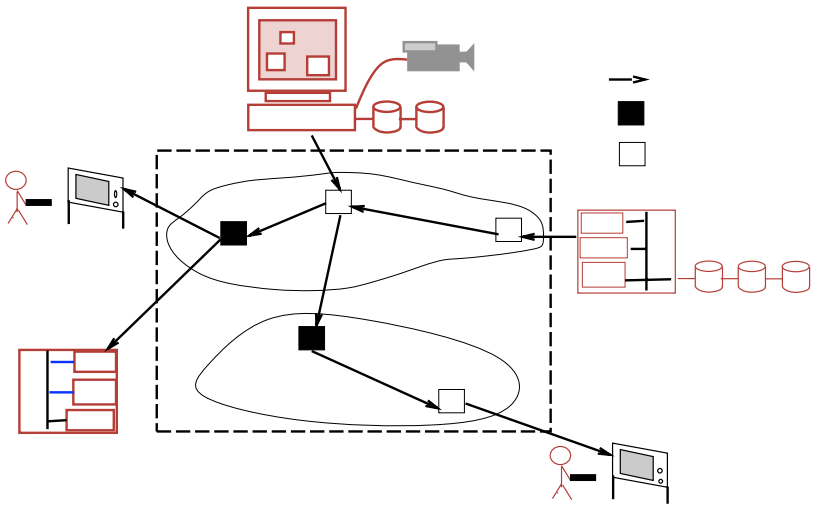


Figure 1: **Prefix proxy in an internetwork:** The prerecorded streams originate from a multimedia Web server or a video-on-demand system. A stream travels through the network, crossing one or more network domains, to one or more clients, including workstations and set-top boxes. Prefix caching is performed at proxy servers inside the network.

streams exhibit significant burstiness on a variety of time scales, due to the frame structure of the encoding scheme and natural variations within and between scenes [6–11]. The transmission can become even more bursty when a video source is combined with text, audio, and images as part of an orchestrated multimedia stream. This variability complicates the design of real-time transport mechanisms capable of achieving high resource utilization. The proxy can reduce the network resource requirements on the path to the client by performing *workahead smoothing* into the playback buffer. By transmitting large frames in advance of each burst, we can substantially lower the peak rate and variability of the traffic between the proxy and the client [12]. Such smoothing at the proxy would typically introduce additional delay, and requires some buffering, since the proxy must accumulate an initial window of frames from the server. The proxy prefix hides this delay, allowing the proxy to perform smoothing without increasing client playback latency.

In Section 3, we present a detailed model of workahead smoothing in the presence of a prefix buffer, and show how to compute smoothed transmission schedules. Section 4 evaluates the combination of workahead smoothing and prefix caching using MPEG-1 and MPEG-2 video traces. The experiments demonstrate that a few megabytes of buffer space can substantially reduce the network resource requirements for transmitting constant-quality, variable-bit-rate video. Additional experiments explore how to allocate the proxy buffer space between the prefix and smoothing functions. Then, in Section 5 we introduce techniques for allocating buffer and bandwidth resources across multiple clients, based on the popularity and resource requirements of each

stream. To further reduce the buffer requirements, we describe how to multiplex access to a shared smoothing buffer across multiple client streams. We then conclude the paper in Section 6 with a discussion of future research directions.

The paper complements recent work on caching and transmission of continuous media. Previous research on Web caching considers effective policies for storing and replacing individual resources based on their access patterns [1–3]. We extend this work by storing a prefix of frames of popular audio and video streams, to hide the latency between the proxy and the server. This allows the proxy to deliver good quality-of-service to the client, while hiding the weaker service model between the server and the proxy. With some additional buffer, the proxy can also perform workahead smoothing to reduce the resource requirements on the path to the client. Earlier work on multimedia caching proposed techniques for storing an interval of successive frames to satisfy requests that arrive close in time [13–15]. In addition to supporting interval caching, our proxy caching model hides the start-up latency even for requests that do not occur near each other in time, and reduces network resource requirements through workahead smoothing.

Previous research on workahead smoothing has considered video-on-demand environments, where the server controls the entire path from the prerecorded stream to the client playback buffer [16–20]. We complement this research by introducing the notion of smoothing at a proxy in the interior of the network. Extending our earlier work on online smoothing [12], we consider how the presence of a prefix buffer reduces the start-up latency and changes the constraints on smoothing. Our model also relates to a recent proposal that the proxy store a portion of the frames across the duration of a multimedia stream [21]. Although this scheme also allows the proxy to perform smoothing, our approach has the advantage of hiding start-up latency, being transparent to the server, and having storage costs that are independent of the length of the stream. Finally, our study complements other recent efforts to perform multimedia services, such as retransmission and transcoding, at proxies inside the network [22, 23].

2 Prefix Caching

The proxy stores the initial frames of the multimedia stream, and retrieves the remainder of the stream from the server through existing Web and real-time streaming primitives. The size of the prefix buffer depends on the trade-off between proxy disk or memory space and the need to shield the client from delay and loss along the path from the server.

2.1 Transparent Operation

A multimedia proxy can be deployed to reduce the burden on a server hosting platform, or to reduce latency to clients, as shown in Figure 1. For example, if a network service provider hosts a collection of users accessing continuous media, the proxy caches the prefixes of the popular streams. Particularly if these streams originate in other network domains, the proxy hides latency on the path from the server to the proxy. The proxy must reside on the path between the client and the server. For example, the client browser or player can be configured to transmit traffic through the proxy, as is common in the Web. This enables the proxy to intercept the client request and server transmissions, and stream frames directly to the client site. Upon receiving the first client request for a particular stream, the proxy retrieves the entire stream from the server, and caches the prefix, along with control information such as frame sizes. For future requests, the proxy can initiate transmission directly from the cache, while receiving the remainder of the stream from the server.

Initial distribution of the prefix of the stream is simpler when the proxy and the server belong to the same service provider. In this case, the proxy can be explicitly configured to store the prefix, and contact the server for the remainder of the stream. Or, the server can push the prefix of popular streams to a set of proxy sites. Satisfying part of a client request at the proxy reduces the total load on the server, and allows the service provider to tolerate additional delay and jitter in the path between the server and the proxy. To ensure that the client request goes through the proxy, the IP address of the streaming service can resolve to the proxy, which in turn contacts the server directly for the rest of the stream. In this case, the proxy acts as a (partial) replica of multiple server sites. Alternatively, the service provider can ensure that a proxy resides on the path from the client to the server, allowing the proxy to intercept the client request, similar to the notion of transparent caching in the Cisco Cache Engine [24].

For successful deployment, the prefix caching model should not require any changes at the server sites, particularly when the multimedia server resides in the domain of a different service provider. The proxy must have an effective way to identify and store the sequence of initial frames, and request the remaining frames from the server. For example, Real-Time Protocol (RTP) [25] encapsulation of packets includes sequence number and timestamp information, enabling the proxy to identify the frames in the prefix and schedule their transmission to the client. When a client requests the video, the proxy must ask the server to initiate transmission of the remaining frames, instead of sending the entire stream from the beginning. If the continuous media stream is a Web resource, the proxy can invoke the *byte range* operation in HTTP 1.1 [4] to request the appropriate

portion of the stream. Similarly, the Real-Time Streaming Protocol (RTSP) [5] supports *absolute positioning* to initiate transmission with an offset from the beginning of the stream. Either mechanism would allow the proxy to retrieve the necessary frames without requiring changes to the underlying HTTP or RTSP protocols.

2.2 Proxy Prefix Size

Through careful selection of the prefix size, the proxy can hide the latency and loss along the path from the server, without affecting the operation at the client site. The prefix may be stored on disk or in main memory. Even if the prefix is stored on a slower disk subsystem, the proxy could cache the first few frames of the prefix in main memory to hide the latency of disk access; the proportion of main memory dedicated to each stream may depend on its popularity. To size the proxy buffers, suppose that the delay from the server to the proxy ranges from $d_{\min}$ to $d_{\max}$, where time is measured in units of frame slots. To provide a start-up delay of s to the client, the proxy stores a prefix of at least $\max\{d_{\max} - s, 0\}$ frames. The proxy also devotes disk or buffer space to part of the stream from the server, to absorb jitter and retransmissions. This buffer must have space to store an interval of at least $d_{\max} - d_{\min}$ frames from the server. The proxy can devote additional space to each buffer to support workahead smoothing, as discussed in Section 3.

Recent Internet measurements of delay, loss, and resource sizes lend insight into how to size the proxy prefix buffer. Round trip delays vary widely, depending on the end-points and the degree of congestion in the network, but delays of several seconds are not uncommon [26–28]. In the absence of the proxy prefix buffer, these delays are visible to the user. Even when large delays do not occur, the audio or video player must introduce delay to build up a large playout buffer, or risk playout disruptions during periods of high delay. Instead, the proxy could store several seconds (say, 5 seconds) of the continuous media stream to hide this latency, as well as the server delay in responding to the request. Even for a high-bandwidth MPEG-2 stream, this would only require around 2.5–3 Mbytes of prefix buffer space at the proxy.

Similarly, Internet packet loss rates range from 2–10% [26–28]. Full motion video transmission requires the transmission of 30-frames a second, i.e., 1 frame (about $15 - 30$ packets) every 33ms. Because the packets are sent out so close to each other, even a short duration congestion situation on the transmission path in the network could potentially result in the loss of a sequence of consecutive packets in the video stream. With additional buffering, the proxy could tolerate bursts of lost packets during periods of heavy congestion. By buffering an extra round-trip time of data, the proxy can hide the delay of a single retransmission from the server (or from a retransmission proxy along the path from the server [23]). One or two seconds of prefix

buffering could handle the common case of the loss of a single packet or a short sequence of packets. For example, for an independent loss rate of 5%, a single retransmission increases the likelihood of receiving a packet from 95% to 99.75%. Placing the proxy close to the clients, or having the proxy-client traffic follow a well-provisioned (or QoS) path ensures that the client receives a high-quality stream without increasing playout delay.

Finally, prefix caching also reduces the traffic between the server and proxy, without having to store the entire stream. This may not a significant advantage for long streams, since the initial part of the stream would not represent a significant proportion of the transfer. But, many Internet audio and video clips are short. A 1997 study found that Web video streams have a median size of of 1.2 MBytes, with 90% of the streams lasting less than 45 seconds [29]. Although the size and duration of continuous media streams are likely to grow dramatically over time, particularly as high-bandwidth access networks become more common, the Internet is still likely to have a large number of short clips, such as advertisements and trailers. The proxy prefix cache can store all, or at least a sizeable portion, of these short streams. For popular streams that are accessed by multiple clients, this prefix caching can significantly reduce the load on the server, and on the network.

3 Workahead Smoothing

Storing a prefix of a multimedia stream enables the proxy to perform workahead smoothing into the client playback buffer without increasing delay. Smoothing minimizes the peak bandwidth and the burstiness of the variable-bit-rate stream, subject to constraints on buffer space and delay. In this section, we present a model of workahead smoothing at the proxy, including the constraints introduced by proxy prefix caching, and describe how to schedule transmissions to the client.

3.1 Smoothing Model

The emergence of high-speed backbone and access networks facilitates a wide range of multimedia applications, including streaming of high-quality video and orchestrated media. These streams consume a significant amount of network bandwidth, ranging from 1–10 Mbits/second, even in compressed form. In addition, compressed video exhibits significant burstiness on a variety of time scales, due to the frame structure of encoding schemes (such as MPEG) and natural variations within and between scenes. The burstiness could be reduced by adjusting the quantization level of frames during scenes with significant detail or motion, at the expense

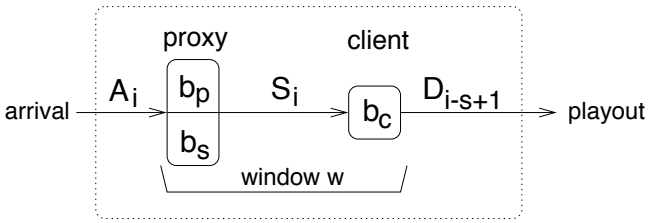


Figure 2: **Smoothing model:** This figure shows the smoothing model for scheduling the transmission of a single video stream from the proxy to a client. The proxy stores the first b_p bits of the video, and uses an additional b_s bits for smoothing into the b_c-bit client buffer. The proxy computes a transmission schedule $\mathbf{S}$, based on the arrival vector $\mathbf{A}$ and the playout vector $\mathbf{D}$, as well as the buffer constraints, the smoothing window w, and the client playback delay d.

of video quality. For the same average bandwidth, a constant-quality variable-bit-rate encoding offers higher quality and more opportunities for statistical multiplexing gain than would be possible for a constant-bit-rate encoding [11, 30]. Exploiting the benefits of variable-bit-rate encoding requires effective techniques for transporting bursty traffic across the network. Workahead smoothing reduces the variability of network resource requirements by transmitting large frames in advance of each burst, without compromising the quality of the video stream. By combining workahead smoothing with prefix caching, the proxy can reduce network overhead on the path to the client without increasing playback delay.

The opportunities for smoothing depend on the frame and buffer sizes, as well as the smoothing window and the playback delay, as shown in Figure 2 and Table 1. Without loss of generality, we consider a discrete time model at the granularity of a frame slot (e.g., $1/30$ of a second for a 30 frames/second video stream). A video stream consists of N frames, where frame i is f_i bits long, $i = 1, \ldots, N$, and $D_i = \sum_{j=1}^{i} f_j$ is the cumulative sum of frame sizes. Upon receiving a client request for a video at time 0, the proxy contacts the server to initiate transmission and sends the initial frames of the video to the client from the prefix buffer. The proxy introduces an s-frame playout delay, beyond the transmission delay between itself and the client, and smoothes the incoming video into the b_c-bit client buffer. Although the proxy smoothes the incoming video over a window of $w \geq s$ frames, the value of s dictates how much time is available for transmitting the first few frames of the video. The client begins playback of the video at time s, following the original unsmoothed schedule D_{i-s}.

The incoming frames from the server beyond the prefix frames are *temporarily* stored in the proxy's b_s-bit *smoothing* buffer while they await transmission to the client. As discussed in Section 2, we define d_{min} and d_{max} respectively as the minimum and maximum delays from the server to the proxy, resulting in a maximum

Parameter	Definition
s	Client playback delay (in number of frames)
w	Proxy smoothing window (in number of frames)
d_{min}	minimum server delay (in number of frames)
d_{max}	maximum server delay (in number of frames)
Δ	jitter on path from server to proxy (in number of frames)
b_p	Proxy prefix buffer size (in number of bits)
b_s	Proxy smoothing buffer size (in number of bits)
b_c	Client playback buffer size (in number of bits)
$\mathbf{D}$	Cumulative frame size vector (in bits per frame slot)
$\mathbf{A}$	Arrival vector at proxy (in bits per frame slot)
$\mathbf{S}$	Transmission vector from proxy (in bits per frame slot)

Table 1: **Parameters in smoothing model:** This table summarizes the key parameters in the smoothing model.

network jitter of $\Delta = d_{max} - d_{min}$. To accommodate the client's s-frame playback delay, the prefix buffer stores the first $d_{\max} - s + w$ frames in a prefix buffer of size $b_p = D_{d_{max}-s+w}$, and requests the server to initiate transmission of the video stream from frame $d_{max} - s + w$. The parameter A_i represents the cumulative amount of data that has arrived at the proxy by time i, including the initial prefix stored in the b_p-bit prefix buffer (i.e., $A_0 = D_{d_{max}-s+w}$). Although A_i could represent any arrival pattern at the proxy, we focus on the case where the proxy receives an unsmoothed sequence of frames, subject to a maximum jitter of Δ. That is, $A_i \in [A_i^{\min}, A_i^{\max}]$, where

$$A_i^{min} = \begin{cases} D_{d_{max}-s+w}, & i = 0, \ldots, d_{max} \\ D_{i-s+w}, & i = d_{max}+1, \ldots, N+s \end{cases}$$

and

$$A_i^{max} = \begin{cases} D_{d_{max}-s+w}, & i = 0, \ldots, d_{min} \\ D_{i-s+\Delta+w}, & i = d_{min}+1, \ldots, N+s. \end{cases}$$

In the simple case where $w = s$ and $d_{max} = d_{min} = 0$, the proxy does not store any of the video in advance, and the model reduces to the online smoothing framework in [12, 31].

3.2 Smoothing Constraints

Based on the frame and buffer sizes, the proxy computes a schedule $\mathbf{S}$ that transmits S_i bits by time $i = 1, 2, \ldots, N+s$. The parameters in the smoothing model translate into a collection of constraints on the smooth-

ing schedule **S**. The client must receive at least D_{i-s} bits by time i to avoid underflow of the playback buffer, resulting in a lower constraint:

$$L_i^a = \begin{cases} 0, & i = 0, \ldots, s-1 \\ D_{i-(s-1)}, & i = s, \ldots, N+s. \end{cases}$$

To compute the upper constraint U_i on any feasible transmission schedule, note that due to the variability in server delay, the proxy can be sure of receiving only A_i^{min} bits of the video by time i. Also, by time i it cannot send more than $L_{i-1}^a + b_c$ bits without overflowing the client playback buffer. Hence,

$$U_i = \min\{L_{i-1}^a + b_c, A_{i-1}^{min}\},$$

where $S_i \leq U_i$ for $i = 1, 2, \ldots, N+s$. Our upper bound computation conservatively uses A_{i-1}^{min} instead of A_i^{min}, as $A_i^{min} - A_{i-1}^{min}$ may reach the proxy only at time $i-$, too late to contribute to S_i.

The constraints L_i^a and U_i do not relate directly to prefix caching, except in the sense that the prefix buffer affects the expression for A_i. However, the prefix buffer does impact the overflow constraint on the proxy's b_s-bit smoothing buffer. At the beginning of the transmission, the proxy sends frames from the prefix buffer and does not free any space in the smoothing buffer. The proxy must complete transmission of the prefix frames before the smoothing buffer overflows. Note that due to the variability in server delay, the maximum cumulative amount of data the proxy can receive by time i is A_i^{max}. The earliest time the arriving stream would exhaust the smoothing buffer is then

$$i^* = \min\{i | A_i^{max} - b_p > b_s\}.$$

For all $i \geq i^*$, the proxy must have transmitted at least $A_i^{max} - b_s$ bits to prevent overflow of the smoothing buffer, resulting in the lower constraint:

$$L_i^b = \begin{cases} 0 & i < i^* \\ A_i^{max} - b_s & i \geq i^*, \end{cases}$$

Combined with the underflow constraint at the client buffer, this results in the following *lower constraint* on any feasible transmission schedule:

$$L_i = \min\{L_i^a, L_i^b\} \quad \text{where } L_i \leq S_i \text{ for } i = 1, 2, \ldots, N+s$$

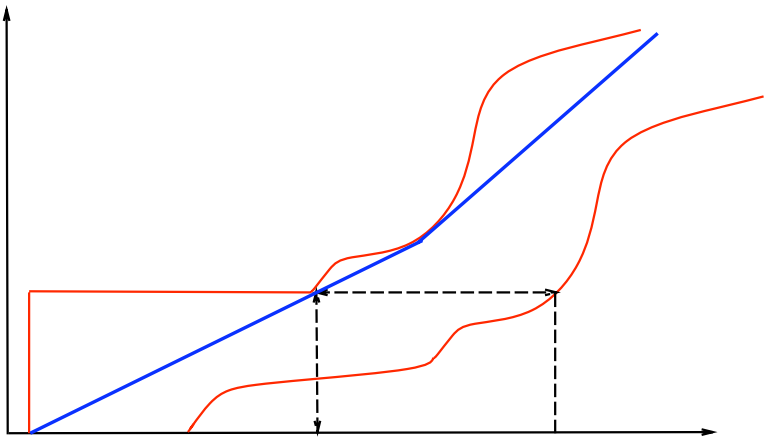


Figure 3: This figure shows the upper and lower constraint curves for a scenario where both the staging buffer b_s and client buffer b_c are sufficiently large that they do not impose any constraints on smoothing, and $d_{max} = d_{min} = d$. As such the upper and lower constraints at time i are effectively A_i and D_{i-s+1} respectively.

Figure 3 shows an example of the upper and lower constraints. Subject to the constraint $L_i \leq S_i \leq U_i$, the proxy can compute a transmission schedule that satisfies $S_0 = 0$ and $S_{N+s} = D_N$ and optimizes a target performance metric. To aid in computing the smoothing constraints and hence the smoothed transmission schedule, the proxy can store information about the sizes of the frames in the video stream.

3.3 Transmission Schedule

Consider the constraint vectors $\mathbf{L} = (L_0, \ldots, L_{N+s})$ and $\mathbf{U} = (U_0, \ldots, U_{N+s})$. Creating a transmission schedule $\mathbf{S}$ involves generating a monotonically non-decreasing path that does not cross either constraint curve. The constraints $\mathbf{L}$ and $\mathbf{U}$ typically result in multiple feasible transmission schedules with different performance properties [20]. In particular, in the context of smoothed transmission of stored video from the server to the client, an $O(N)$ smoothing algorithm [18] was proposed, that computes the shortest-path transmission schedule $\mathbf{S}^*$. The schedule $\mathbf{S}^*$ minimizes a variety of important performance metrics, including the peak and standard deviation of the transmission rates, and the effective bandwidth. We therefore adopt this shortest-path algorithm for evaluating proxy smoothing in Section 4. Figure 3 shows a sample transmission schedule.

Since we are considering prerecorded video, where the frame-size information is available in advance, the schedule $\mathbf{S}^*$ could be computed in advance and stored at the proxy, along with the initial frames of the

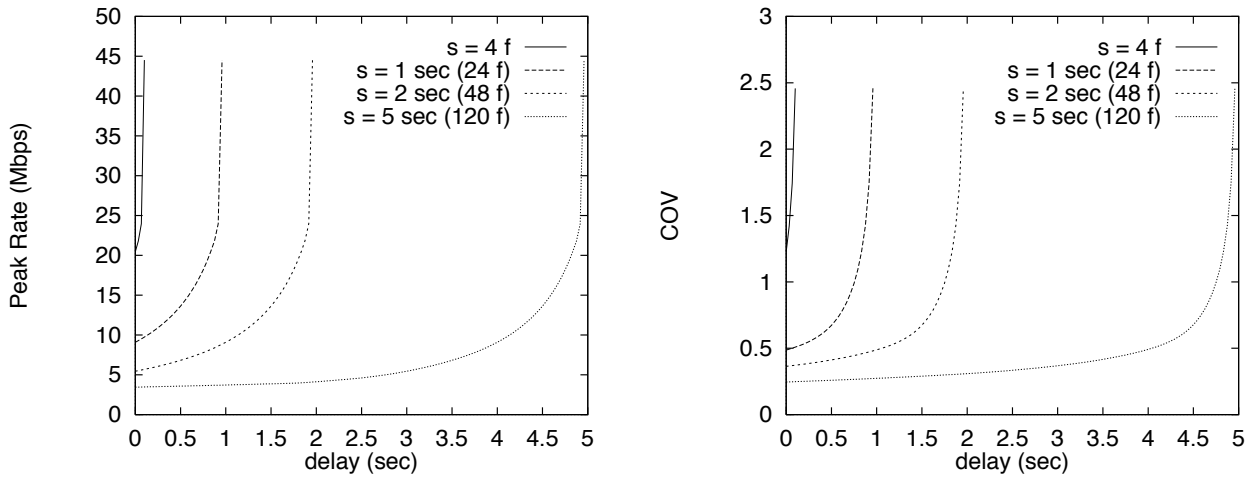
video. However, to accommodate jitter, the offline algorithm would have to use A_i^{min} for computing the upper constraint and A_i^{max} for computing the lower constraint, as outlined above. This could result in a somewhat conservative transmission schedule, particularly if the path between the server and the proxy has a large amount of jitter Δ. For more aggressive smoothing, the proxy can dynamically compute the schedule as frames arrive from the server. Each online computation applies the shortest-path algorithm to a sliding window of frames, following the approach in [31]. Since the online algorithm operates on a small sequence of frames, and does not have to execute on every frame arrival, the dynamic computation of the schedule does not consume significant processing resources, making it feasible for use at the proxy. In either case, note that our approach does not require a static buffer allocation based on worst case jitter, for de-jittering. Instead, the staging buffer b_s is used for both smoothing and as the jitter buffer.

4 Performance Evaluation

Caching a prefix of the incoming video permits a temporal decoupling of the arrival process at the proxy and the transmission to the client. Through simulation experiments with MPEG-1 and MPEG-2 traces, we demonstrate the performance benefits of having both a prefix buffer and a smoothing buffer at the proxy. The evaluation varies the parameters client startup delay s, server delay d, smoothing window w, prefix buffer b_p, and staging buffer b_s, and measures the impact on the peak rate and the coefficient of variation (standard deviation divided by the mean rate) of the resulting smoothed schedule. The results can help guide the selection of these parameters in a real system, to allow network service providers to maximize the benefits of the proxy prefix model.

4.1 Video Traces

The simulation experiments draw on two constant-quality MPEG video traces. MPEG streams consist of a mixture of I, P, and B frames. The I frames can be encoded and decoded independently. The P frames are coded using motion compensation from the preceding I (or P) frame, and B frames are coded using the preceding and succeeding I (or P) frame. The mixture of frame types introduces burstiness on a small time scale, while differences in motion and detail introduce burstiness at the time scale of scene changes. The first trace is an MPEG-1 encoding of a 23-minute segment of the movie *The Wizard of Oz*. The clip is encoded at 30 frames/second with a mean rate of 1.25 Mbits/second, peak rate of 10.9 Mbits/second, and a 15-frame MPEG



(a) Peak Rate

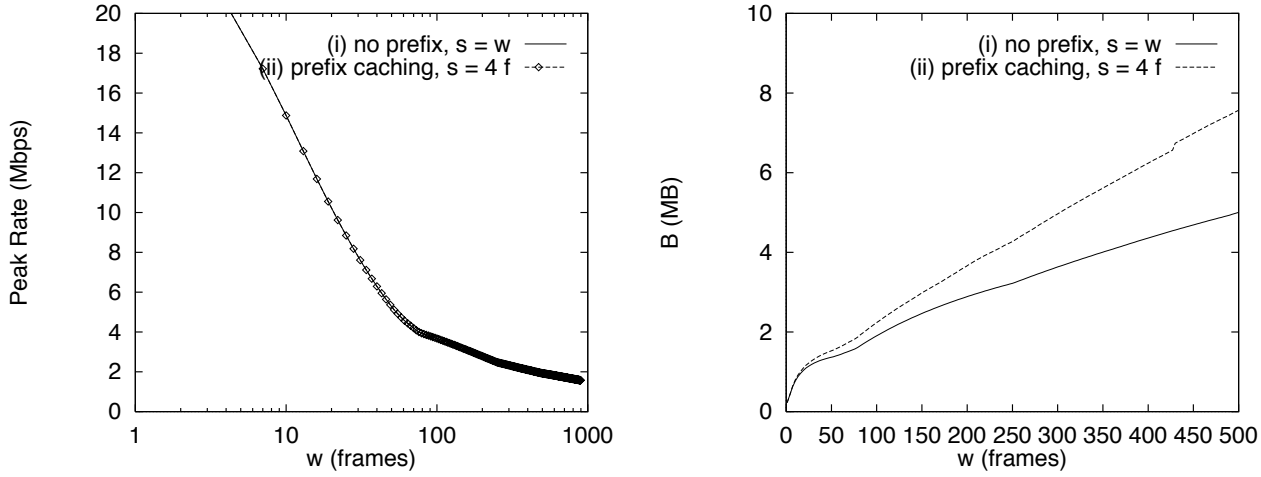
(b) Coefficient of Variation

Figure 4: These graphs plot the peak rate and coefficient of variation as a function of the server delay d for an MPEG-2 encoding of *Blues Brothers*, across different values of the client playback delay s. The proxy does not have a prefix cache (i.e., $b_p = 0$), and the client buffer size is $b_c = 8$ Mbytes.

group-of-pictures structure ($IBBPBBPBBPBBPBB$). The second trace is an MPEG-2 encoding of a 17-minute segment of the movie *Blues Brothers*. The clip is encoded at 24 frames/second with a mean rate of 1.48 Mbits/second, peak rate of 44.5 Mbits/second, and an irregular group-of-pictures structure ($IPPPPP\ldots$), with no B frames, and I frames only at scene changes. Note that a B frame is inter-frame coded, and the the preceding and succeeding I (or P) frame are required for decoding a B frame. As such, to guarantee starvation-free playback at the client, both preceding and succeeding I (or P) frames must arrive before the display deadline of a B frame. The playout vector needs to be constructed with this in view [32]. For *Wizard of Oz*, which has B frames, we use such a playout vector.

4.2 Smoothing Without a Prefix Buffer

In the absence of a prefix cache, the smoothing window is limited by the server delay and the client's willingness to tolerate playback latency. We consider a fixed server delay d with no jitter (i.e., $d = d_{max} = d_{min}$) and a client playback delay s, resulting in a smoothing window of $w = s - d$. Figure 4 plots the peak rate and the coefficient of variation as a function of the server delay for several different client delays. For a given playback delay s, moving from right to left in the graphs, we see that the peak rate and coefficient of variation decrease as the server delay d decreases, with substantial reductions when $d = 0$. For a given server delay, a larger playback delay gives the proxy more time to send each frame, resulting in a smoother transmission



(a) Peak Rate (x-axis logscale)

(b) Total proxy buffer (B)

Figure 5: These graphs plot the peak rate and total proxy buffer ($B = b_p + b_s^*$) as a function of the smoothing window for the MPEG-2 *Blues Brothers* clip. The experiments compare the performance of (i) $s = w$ and no prefix caching ($b_p = 0$) to (ii) $s = 4$ frames and the prefix caching is used to guarantee a w frame smoothing window. The playback buffer is $b_c = 8$ Mbytes and the server delay is $d = 0$.

schedule. For example, $d = 1$ and $s = 4$ result in a peak rate is 22 Mbits/second. The peak rate drops by more than a factor of two to 9.3 Mbits/second under a 24-frame (1-second) playback delay, and by almost another factor of two to 5.5 Mbits/second for a 2-second playback delay.

A smoothing window of several seconds offers substantial reductions in network resource requirements, as indicated by the y-intercepts of the graphs in Figure 4. However, in the absence of a prefix buffer, these gains are not achievable unless the client tolerates a relatively high playback latency. The use of a prefix cache is especially important when the client has a small delay tolerance, relative to the latency along the path from the server. When the server and playback delays are equal ($s = d$), the proxy cannot perform workahead smoothing and the stream has a peak rate of 44.5 Mbits/second. In addition to hiding server latency, a prefix buffer allows the proxy to perform workahead smoothing over a larger smoothing window. Previous studies have shown that, depending on the video, windows of several seconds to a few minutes offers significant reductions in the peak rate and coefficient of variation [12]. In the next subsection, we demonstrate that these gains are achievable in the proxy prefix model, even with a relatively small client playback delay.

4.3 Smoothing With a Prefix Buffer

The proxy can perform workahead smoothing over a w-frame interval by storing a prefix of the first

$d - s + w$ frames of the video. This is conceptually similar to smoothing without a prefix buffer, if the server can tolerate a delay of $s = w + d$ frames. However, there is a difference between the two scenarios. In the latter case, i.e., when the client can tolerate the higher latency $s = w + d$, the proxy has lower storage requirements (no prefix buffer, only the staging buffer), and more time to deliver the initial frames of the stream. Figure 5(a) compares a system with a w-frame playback delay and no prefix buffer to a system with an 4-frame playback delay and a $(w-4)$-frame prefix buffer; both scenarios assume a server delay of $d = 0$. Despite the differences between the two configurations, the graph shows that the performance is indistinguishable (the coefficient of variation also was very similar). For example, a window of $w = 148$ frames (6 seconds) decreases the peak rate to 3.2 Mbits/second, compared to 20.4 Mbits/second for a 4-frame window. Also, larger windows offer even further reductions. We observed the same behavior with the *Wizard of Oz* trace. Note that the very similar performance in spite of a small client delay of four frames ($1/6$ of a second) suggests that either this small startup delay was sufficient to smooth the first frame in the video, or the peak rate occurred later on in the video. However, realistic values of s will typically in the range of 0.5 seconds to a couple of seconds, much larger than the 4 frame delay considered here. Hence, although there may be pathological cases where the peak rate is dominated by the first frames in the video, the proxy would have sufficient time to smooth the initial frames of the stream. Hence in reality, we expect the performance with the prefix caching to be quite close to what can be achieved with the longer startup delay $s = w + d$.

The graph in Figure 5(a) also shows that increasing the window size offers dramatic reductions in the peak bandwidth. However, a larger window also increases the buffer requirements at the proxy, as shown in Figure 5(b). The graph plots the total proxy buffer requirement for the two scenarios. In both scenarios, the size of the smoothing buffer is determined by first computing the optimal transmission schedule $\mathbf{S}$ without any constraint on the buffer size. The actual buffer occupancy profile as a function of time ($0 \leq i \leq N + w$) can be then computed as follows:

$$b_{s,i} = \begin{cases} A_i - b_p, & S_i < b_p \\ A_i - S_i, & S_i \geq b_p \end{cases}$$

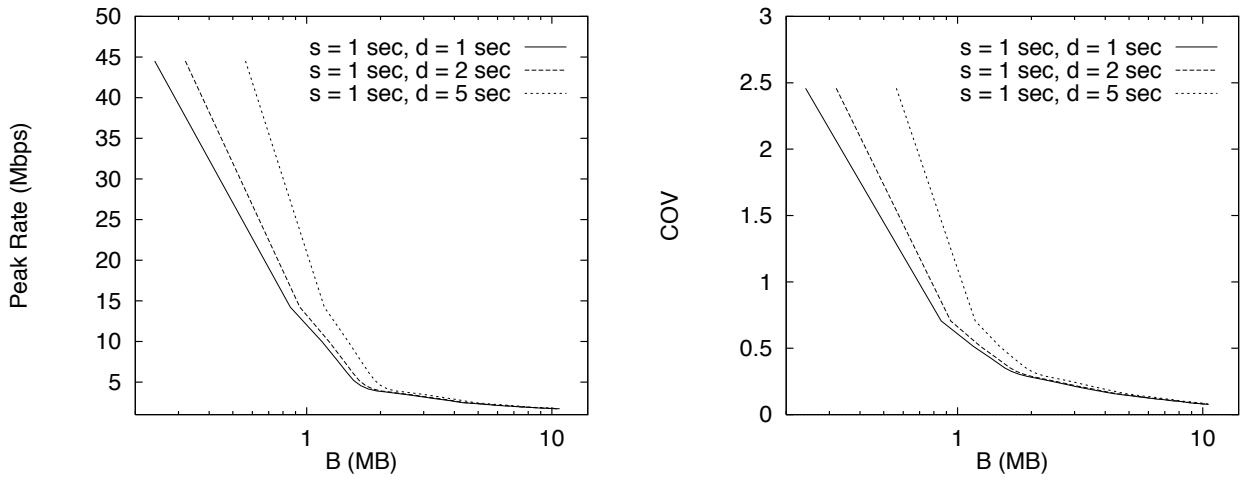
Then the worst-case staging buffer requirement is $b_s^* = \max_i(b_{s,i})$. The total buffer requirement is then the computed b_s^* for Case (i) (in the absence of a prefix buffer) and $b_p + b_s^*$ for Case (ii) (in the presence of a prefix buffer). The graphs in Figure 5(b) show that both scenarios can achieve the benefits of workahead smoothing with a relatively small proxy buffer. For example, a 9-second smoothing window requires a 3-Mbyte smoothing

buffer; the size increases to 3.9 Mbytes if the proxy also stores the prefix of the stream. Devoting a 3–5 Mbyte proxy buffer to each video stream is quite reasonable, particularly given the low costs of memory and disk space. Although the difference in the buffer requirements increases with w, the total buffer size is reasonable for practical window sizes. Once the window size grows beyond a few hundred frames, workahead smoothing begins to offer diminishing returns, as shown in Figure 5(a).

To illustrate the trade-off between buffer space and smoothing gain, Figure 6 plots the peak rate and coefficient of variation as a function of the buffer size $B = b_p + b_s^*$, for three different values of the server delay d. For each combination of values of s and d, we vary the smoothing window w as an independent parameter. For each value of w, we plot the corresponding peak rate (coefficient of variation) versus the corresponding total buffer allocation to arrive at these plots. Large values of d require more buffer space for the same reduction in the peak rate, due to the extra prefix storage requirement to hide the server delay from the client. The graphs show that dramatic smoothing gains can be obtained with a few Mbytes of proxy buffer, even while hiding a server delay that is several times larger than the playback delay. For example, for $d = 5$ seconds, a 2-Mbyte proxy buffer reduces the peak rate and coefficient of variation to 4 Mbits/second and 0.33 respectively from the unsmoothed values of 44.5 Mbits/second and 2.50. In addition, the performance is very similar across different values of d once the proxy has a few Mbytes of buffer space. This suggests that it is practical to inflate the server delay d to handle retransmissions at the proxy, making it possible to improve the quality of video transmissions on the Internet, as discussed in Section 2.2.

4.4 Proxy Prefix and Smoothing Buffers

After dedicating a certain amount of buffer space to a video transmission, the proxy must select a window size w and, in turn, divide the buffer space between the prefix and smoothing buffers. The proxy must balance the trade-off between allocating a larger prefix buffer b_p (which will permit smoothing over a larger window w of frames), and a large smoothing buffer b_s (having enough storage to absorb transient bursts of large frames). The graph in Figure 7(a) plots the peak rate as a function of w for a buffer of size M, for *Wizard of Oz*. As the window size increases, the peak rate decreases, first rapidly, then more slowly, before ultimately increasing. To explain this trend, Figure 7(b) plots the resulting size of the smoothing buffer ($b_s = M - b_p$) and the maximum amount of space consumed by the resulting transmission schedule (b_s^*). We observe that b_s^* initially increases with w and, in this increasing region, b_s^* is less than the allocated b_s. As w increases further, b_s^* becomes identical to b_s and both then decrease together. The graphs also show that the peak rate decreases in



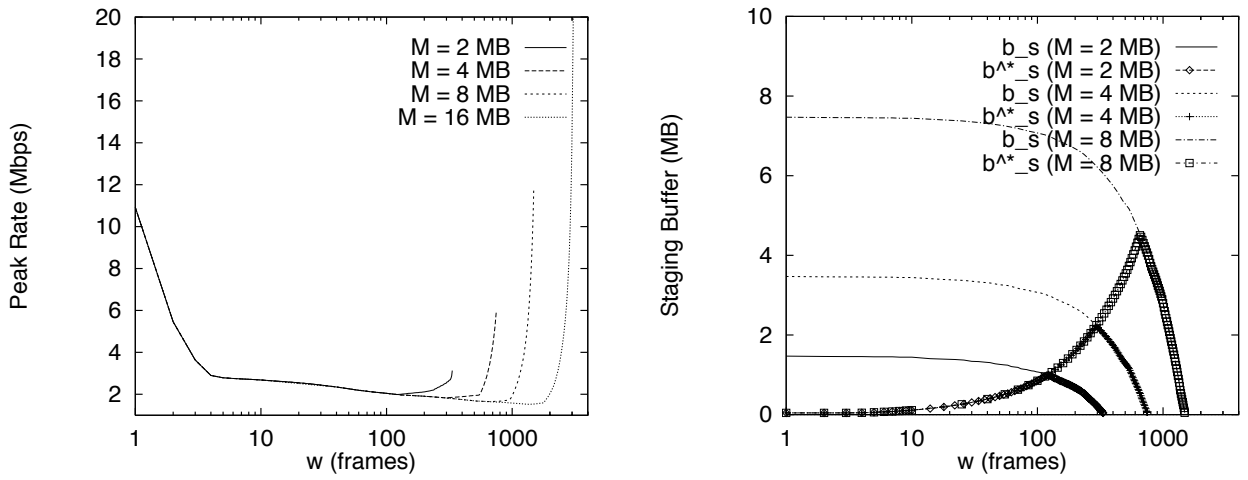
(a) Peak Rate

(b) Coefficient of Variation

Figure 6: These graphs plot the peak rate and coefficient of variation as a function of the total proxy buffer $B = b_p + b_s$, for different values of the server delay d and client startup delay s, for MPEG-2 *Blues Brothers*. (client buffer size $b_c = 8$ Mbytes.)

the increasing region of b_s^* and starts increasing when b_s^* starts decreasing. All this suggest the following system dynamics. Small window sizes result in small values for both b_p and b_s^*, such that $b_p + b_s^*$ is substantially less than M. As w increases, the peak rate initially decreases as the proxy can smooth across a larger window of frames. In this region, the allocated b_s is larger than what is required. However, as w grows, the prefix buffer size b_p also increases, and reduces the space available for the smoothing buffer. At the same time, a larger w will result in frames arriving earlier at the proxy staging buffer (w time units ahead of their consumption deadlines at the client). As w continues to increase, the smoothing buffer space b_s becomes sufficiently small that it imposes a constraint on transmissions to the client. As a result, the proxy must transmit aggressively to avoid overflowing the proxy smoothing buffer. These workahead transmissions are detrimental to smoothing, and actually increase the peak bandwidth. In the worst case, the transmission schedule has a higher bandwidth requirement than the original unsmoothed stream.

Since b_p is an increasing function of w, increasing w results in smaller values of b_s. The performance degradation is especially dramatic for small b_s, since a larger window w decreases b_s precisely when a larger smoothing buffer is necessary. Figure 8 plots the peak rate and coefficient of variation as a function of the prefix buffer size, for several total buffer allocations M. As in Figure 7(a), the metrics initially decrease rapidly before flattening, and then increase dramatically. For example, for a total buffer budget of $M = 8$ Mbytes, the peak rate decreases from 10.9 Mbits/second for $b_p = 0.5$ Mbytes to 2.2 Mbits/second for a mere 0.2-Mbyte



(a) Peak Rate

(b) Smoothing Buffer

Figure 7: These graphs plot the peak rate and the size of the smoothing buffer (b_s and b_s^*) as a function of the smoothing window w, for different total buffer budget M. The experiment evaluates the MPEG-1 *Wizard of Oz* clip with a 1-second start-up delay, a 5-second server delay, and a 32-Mbyte client buffer.

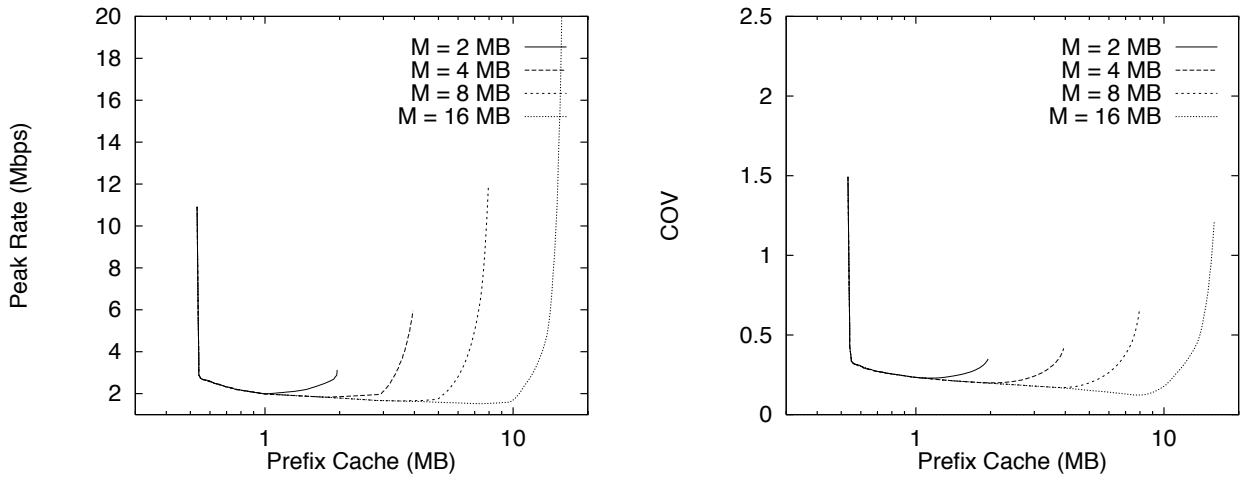
increase in b_p; in this region, the smoothing buffer b_s is sufficiently large that it does not impose any constraint on smoothing. This underlines the importance of operating in the middle region of the graphs, to the left of where the peak rate starts increasing with increasing b_p, and to the right of where the peak rate reduction is substantial. In each of the curves, the performance degrades rapidly when b_p grows larger than $M/2$, suggesting a symmetric allocation of the prefix and smoothing buffers.

5 Multiple Clients and Multiple Streams

Supporting multiple clients and a collection of continuous media streams requires effective techniques for allocating buffer and bandwidth resources at the proxy. Drawing on the results in Section 4, we describe how to balance the trade-off between buffer and bandwidth resources in handling multiple streams and multiple clients. Then, we discuss how to achieve further reductions in buffer requirements by sharing a common smoothing buffer among multiple independent client streams.

5.1 Buffer and Bandwidth Allocation

The smoothing model in Section 3 and the performance evaluation in Section 4 each focus on the resource requirements for transmitting a single stream. In a broader context, the proxy needs an effective way to allocate



(a) Peak Rate

(b) Coefficient of Variation

Figure 8: These graphs plot the peak rate, and coefficient of variation as a function of the prefix buffer b_p, for different values of the total proxy buffer budget M. The experiment evaluates the MPEG-1 *Wizard of Oz* clip with a 1-second start-up delay, a 5-second server delay, and a 32-Mbyte client buffer.

buffer and bandwidth resources across *multiple* streams with different popularities. In general, a popular stream should have a larger smoothing window than other streams, since the overhead of the prefix buffer can be amortized over a larger number of clients. However, the problem is complicated by the diverse characteristics of video streams, which affect how much a larger smoothing window and/or a larger smoothing buffer can reduce the network bandwidth requirements. We consider a system with V streams, where stream v is selected by n^v different clients, $v = 1, 2, \ldots, V$; typically, the $\{n^v\}$ follow a Zipf's law, with a small number of popular streams accounting for most of the requests. We assume that the clients do not necessarily arrive close together in time. Hence, clients of the same stream do not share a common smoothing buffer, though they do share the prefix buffer. In addition, we assume that the playback delay, server delay, and client buffer size are given, and are fixed for each stream.

The proxy allocates bandwidth for each stream based on the peak rate in the smoothed schedule, and allocates a fixed smoothing buffer to each client of each stream. This suggests two basic problems:

- Bandwidth allocation: Minimize the bandwidth requirement B, subject to a proxy buffer space constraint of M bits.
- Buffer allocation: Minimize the buffer allocation M, subject to a bandwidth constraint of B bits/second.

Despite the complex nature of these two optimization problems, the experimental results offer several insights that allow us to develop effective heuristics. The first observation concerns the selection of the smoothing buffer

size b_s. The experiments in Section 4.4 show that allocating less than b_s^* introduces significant performance degradation. Hence, we assume that $b_s = b_s^*$, which can be computed directly from the smooth transmission schedule once w is determined. The second observation concerns the diminishing returns of larger smoothing buffers and smoothing windows. These trends are apparent in Figures 5(a) and 7(a), respectively, even when plotted on a logarithmic scale. This property has two important implications. First, for a given stream, the proxy should not allocate different values of w and b_s to different clients. Assigning $w + 1$ to one client and $w - 1$ to another client would not offer as much total reduction in the network bandwidth requirements. In addition, the proxy must allocate the prefix buffer with the largest client window size in mind, further arguing for an identical allocation across requests for the same stream.

Since b_s and b_p are determined directly from w, and each client has the same allocation for each stream, we have narrowed the problem to selecting the value of w^v for stream $v = 1, 2, \ldots, V$. Any choice of the w^v values results in some total buffer allocation of $\sum_v (b_p^v + n^v b_s^v)$ and a total bandwidth allocation of $\sum_v n^v p^v$, assuming that the clients do not share buffer or bandwidth resources. The allocation of the buffer and bandwidth resources is simplified by the convex shape of the peak-rate curves. For example, suppose we are trying to minimize the buffer space M subject to an upper bound on the bandwidth B. Then, the proxy can first allocate memory to the stream v that offers the most dramatic reduction in the bandwidth requirement $n^v p^v$. The appropriate choice is the stream v with the steepest curve for the peak rate as a function of the buffer space ($n^v p^v$ vs. $b_p^v + n^v b_s^v$). The resource allocation heuristic continues in a greedy fashion, always selecting the stream that offers the largest incremental reduction in the peak bandwidth for the increase in buffer space.

The algorithm continues to incrementally increase the window sizes for the streams until the bandwidth constraint B is reached. The greedy approach favors streams with more opportunities for smoothing gain, since these streams offer greater reductions in bandwidth requirements for the same buffer allocation. Similarly, the greedy approach favors more popular movies (larger n^v values), since the multiple clients amortize the cost of the prefix buffer. Since smoothing offers the most dramatic returns with small window sizes, the greedy algorithm would typically allocate some minimum window and prefix buffer to each stream to allow the proxy to remove the substantial short-term burstiness. Then, only more popular or more bursty streams would have large windows to further decrease their bandwidth requirements. A similar approach can be applied to allocate bandwidth resources subject to an upper bound on the total size M of the proxy buffer.

5.2 Buffer Multiplexing

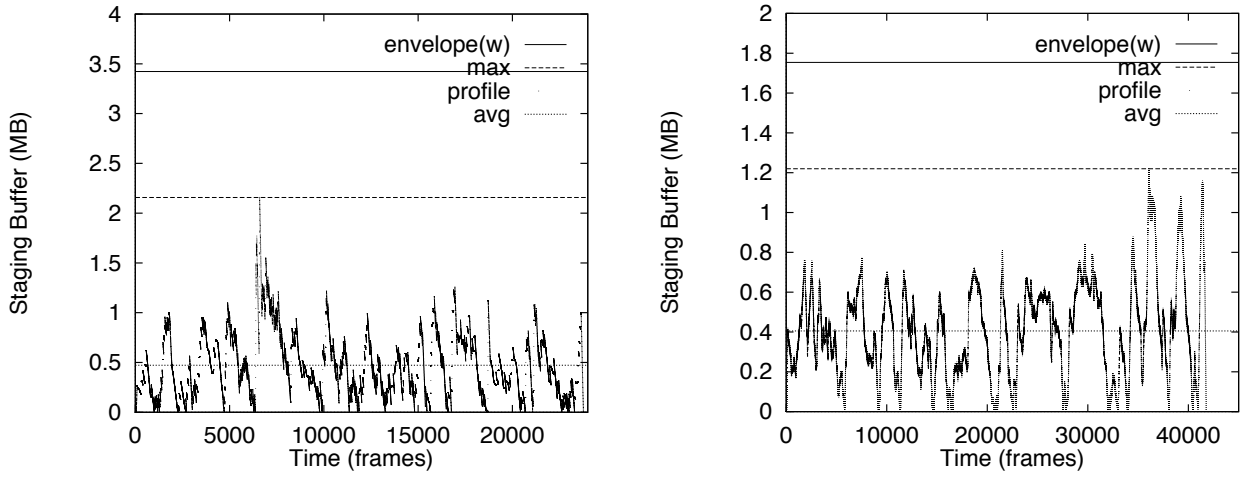
Although prefix caching and workahead smoothing only require a few megabytes at the proxy, the resource requirements increase as the proxy handles multiple streams and multiple clients. For a more efficient allocation of proxy buffer space, we consider techniques for sharing the prefix and smoothing buffers. Inherently, the prefix buffer can be shared across successive requests for the same stream. However, unless client requests arrive close together in time, it is difficult to share the contents of the smoothing buffer. Instead, we consider how to multiplex the smoothing buffer space across several independent streams. For an individual stream, the utilization of the smoothing buffer changes across time, as shown in the examples in Figure 9. The utilization profile $b_{s,i}$ is determined by computing an optimal transmission schedule without a constraint on the size of the smoothing buffer. For comparison, the graphs also plot

- Envelope: maximum size of w frames ($\max_j\{\sum_{i=j}^{j+w-1} f_i\}$)
- Maximum: maximum smoothing buffer requirement ($\max_i\{b_{s,i}\}$)
- Average: average smoothing buffer requirement (avg$\{b_{s,i}\}$)

all for a window of $w = 5$ seconds. The graphs show a significant difference between the envelope and the worst-case buffer requirement. When the proxy encounters the largest set of frames in the stream, workahead transmission ensures that some of these frames have been transmitted to the client in advance of the burst.

Exploiting frame-size information offers a significant reduction in buffer requirements over a simple characterization of the traffic envelope. In fact, even the maximum buffer usage (b_s^*) is a fairly conservative estimate of resource requirements, since the peak utilization occurs for a very short period of time. These trends persist across a range of window sizes, as shown in Figure 10. The temporal fluctuations suggest that that multiplexing of the proxy buffer space can offer a substantial reduction in the resource requirements. These benefits are attainable even if clients do not issue requests at the same time or for the same stream. In addition, this multiplexing occurs in a deterministic fashion, based on the buffer requirements of the various streams across time. Sharing the buffer space amongst multiple clients and streams can substantially reduce the resource requirements, particularly when the various streams do not all experience their worst-case buffer requirements at the same time.

To exploit these potential gains, the proxy maintains a profile of the allocated buffer space across time, based on the transmission schedules of the current streams. Instead of storing information for every frame slot,



(a) *Blues Brothers* (MPEG-2)

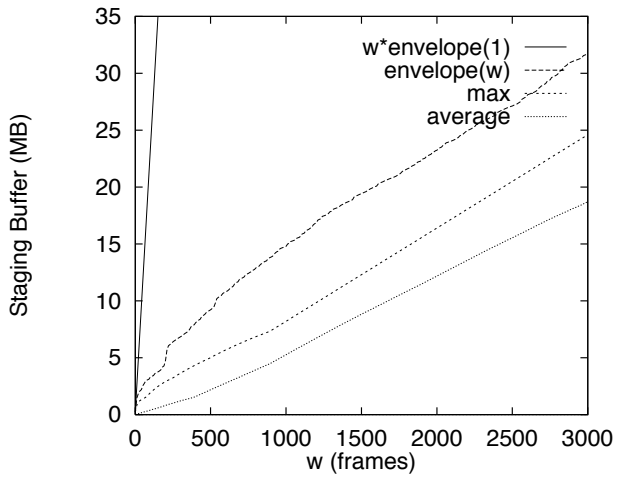
(b) *Wizard of Oz* (MPEG-1)

Figure 9: These graphs plot the utilization of the smoothing buffer as a function of time for a 5-second smoothing window, with $b_c = 32$ Mbytes, $s = 0.5$ seconds, and $d = 2$ seconds.

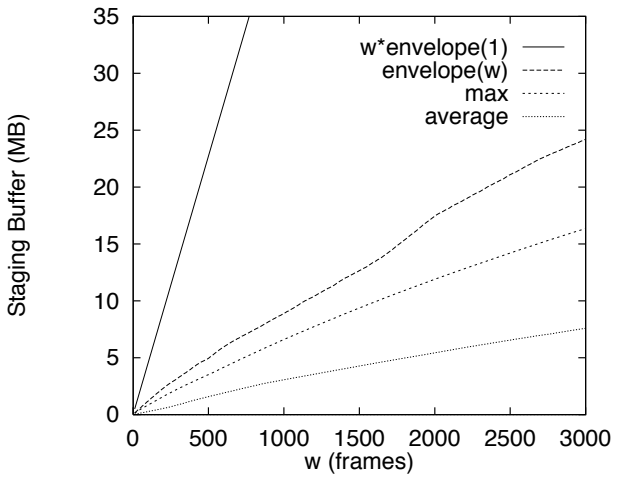
the proxy can track the buffer requirements over coarser time scale, with some trade-off in multiplexing gain. Upon receiving a new client request, the proxy determines whether or not it can accommodate the additional buffer requirements $b_{s,i}$ across the duration of the stream. If so, the proxy accepts the request and updates the view of the available buffer resources. To increase the likelihood of accepting requests, the proxy could compute the schedule for the new stream, based on the time-varying buffer resources in the system. Instead of applying a precomputed transmission schedule, the proxy would compute S_i based on new smoothing constraints that capture the limitations on b_s across time. If buffer resources are especially constrained, the proxy could conceivably select a smaller window size w for the new request, though the bandwidth requirements for transmissions to that client will be higher. The cost-performance trade-offs of these proxy policies can be evaluated as part of future work.

6 Conclusions

Network service providers can reduce response time, server load, and traffic load by deploying proxy caches. Existing techniques for caching text and images are not appropriate for the rapidly growing number of continuous media streams. In addition, high latency and loss rates in the Internet make it difficult to stream audio and video without introducing a large playback delay at the end-client. To address these problems, we proposed that, instead of caching entire audio or video streams (which may be quite large), the proxy should store a prefix



(a) *Blues Brothers*



(b) *Wizard*

Figure 10: These graphs plot the size of the proxy smoothing buffer as a function of the window size w for $s = 0.5$ seconds, $d = 2$ seconds, and $b_c = 32$ Mbytes.

consisting of the initial frames of popular clip. In addition to hiding the effects (loss, delay, jitter, etc.) of a weaker service model between the server and the proxy, prefix caching aids the proxy in performing workahead smoothing of variable-bit-rate streams. Smoothing substantially reduces the resource requirements along the path from the proxy to the client. This is particularly important when the client has relatively limited access bandwidth, compared to the higher-speed backbone network.

As future work, we plan to explore the multiple-client bandwidth and buffer allocation problem in greater depth. Our initial experiments illustrate the significant potential for sharing the smoothing buffer across multiple streams. Based on these results, we are investigating and evaluating policies for online resource allocation as client requests arrive. In addition, we are studying the effects of variable delay on the path between the server to the proxy, including how to estimate jitter and incorporate these measurements in the smoothing constraints. Also, we are evaluating whether or not computing the transmission schedule dynamically at the proxy, based on the actual sequence of arrivals from the server, offers substantial gains over using a precomputed schedule. This ongoing work extends the prefix caching and smoothing model presented in this paper, and can aid in the design of effective proxy services for streaming high-quality continuous media in the Internet.

References

[1] A. Bestavros, R. L. Carter, and M. E. Crovella, "Application-level document caching in the Internet," in *Proc. Inter. Workshop on Services in Distributed and Networked Environments*, June 1995. http://www.cs.bu.edu/~best/res/papers/sdne95.ps.

[2] S. Williams, M. Abrams, C. R. Standbridge, G. Abdulla, and E. A. Fox, "Removal policies in network caches for World Wide Web documents," in *Proc. ACM SIGCOMM*, pp. 293–305, August 1996. http://www.acm.org/sigcomm/sigcomm96/papers/williams.html.

[3] P. Cao and S. Irani, "Cost-aware WWW proxy caching algorithms," in *Proc. USENIX Symp. on Internet Technologies and Systems*, pp. 193–206, December 1997. http://www.cs.wisc.edu/~cao/papers/gd-size.html.

[4] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1, request for comments 2068," January 1997. ftp://ftp.isi.edu/in-notes/rfc2068.txt.

[5] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP), request for comments 2326," April 1998. ftp://ftp.isi.edu/in-notes/rfc2326.txt.

[6] E. P. Rathgeb, "Policing of realistic VBR video traffic in an ATM network," *International Journal on Digital and Analog Communication Systems*, vol. 6, pp. 213–226, October–December 1993.

[7] M. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. ACM SIGCOMM*, September 1994.

[8] A. R. Reibman and A. W. Berger, "Traffic descriptors for VBR video teleconferencing over ATM networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 329–339, June 1995.

[9] M. Grossglauser, S. Keshav, and D. Tse, "RCBR: A simple and efficient service for multiple time-scale traffic," *IEEE/ACM Trans. Networking*, December 1997.

[10] M. Krunz and S. K. Tripathi, "On the characteristics of VBR MPEG streams," in *Proc. ACM SIGMETRICS*, pp. 192–202, June 1997.

[11] T. V. Lakshman, A. Ortega, and A. R. Reibman, "Variable bit-rate (VBR) video: Tradeoffs and potentials," *Proceedings of the IEEE*, vol. 86, May 1998.

[12] J. Rexford, S. Sen, J. Dey, W. Feng, J. Kurose, J. Stankovic, and D. Towsley, "Online smoothing of live, variable-bit-rate video," in *Proc. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 249–257, May 1997. http://www.research.att.com/~jrex/papers/nossdav97.ps.Z.

[13] M. Kamath, K. Ramamritham, and D. Towsley, "Continuous media sharing in multimedia database systems," in *Proc. of 4th International Conference on Database Systems for Advanced Applications (DASFAA'95)*, April 1995. http://www-ccs.cs.umass.edu/~kamath/DASFAA95.ps.

[14] A. Dan and D. Sitaram, "Multimedia caching strategies for heterogeneous application and server environments," *Multimedia Tools and Applications*, vol. 4, pp. 279–312, May 1997.

[15] R. Tewari, H. M. Vin, A. Dan, and D. Sitaram, "Resource-based caching for Web servers," in *Proc. SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998. http://www.cs.utexas.edu/users/tewari/psfiles/mmcn98.ps.

[16] W. Feng and S. Sechrest, "Smoothing and buffering for delivery of prerecorded compressed video," *Computer Communications*, vol. 18, pp. 709–717, October 1995.
http://www.cis.ohio-state.edu/~wuchi/CompComm.ps.gz.

[17] W. Feng, F. Jahanian, and S. Sechrest, "An optimal bandwidth allocation strategy for the delivery of compressed prerecorded video," *Springer-Verlag Multimedia Systems Journal*, vol. 5, pp. 297–309, September 1997.
http://www.cis.ohio-state.edu/~wuchi/mmsj.ps.gz.

[18] J. D. Salehi, Z.-L. Zhang, J. F. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," in *Proc. ACM SIGMETRICS*, pp. 222–231, May 1996.
ftp://gaia.cs.umass.edu/pub/Sale96:Supporting.ps.gz.

[19] J. M. McManus and K. W. Ross, "Video on demand over ATM: Constant-rate transmission and transport," in *Proc. IEEE INFOCOM*, pp. 1357–1362, March 1996. Extended version appears in *IEEE J. Selected Areas in Communications*, August 1996, pp. 1087–1098.

[20] W. Feng and J. Rexford, "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video," in *Proc. IEEE INFOCOM*, pp. 58–66, April 1997.
http://www.research.att.com/~jrex/papers/infocom97a.ps.Z.

[21] Y. Wang, Z.-L. Zhang, D. Du, and D. Su, "A network conscious approach to end-to-end video delivery over wide area networks using proxy servers," in *Proc. IEEE INFOCOM*, April 1998.
http://www.cs.umn.edu/~zhzhang/Papers/Wang98:Info98.ps.gz.

[22] E. Amir, S. McCanne, and H. Zhang, "An application level video gateway," in *Proc. ACM Multimedia*, November 1995.
http://www.cs.berkeley.edu/~elan/pubs/papers/vgw.ps.

[23] N. F. Maxemchuk, K. Padmanabhan, and S. Lo, "A cooperative packet recovery protocol for multicast video," in *Proc. International Conference on Network Protocols*, October 1997.
http://www.research.att.com/~nfm/ref.1463.ps.

[24] "Cisco cache engine."
http://www.cisco.com/warp/public/751/cache.

[25] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications, request for comments 1889," January 1996.
ftp://ftp.isi.edu/in-notes/rfc1889.txt.

[26] N. F. Maxemchuk and S. Lo, "Measurement and interpretation of voice traffic on the Internet," in *Proc. International Conference on Communications*, June 1997.
http://www.research.att.com/~nfm/ref.1443.ps.

[27] V. Paxson, "End-to-end Internet packet dynamics," in *Proc. ACM SIGCOMM*, pp. 139–152, September 1997.
ftp://ftp.ee.lbl.gov/papers/vp-pkt-dyn-sigcomm97.ps.Z.

[28] D. Sanghi, A. K. Agrawala, O. Gudmundsson, and B. N. Jain, "Experimental assessment of end-to-end behavior on Internet," in *Proc. IEEE INFOCOM*, March/April 1993.

[29] S. Acharya and B. Smith, "An experiment to characterize videos on the World Wide Web," in *Proc. SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998.

[30] I. Dalgic and F. A. Tobagi, "Performance evaluation of ATM networks carrying constant and variable bit-rate video traffic," *IEEE J. Selected Areas in Communications*, vol. 15, August 1997.

[31] J. Rexford and D. Towsley, "Smoothing variable-bit-rate video in an internetwork," in *Proc. SPIE Symposium on Voice, Video, and Data Communications: Multimedia Networks: Security, Displays, Terminals, and Gateways*, November 1997.
http://www.research.att.com/~jrex/papers/spie97.ps.Z.

[32] S. Sen, J. Rexford, J. Dey, J. F. Kurose, and D. Towsley, "Online Smoothing of Variable-Bit-Rate Streaming Video," Tech. Rep. 98-75, Department of Computer Science, University of Massachusetts, 1998.