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

Abstract— **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 a *prefix caching* technique whereby a proxy stores the initial frames of popular clips. Upon receiving a request for the stream, the proxy initiates transmission to the client and simultaneously requests the remaining frames from the server. In addition to hiding the delay, throughput, and loss effects of a weaker service model between the server and the proxy, this novel yet simple prefix caching technique 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. Experiments with MPEG traces show how a few megabytes of buffer space at the proxy can substantially reduce 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**

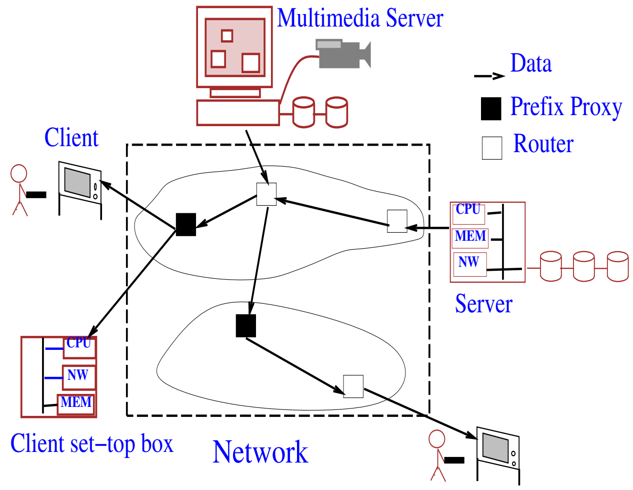


Fig. 1. Prefix proxy in an internetwork

I. 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, 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. 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 an initial time duration (which can be of the order of a few round-trip delays to the server, depending on the startup protocol and also server latency), 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.

Instead, we propose that service providers can deploy multimedia proxies along the path from the server to the client, as shown in Fig. 1. Similar to traditional caching of text and image data, storing (part of) each audio/video stream enables the proxy to reduce client delay without sacrificing quality. Upon receiving a client request, the proxy 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 II 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 [1] and RTSP (Real-Time Streaming Protocol) [2].

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 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 [3, 4]. The transmission can become even more bursty when one or more video sources are combined with text, audio, and images in 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 by performing *workahead smoothing* into the client playback buffer. Transmitting large frames in advance of each burst can substantially reduce the peak rate and variability of the video stream [5]. Smoothing at the proxy would typically introduce delay, and require 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 playback latency. To aid in computing the transmission schedule, the proxy can store frame-size information for the video stream.

In Section III, we present a detailed model of workahead smoothing in the presence of a prefix buffer, and show how to compute smoothed transmission schedules. Section IV 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, Section V introduces 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 streams. Section VI concludes the paper 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 [6–8]. 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 [9–11]. 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 [12, 13]. 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 [5], 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 [14]. 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 [15, 16].

II. Prefix Caching

By storing the first few seconds of popular video clips, a multimedia proxy can shield users from the delay, throughput, and loss properties of the path between the server and the proxy. We first present an overview of Internet performance characteristics to motivate the use of prefix caching. Then, we describe how the proxy can store the initial frames of a video, and retrieve the remainder of the clip using existing Web and real-time streaming primitives.

A. Internet Performance Characteristics

The best-effort service model of traditional IP networks undermines the effort to stream high-quality video across the Internet in real time. The rapid growth in the popularity of the Internet exacerbates the problem by increasing the average delay, as well as the variability of the latency, throughput, and loss, along the path between the server and the client. The poor performance also stems from the many steps involved in a typical download of a video clip via the Web [17]. For example, clicking on a hypertext link for a video clip causes the Web browser to establish a TCP connection to the server, incurring the network round-trip delay, plus latency at the server to open the connection. Additional delay is experienced if the browser must first perform a DNS (domain name service) query to learn the server's IP address. Then, the browser issues a HTTP request, and the server typically responds with a small Web resource which contains a pointer to the actual video clip.

The browser forks a multimedia player that contacts the video server, which may or may not be the same machine as the Web server. If the server streams the video over a TCP connection, because the client resides behind a firewall that filters UDP packets, then the browser must establish a TCP connection to the server before sending the actual request for the video. Then, the server initiates the video transmission, perhaps starting with a test sequence to learn the achievable throughput along the path to the client. Before starting to display the video clip, the client player accumulates enough frames to tolerate delay jitter that may occur in the future. Without a sufficiently large playback delay, the player may be required to disrupt the playout of the video stream during periods of transient network congestion. In the end, video streaming typically involves three or four network round-trip times, processing at the server and client, and accumulation of a client de-jitter buffer.

Although new multimedia protocols may be able to circumvent some of these steps, the delay in contacting the server and accumulating a de-jitter buffer is difficult to avoid without introducing a multimedia proxy. Recent Internet measurements lend insight into how long the video playback latency would be in realistic settings. Round-trip times vary widely, but delays of several seconds are not uncommon [18, 19]. Similarly, Internet packet loss rates range from 2–10% [18–20]. Full motion video requires the transmission of 30-frames every second — one frame of $15 - 30$ packets every 33 milliseconds. Because the packets are sent back-to-back, even transient network congestion could result in the loss of several consecutive packets in the video stream. Retransmission schemes could recover lost packets, at the expense of additional round-trip delays.

Packet loss also has a direct impact on throughput if the video streaming application runs over TCP, or some other "TCP-friendly" transport protocol. With the phenomenal growth of Web traffic, and the emergence of audio and video streaming applications, pressure is mounting for Internet multimedia transport protocols to employ congestion-control policies that share resources fairly with traditional TCP traffic. Viewing packet loss as an indication of network congestion, the video server should react by decreasing its sending rate (lowering its window size). However, the smaller available bandwidth may not permit the server to continue transmitting the video stream at the same quality level and playback delay. To avoid potential disruptions in the video playout, the player could introduce an even larger de-jitter buffer (and playback delay) to tolerate these transient fluctuations in available bandwidth. Alternatively, the server could lower its transmission rate by decreasing the quality of the video encoding.

B. Proxy Prefix Size

To shield the client from the poor performance properties of the Internet, network service providers could instantiate multimedia proxies that store the initial frames of popular video clips. Proxy prefix caching provides a way for a service provider to provide high-quality video to its clients with low playback delay, even for video streams that originate in other networks. Prefix caching also enables service providers to capitalize on quality-of-service support along the path between the proxy and the client, even if the path between the server and the proxy has a weaker service model. Similarly, while the server and the proxy may communicate via TCP, or some other TCP-friendly transport protocol, the proxy could use a different protocol (such as UDP, or even ATM) to transmit to the client. This is possible because the proxy may be a trusted entity, to both the service provider and the client machine/network, and because the service provider has control of the entire path between the proxy and the client.

The size of the video prefix depends on the performance properties of the path between the server and the proxy, as well as the target client playback delay. In general, the prefix should be large enough for the proxy to tolerate a few round-trip delays, to absorb server-proxy latency, retransmission of lost packets, and the variability of TCP throughput. 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%, which would substantially improve video quality at the client. In total, a proxy might store several seconds of the video stream. Even for a high-bandwidth MPEG-2 clip, storing 5 seconds would only require around 2.5–3 Mbytes of buffer space, which is reasonable given today's memory and disk prices. If the prefix is stored on disk, the first few frames could be cached in memory to hide the disk access latency.

The proxy must also devote buffer space for incoming packets. For example, 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 support discontinuity-free playback with 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. For some video clips, the proxy may also cache other portions of the stream. For example, the proxy could store the beginnings of particular scenes or chapters of the video, particularly for orchestrated media streams that support browsing and restricted jump operations. This would allow the proxy to support such coarse-timescale interactivity without increasing client playback delay. Arbitrary VCR functions, such as fast-forward and rewind, would still require increasing the playback delay, except when the desired video frames reside in the proxy cache or the client buffer. Small time-scale interactivity is most efficiently handled directly at the client, without incurring additional playback delay.

Although the proxy does not store the entire video stream, prefix caching still reduces the total amount of traffic between the server and the proxy, particularly for short video clips. A 1997 study found that Web video streams have a median size of 1.2 MBytes, with 90% of the streams lasting less than 45 seconds [21]. Although the size and duration of video 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. If the video cache is integrated with the regular Web data cache, the ability to select arbitrary prefix sizes gives greater flexibility to the cache replacement algorithm.

C. Transparent Operation

The proxy must reside on the path between the client and the server. For example, the client browser and/or player can be configured to transmit traffic through the proxy, as is common in the Web today. 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. This approach is analogous to the notion of transparent caching in the Cisco Cache Engine [22]. In either scenario, the proxy can 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 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. The push model of prefix distribution could be used even when the proxy belongs to a different service provider, if the two providers have a prior agreement to distribute the prefix. This may be a useful model for live video clips, where the proxy could store short prerecorded trailers or lead-ins (e.g., interviews, commentaries, or advertisements before a live sporting event).

For successful deployment, the proxy must have an effective way to identify and store the sequence of initial frames, and request the remaining frames from the server, without requiring modifications to the server sites. For example, Real-Time Protocol (RTP) [23] 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 *range request* operation in HTTP 1.1 [1] to request the appropriate portion of the stream. Similarly, the Real-Time Streaming Protocol (RTSP) [2] 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.

III. WORKAHEAD SMOOTHING

Prefix caching 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.

A. Smoothing Model

The emergence of high-speed backbone and access networks facilitates a wide range of multimedia applications, including streaming of high-quality video. 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.

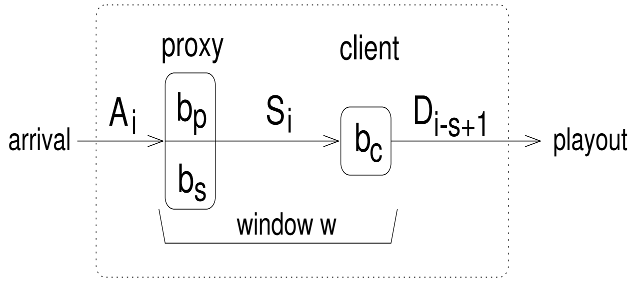


Fig. 2. Smoothing model

The burstiness could be reduced by adjusting the quantization level of frames during scenes with significant detail or motion, at the expense 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 [3, 4]. 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.

The opportunities for smoothing depend on the frame and buffer sizes, as well as the smoothing window and the playback delay, as shown in Fig. 2 and Table I. 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 $D_i = \sum_{j=1}^{i} f_j$. Upon receiving a client request 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+1}.

The incoming frames from the server beyond the prefix frames are *temporarily* stored in the proxy's b_s-bit *staging* buffer while they await transmission to the client. As discussed in Section II, we define d_{min} and d_{max} respectively as the minimum and maximum delays from the server to the proxy, resulting in a maximum network jitter of $\Delta = d_{max} - d_{min}$. To accommodate the client's s-frame playback delay, the prefix proxy 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 $d_{max} - s + w + 1$. 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,

TABLE I

PARAMETERS IN SMOOTHING MODEL

Parameter	Definition
s	Client playback delay
w	Proxy smoothing window
d_{min}	Minimum server delay
d_{max}	Maximum server delay
Δ	Jitter between server and proxy
b_p	Proxy prefix buffer size
b_s	Proxy staging buffer size
b_c	Client playback buffer size
$\mathbf{D}$	Cumulative frame size vector
$\mathbf{A}$	Arrival vector at proxy
$\mathbf{S}$	Transmission vector from proxy

$A_i \in [A_i^{\min}, A_i^{\max}]$, where

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

and

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

When $w = s$ 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 [5, 24].

B. 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 client must receive at least D_{i-s+1} 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 the proxy only can be sure of receiving 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 $S_i \le U_i$ for $i = 1, 2, \ldots, N+s$. The computation conservatively uses A_{i-1}^{min} instead of A_i^{min}, as $A_i^{min} - A_{i-1}^{min}$ reaches the proxy 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 staging buffer. At the beginning of the transmission, the proxy sends frames from the prefix buffer and does not free any space in the staging buffer. The proxy must complete transmission of the prefix frames before the staging buffer overflows. Note that, due to the variability in server delay, the maximum cumulative amount of data

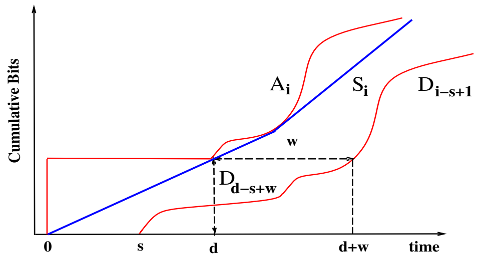


Fig. 3. Sample smoothing constraints

the proxy can receive by time i is A_i^{max}. The earliest time the arriving stream would exhaust the staging buffer is then

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

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

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

Combined with the underflow constraint at the client buffer, this results in the following *lower constraint* $L_i = \max\{L_i^a, L_i^b\}$ on any feasible transmission schedule, where $L_i \le S_i$ for $i = 1, 2, \ldots, N+s$. Fig. 3 shows an example of the upper and lower constraints. Subject to the constraint $L_i \le S_i \le U_i$, 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.

C. Transmission Schedule

Consider the constraint vectors $\mathbf{L} = (L_0, \ldots, L_{N+s})$ and $\mathbf{U} = (U_0, \ldots, U_{N+s})$. Generating a 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 [13]. In particular, in the context of smoothed transmission of stored video from the server to the client, an $O(N)$ smoothing algorithm [12] was proposed, that computes the shortest-path transmission schedule $\mathbf{S}^*$. $\mathbf{S}^*$ 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 IV. Figure 3 shows a sample transmission schedule.

For prerecorded video, the frame-size information is available in advance, and the schedule $\mathbf{S}^*$ could be precomputed 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

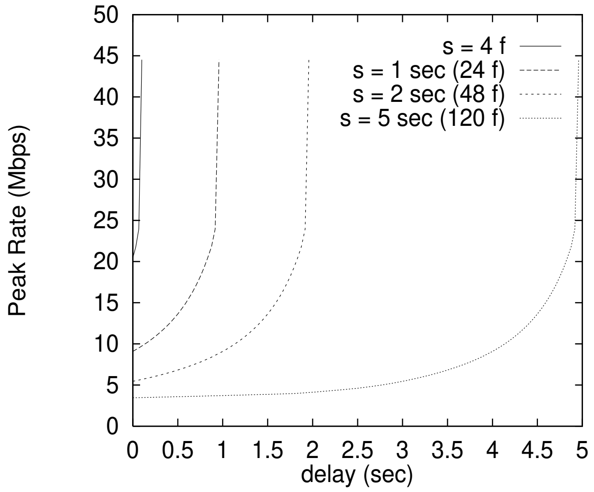


Fig. 4. Peak rate vs. d for *Blues Brothers* ($b_p = 0$, $b_c = 8$ MB)

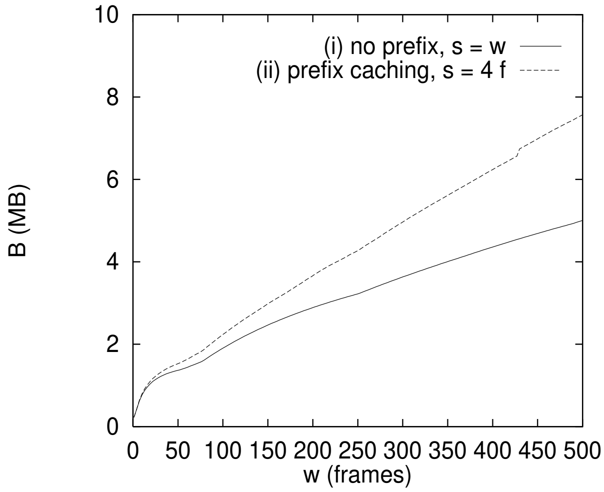


Fig. 5. Proxy buffer B vs. w for *Blues Brothers* ($d = 0$, $b_c = 8$ MB)

from the server. Each online computation applies the shortest-path algorithm to a sliding window of frames, following the approach in [24]. Online computation is also required when transmitting live video content with an initial prerecorded segment. In either case, 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.

IV. Performance Evaluation

Through experiments with MPEG traces, we demonstrate the performance benefits of having prefix and staging buffers at the proxy. We vary the key parameters in the model, and measure the impact on the peak rate of the smoothed schedule. A more detailed evaluation, including other variability metrics, is presented in [25].

A. Video Traces

The experiments draw on two constant-quality MPEG video traces. MPEG streams consist of a mixture of I, P, and B frames. The first trace is a 23-minute MPEG-1 clip from the movie *The Wizard of Oz*, 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 group-of-pictures structure ($IBBPBBPBBPBBPBB$). Note that a B frame is inter-frame coded, and 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 vectors for *Wizard of Oz* were constructed to capture these constraints [5]. The second trace is a 17-minute MPEG-2 clip from the movie *Blues Brothers*, 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.

B. 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 $w = s - d$. Fig. 4 plots the peak rate 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 decreases 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 schedule. For example, $d = 1$ and $s = 4$ result in a peak rate of 22 Mbits/second. The peak rate drops by more than 50% to 9.3 Mbits/second under a 24-frame (1-second) playback delay, and by almost another 50% 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 Fig. 4. However, in the absence of a prefix buffer, these gains are not achievable unless the client tolerates a relatively high playback latency. The 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 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 [5]. In the next subsection, we demonstrate that these gains are achievable in the proxy prefix model, even with a relatively small playback delay.

C. 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. We compare a system with a w-frame playback delay and no prefix buffer to a system with a 4-frame playback delay and a $(w-4)$-frame prefix buffer; both scenarios assume a server delay $d = 0$. Despite the differences between the two configurations, the variability metrics (not shown) are virtually indistinguishable. 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. 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. Therefore, 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 prefix caching to be quite close to what can be achieved with the longer startup delay $s = w + d$.

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

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

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 Fig. 5 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 staging 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.

To illustrate the trade-off between buffer space and smoothing gain, Fig. 6 plots the peak rate 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. 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

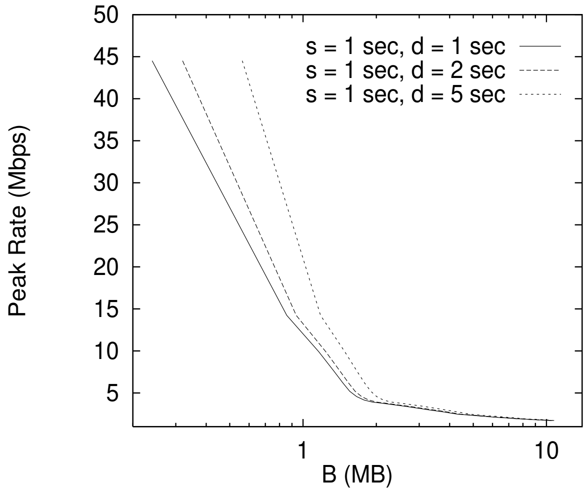


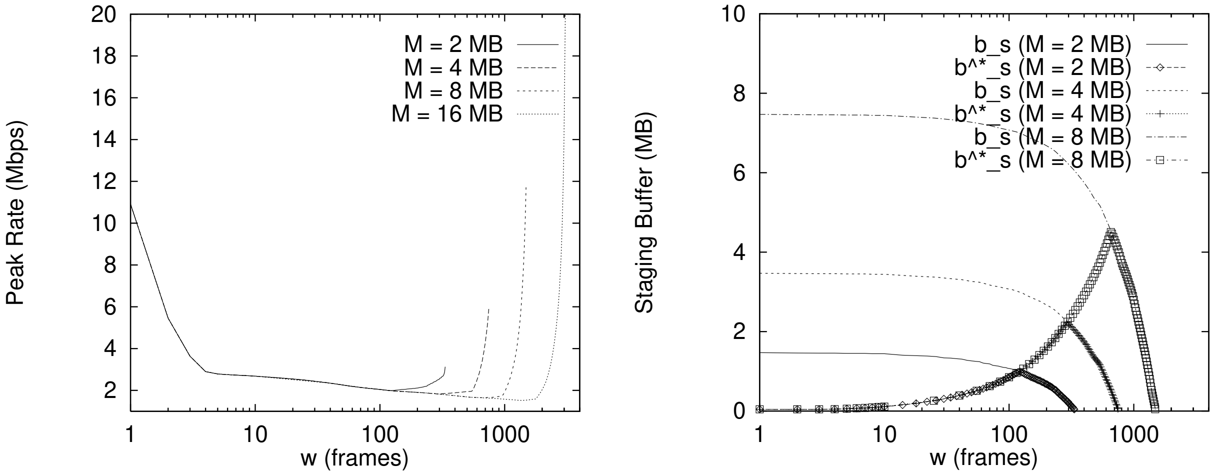
Fig. 6. Peak rate vs. $B = b_p + b_s$ for *Blues Brothers* ($b_c = 8$ MB)

proxy buffer reduces the peak rate to 4 Mbits/second from the unsmoothed values of 44.5 Mbits/second. In addition, the performance is very similar across different values of d once the proxy has a few Mbytes of buffer space.

D. *Proxy Prefix and Staging 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 staging 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 staging buffer b_s (having enough storage to absorb transient bursts of large frames). The graph in Fig. 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, Fig. 7(b) plots the resulting size of the staging 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 the increasing region of b_s^* and starts increasing when b_s^* starts decreasing.

These results 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 staging 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 small size of the staging buffer begins to impose a constraint on the transmission schedule. This forces the proxy to transmit aggressively to avoid overflowing its staging buffer. These workahead transmissions are detrimental to smoothing, and actually increase the peak bandwidth.



(a) Peak rate

(b) Staging buffer

Fig. 7. Peak rate and size of smoothing buffer (b_s and b_s^*) vs. w, for fixed buffer budget M for *Wizard of Oz* ($s = 1$ sec, $d = 5$ sec, $b_c = 32$ MB)

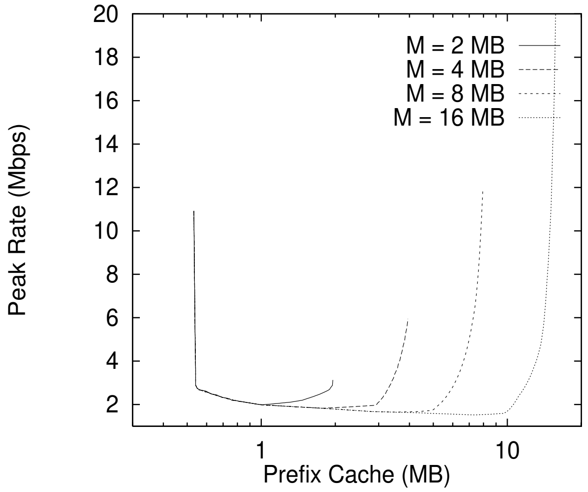


Fig. 8. Peak rate vs. b_p for fixed buffer budget M for *Wizard of Oz* ($s = 1$ sec, $d = 5$ sec, $b_c = 32$ MB)

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 staging buffer is necessary. Fig. 8 plots the peak rate as a function of the prefix buffer size, for several total buffer allocations M. As in Fig. 7(a), the metric initially decreases rapidly before flattening, and then increases 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 increase in b_p; in this region, the staging 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 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 staging buffers.

V. Multiple Clients and Multiple Streams

Drawing on results in Section IV, we describe how to allocate the proxy's buffer and bandwidth resources across multiple streams and multiple clients. Then, we discuss how to reduce the buffer requirements by sharing a common staging buffer among multiple client streams.

A. *Buffer and Bandwidth Allocation*

The proxy needs an effective way to allocate 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, $v = 1, 2, \ldots, V$. We assume that the clients do not necessarily arrive close together in time. Hence, clients of the same stream do not share a common staging 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 of the smoothed schedule, and allocates a fixed staging 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. Section IV-D shows that assigning a staging buffer smaller than b_s^* bits significantly degrades performance. 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 staging buffers and smoothing windows. This trend is apparent in Fig. 7(a), 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 the same allocation across different clients.

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, until the limit 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 used to allocate bandwidth subject to a limit M on proxy buffer space.

B. *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 staging buffer space across several independent streams. For an individual stream, the utilization of the staging buffer changes across time, as shown in Fig. 9(a). 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 the envelope of window of w frames ($\max_j\{\sum_{i=j}^{j+w-1} f_i\}$), as well as the maximum ($\max_i\{b_{s,i}\}$) and average ($\text{avg}\{b_{s,i}\}$) staging buffer requirements. 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.

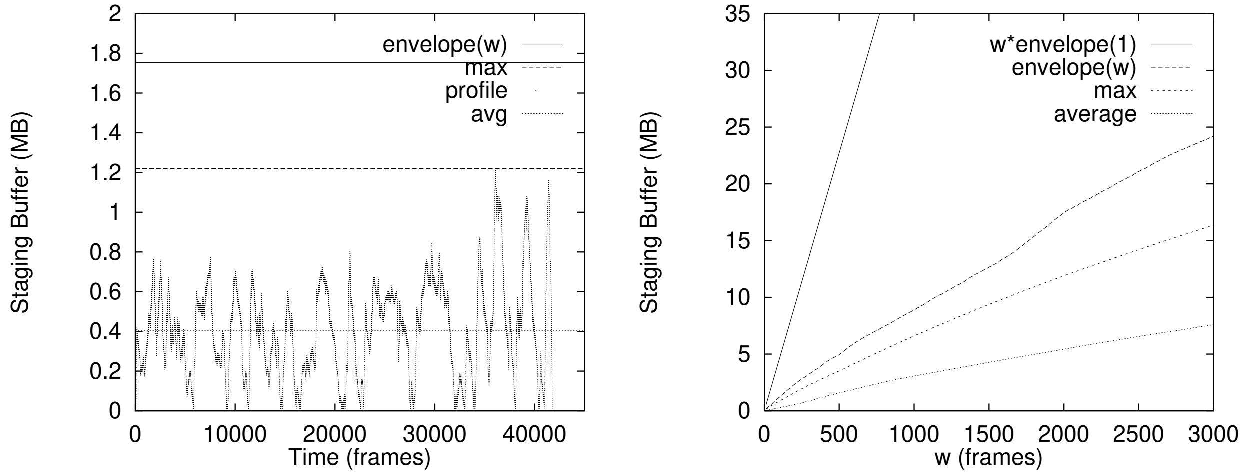
Exploiting frame-size information offers a significant reduction in buffer requirements over a simple characterization of the traffic envelope. In fact, even the maximum buffer usage (b_s^*) is a fairly conservative estimate of resource requirements, since the peak utilization occurs for a very short period of time. These trends persist across a range of window sizes, as shown in Fig. 9(b). The temporal fluctuations suggest that multiplexing the proxy buffer space could substantially reduce the resource requirements. These benefits are attainable even if clients do not issue requests at the same time or for the same stream. 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, 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 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.

VI. Conclusions

To address the poor delay, throughput, and loss properties of the Internet, we proposed that network service providers deploy proxies that cache the intial frames of popular audio/video clips. In addition to hiding the effects 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. As future work, we plan to explore the multiple-client bandwidth and buffer allocation problem in greater depth. 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 schedule dynamically at the proxy, based on the actual sequence of arrivals from the server, offers substantial gains over using a precomputed schedule.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback.



(a) Staging buffer utilization

(b) Staging buffer size

Fig. 9. Staging buffer usage across time ($w = 5$ seconds), and vs. w, for *Wizard of Oz* ($s = 0.5$ sec, $d = 2$ sec, $b_c = 32$ MB)

REFERENCES

[1] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1 (draft-ietf-http-v11-spec-rev-06.txt)," November 1998.

[2] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP), request for comments 2326," April 1998.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] M. Kamath, K. Ramamritham, and D. Towsley, "Continuous media sharing in multimedia database systems," in *Proc. International Conference on Database Systems for Advanced Applications*, April 1995.

[10] 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.

[11] 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.

[12] 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," *IEEE/ACM Trans. Networking*, vol. 6, pp. 397–410, August 1998.

[13] 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.

[14] 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.

[15] E. Amir, S. McCanne, and H. Zhang, "An application level video gateway," in *Proc. ACM Multimedia*, November 1995.

[16] 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.

[17] K. Ross and J. Kurose, *Computer Networking and Internet Protocols*. Online Document, 1998.
`http://www.seas.upenn.edu/~ross/book/Contents.htm`.

[18] N. F. Maxemchuk and S. Lo, "Measurement and interpretation of voice traffic on the Internet," in *Proc. International Conference on Communications*, June 1997.

[19] V. Paxson, "End-to-end Internet packet dynamics," in *Proc. ACM SIGCOMM*, pp. 139–152, September 1997.

[20] J. M. Boyce and R. D. Gaglianello, "Loss effects on MPEG video sent over the public Internet," in *Proc. ACM Multimedia*, September 1998.

[21] 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.

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

[23] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications, request for comments 1889," January 1996.

[24] J. Rexford and D. Towsley, "Smoothing variable-bit-rate video in an internetwork," *IEEE/ACM Trans. Networking*, 1999.

[25] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," Tech. Rep. 98-27, Department of Computer Science, University of Massachusetts Amherst, 1998.