Proxy caching for efficient video services over the Internet [1]

Zhourong Miao, Antonio Ortega
Integrated Media Systems Center, Signal and Image Processing Institute,
Department of Electrical Engineering-Systems,
University of Southern California,
Los Angeles, CA 90089, Email: {zmiao, ortega}@sipi.usc.edu

Abstract: Proxy caching has proven to be a key component of emerging Internet middle-ware. Currently, proxy caching is used to speed up web browsing and reduce networking costs, as popular web objects are likely to be present in the proxy cache. In this paper we study the extension of proxy caching techniques to video. A trivial extension consists of storing complete video sequences in the cache. We will show how other approaches, where only a few frames are cached due to the large scale of video data size and possibly limited cache space on proxy, can also contribute to significant improvements in performance. In particular we will discuss two video caching strategies, initial caching and selective caching, to store part of the video stream onto proxy. We will show that by selective caching, we can maximize the robustness of the video stream against network congestion, while not violating the limited decoder (user) buffer size.

1 Introduction

Interest in proxy based caching has increased with the growth in Internet traffic and the initial research in this area (e.g. within the Harvest project [1]) has quickly led to the development of commercial products (e.g., [2, 3]) and to continuing research activity (e.g., [4, 5]). Currently, proxy caching is used to speed up web browsing and to reduce networking costs, i.e, some of the most popular web objects are likely to be in the cache and thus will be fetched from the proxy, which is close to the client, instead of requiring access to the web server. Studies and everyday use have demonstrated that proxy caching can result in significant improvements in network access efficiency.

Video is becoming a significant proportion of Internet traffic and, given the high data volumes involved, even a few popular video applications can result in potential congestion problems. Thus,

[1] This work has been funded in part by the the Integrated Media Systems Center, a National Science Foundation Engineering Research Center and by the National Science Foundation under grant MIP-9804959.

when pre-encoded video sequences are likely to be accessed by many users, ISPs have resorted to approaches such as replicating complete video sequences in mirror sites "closer" to the end users. Obviously, this is already a form of caching, and it indicates the potential benefits arising from caching strategies specifically targeted for video. These will be the focus of this paper. Note that while for objects such as images the main advantage of caching is the reduction in latency, video caching has other additional benefits. For example, we will show that video caching allows us to increase the robustness in the delivery of streamed video, by reducing the effect of variations in channel bandwidth. Likewise, caching may prove to be useful to facilitate efficient implementation of additional functionalities, such as random access (rewind, fast forward, etc.).

The great majority of recent proxy caching research and development has focused on techniques that can handle generic web objects, i.e., a decision is made about whether an object should be cached based on the type of object, or on meta-data provided by the content creator, but among "cacheable" objects no distinction is made between, say, an HTML text file and a JPEG image. Some recent work has proposed that having caching strategies that are specific of particular types of objects can help improve the overall performance. For example, the idea of "soft" caching as applied to images [6, 7, 8, 9] results in images being recoded, i.e., compressed with lower quality, instead of simply being removed from the cache when there is not sufficient space in the cache. Other object types, such as for example images stored in Flashpix format, may not be cacheable unless the proxy supports the specific access syntax for these objects (e.g., the proxy is able to understand instructions specifying the image tiles to be provided to the users.)

In this paper, we consider the role of proxies for *video stream caching* to improve the performance of playback of pre-encoded video over the Internet. We will discuss the potential advantages of caching frequently accessed video streams, or parts of them. We will assume that the client access to the proxy is faster, or at least more reliable (in the sense of being subject to fewer rate variations), than access to the server[2]. We will assume that the proxy can support the same streaming protocol used by client and server, so that it can provide the requested frames, if they are stored in the proxy cache, or alternatively, pass the request to the server.

Clearly, the most immediate approach to video caching would be to let only complete video sequences be cached. However, since the size of video data is significant it is also important to

[2]Obviously video sequences that do not meet these characteristics, e.g., those stored in a fast and reliable server, need not be cached.

consider approaches that will store only parts of the video sequences. We will call this approach *selective caching*. This can be seen as a counterpart of the soft caching approach discussed earlier, with image recoding replaced by temporal video scalability.

The paper is organized as follows. We start by discussing in Section 2 how latency combined with appropriate rate control can help increase the video quality and reduce the effect on the decoded video of network unreliability. Then, in Section 3, after introducing the concept of decoder buffer contents trace (3.1), we study the role of initial segment caching (3.2) and selective caching (3.3) in reducing latency and improving reliability. We will also show how caching can provide an improved implementation of functionalities such as rewind, fast forward, etc.. In Section 4, we show the simulation and implementation results to evaluate the approaches proposed in Section 3. Section 5 concludes the paper.

Note that because the video sequences in this paper are pre-encoded, there are no constraints on buffering at the source. Therefore, in the rest of the paper the term *buffer* in always refers to the decoder buffer.

1.1 Related work

Both [10] and [11] address caching of video sequences. In [10] complete sequence are stored, with their "tails" being removed from the proxy as the cache fills up. Instead, [11] propose caching of the initial few frames (prefix) of a video sequence, as a way of reduces the overall latency. A large number of frames can potentially be prefetched without substantial latency (since fetching from a proxy is fast) and that will ensure a smooth playback. In our approach we consider the more general case where any frames can be cached. The approaches of [10, 11] can then be seen as particular cases, i.e., our algorithm can generate those solutions given a appropriate cost function.

The work in [12] also allows arbitrary parts of a video sequence to be stored, but it assumes that a *layered* coder has been used. Instead, we consider *non-layered* coder and our caching discussions are make on a frame by frame basis.

2 Initial latency and initial buffering

Most popular video compression algorithms and standards (e.g. MPEG-1, MPEG-2 [13]) produce variable bit rate (VBR) compressed bit streams. When transmitting the pre-encoded data over

a CBR channel (or even a VBR channel), a decoder buffer is usually required to smooth out the variations in rate of the video data (i.e, the decoder receives a variable number of frames per second, but decodes frames at a constant rate). Some studies [14, 15, 16] show different approaches to cope with the decoder buffer underflow and/or overflow (note, that decoder buffer underflow is the most significant problem since it will produce jitter in the display of video frames) caused by the VBR video data, and possibly, by network congestion. Most strategies among those, as well as some commercial applications[3] show that an initial latency will lead to a smoother playback, by increasing the robustness against potential decoder buffer underflow (jitter).

Let us consider first the trade-off between quality of the compressed video and its output data rate. We discretize the time t into units of duration of one frame display period (e.g, 1/24 second), assuming that the user starts to playback the stream at time $t = 0$. Each frame is labeled as F_i, indicating that frame F_i would be scheduled to be displayed at time $t = i$. Assume the video length (total number of frames) is T.

Frame F_i is compressed into R_{i,q_k} bits with quantization step size q_k. Different quantization step sizes produce different compression ratios and quality. A large step size can give more compression gain (less output bits for a frame) and poor quality (high distortion), and vice versa. In this paper we assume that a generic quantization parameter (step size) x_i is used for each frame. We measure the distortion D_{i,q_k} for frame F_i, with quantization parameter q_k, and denote D as the total distortion of the video stream.

$$D = \sum_{i=0}^{T} D_{i,q_k}. \tag{1}$$

Let $\vec{x} = [x_1, x_2, ..., x_T]$, be a T-dimensional vector where x_i is the quantization step size for frame F_i. If we have K different quantization step sizes available for each frame, then each element x_i in vector $\vec{x}$ can take K different values, $x_i \in \{q_1, q_2, ..., q_k\}$. Thus in total there can be T^K different vectors $\vec{x}$. Define Q as the set including all the possible vectors $\vec{x}$.

For a pre-encoded video stream, the quality of each compressed frame can be adjusted to produce a desired rate with high quality requiring more bits, and vice versa. Then for a given deterministic channel rate model, decoder buffer size, and initial latency (measured in time or by the number of frames to be pre-fetched before playback start), we can use rate control algorithms to produce jitter-free compressed video data that meets the corresponding set of constraints. The

[3]For example, the RealAudio and RealPlayer [17].

problem can be formalized as follows.

Our goal will be to find the optimal solution $\vec{x^*} \in Q$ that minimizes D, under a constraint set $C = \{R_{ch}, \tau_{ini}, B_{high}\}$, where D is the total distortion of the video stream, R_{ch} is the channel rate, τ_{ini} is the initial latency, and B_{high} is the decoder buffer size. The initial latency τ_{ini} is the time between the user starting to receive data and the time playback of the first frame starts. This latency allows the user to pre-fetch some frames into buffer, aiding the smoothing of VBR video data. τ_{ini} can also be replaced as S_{req}, where $S_{req} = \tau_{ini} R_{ch}$. We define S_{req} as the *required initial segment* (in bits), and assume the user will prefetch it before playback. Note that because we are considering pre-encoded video, we assume a certain set of constraints has been chosen based on typical decoder configurations. Thus since video has been encoded assuming a latency τ_{ini}, the user will have to prefetch S_{req} bits before playback starts.

We would like to find $\vec{x^*}$, such that the output video stream has the best quality (lowest distortion D), and, under constraint set C, can be played without any buffer underflow (jitter) or overflow (exceeding B_{high}).

More formally, since the channel rate is assumed to be deterministic when compressing the video data, it can be expressed as $R_{ch}(i)$ at time $t = i$. Thus the explicit constraint equations are:

$$\sum_{i=0}^{t} R_{ch}(i) + S_{req} - \sum_{i=0}^{t} R_{i,x_k} \geq 0 \qquad \forall t \in \{1, 2, ..., T\} \tag{2}$$

$$\sum_{i=0}^{t} R_{ch}(i) + S_{req} - \sum_{i=0}^{t} R_{i,x_k} \leq B_{high} \qquad \forall t \in \{1, 2, ..., T\} \tag{3}$$

The left hand side of (2) is the total amount of data transmitted to decoder buffer (including initial segment) by time t, minus the total amount of data consumed by time t (i.e., the data corresponding to the first t frames), the result should be greater than zero to prevent buffer underflow. Similarly, the amount of data in the buffer should be smaller than the buffer upper bound B_{high} to prevent buffer overflow.

A simple case of deterministic channel rate is when $R_{ch}(i)$ is constant (i.e.,a CBR channel) which is a reasonable assumption for off-line compression, because we can not know a prior what the channel conditions will be when transmitting. However, the actual channel rate during transmission is not assumed to be deterministic or constant, due to network congestion, delay, etc.. This leads to the potential for frame losses, if frames arrive "too late" at the decoder. If we force the decode buffer to hold always at least a few frames, then it may be possible to

ensure playback during short network congestion periods. This can be achieved by introducing a further constraint in the rate control at the compression stage. For example, we can set a buffer lower bound, B_{low}, as an additional constraint, so that the new constraint set is $C = \{R_{ch}, \tau_{ini}, B_{high}, B_{low}\}$, and we rewrite (2) as,

$$\sum_{i=0}^{t} R_{ch}(i) + S_{req} - \sum_{i=0}^{t} R_{i,x_k} \geq B_{low} \qquad \forall t \in \{1, 2, ..., T\} \tag{4}$$

Thus after compression with rate control, the decoder buffer will never drop below B_{low} under a pre-defined channel rate R_{ch}. If network congestion occurs, the decoder will have at least B_{low} data for playback while waiting for the delayed frame. The larger B_{low}, the more time the decoder will have to survive the congestion without losing a frame. For example, if k frames are still in the buffer when congestion occurs, the decoder will have k more frame intervals (e.g., $k/24$ seconds) before running out of data to playback. Therefore a video stream generated this way is more *robust* against network congestion as B_{low} grows.[4] This is illustrated by Fig. 1. We will discuss and define robustness in more detail in Section 3.1.

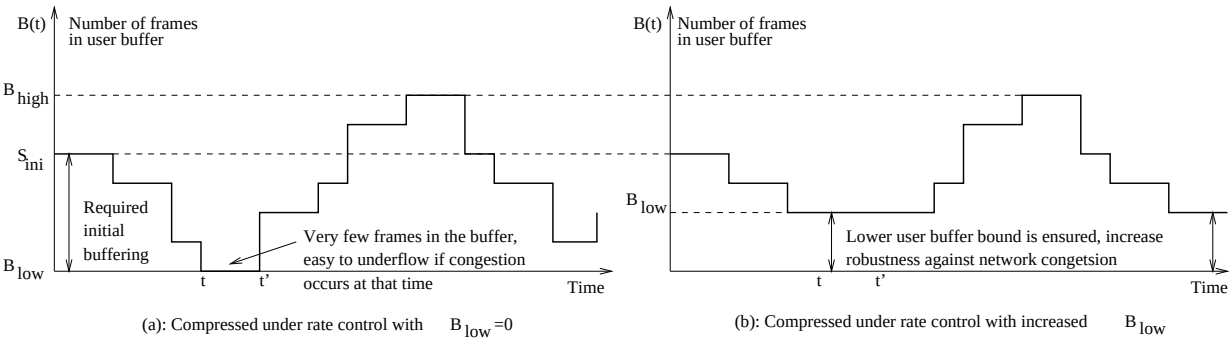

Figure 1: Trace of number of frames in buffer during playback. (a): Trace is sketched with $B_{low} = 0$, which means buffer will be drained to zero at sometime t. (b): Trace with increased B_{low}. Usually the PSNR in left figure is better than that of in right figure, because of the more variation allowed in compression stage. But the trace in (b) is more robust against network congestion.

More details of rate control algorithms and method to find optimal solution x can be found in [18, 19, 20]. We only point out the impact of the parameters in C to the quality of compressed

[4] Actually, the actual size of buffer should be expressed in number of bits, which may not be proportional to the number of frames, the detail explanation for this can be found in Fig. 4 in Section 3.1.

video (corresponding to the optimal quantization step size $\vec{x}$).

From (3), (2) we can see,

1. A large required initial segment size S_{req}, (or τ_{ini} in the constraint set C) in (2, 3) allows more *bits* for the frames ($\sum R_{i,x_k}$), thus it can improve the overall quality (PSNR) of compressed video.

2. A large buffer size B_{high} allows more *variation* in VBR compressed video data, as in (3). For example, a long series of "simple" frames (resulting in fewer bits after compression) might be accumulated in user buffer, since the playback rate of frames is much less the arrival rate of frames in this case, so that the user buffer will be fill up during this period. Thus, a large B_{high} will allow more significant changes in video data rate, and improving the video quality.

3. If we set B_{low} in (4) to be larger, the variation of the video data rate is limited. This would be equivalent to having a smaller user buffer (B_{high}) in (2, 3). Thus a larger B_{low} results in worse quality. The difference is that we increase the robustness of the video stream against network congestion. So there is trade-off between robustness and output quality (as well as between initial latency and quality).

In Fig. 2 we show experimental results of rate control with different parameters in constraint set C for compression to illustrate the trade-offs described above. In this experiment we use *Multiple Lagrange Multipliers* algorithm for optimal rate control in the compressing the video stream. Refer to [18, 20, 21] for more details about this algorithm.

We set the channel rate R_{ch} in C to be constant for compression. The number of pre-fetched frames (required initial segment, S_req) ranges from 0 to 100. A large number of pre-fetched frames indicates large required initial buffering size S_{req}, or large initial latency τ_{ini}. We show results for two different buffer lower bounds ($B_{low} = 0$ and $B_{low} = 100Kbits$). We can see the quality goes up as the initial buffering (latency) increases, and a large low buffer bound results in lower quality (with higher robustness against network congestion).

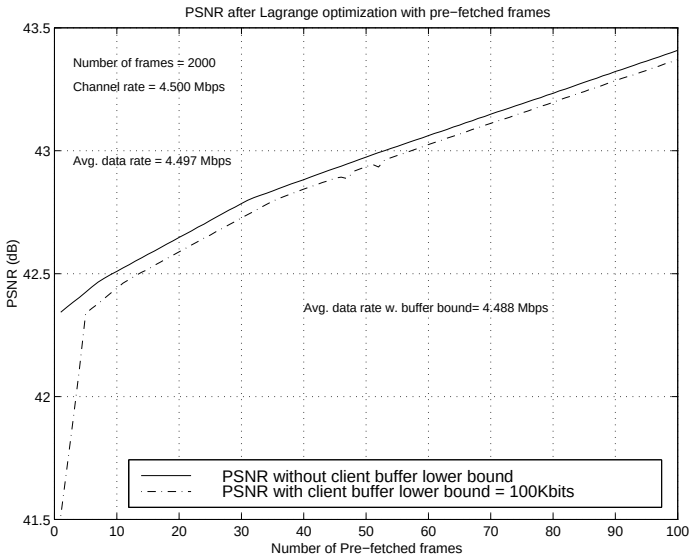

Figure 2: Quality vs Initial Latency. Top curve: as the latency increases so does the quality. Bottom curve: we restrict the range of variation in rates so as to ensure a large lower buffer bound. This results in lower PSNR performance but increased robustness against network congestion. The horizontal axis is the number of prefetched frames, and vertical axis is the overall quality (in PSNR) of video stream. The video source is part of the movie "Mission Impossible", compressed with MPEG-2 intra frame mode. We use the same sequence in our other experimental results.

3 Proxy caching for video sequences

3.1 Trace of decoder buffer contents

Our goal is now to analyze what frames should be cached. We first introduce the concept of the trace of decoder buffer contents during playback, and discuss its properties with/without caching.

Denote $B^f(t)$ and $B^b(t)$ as the number of frames and bits, respectively, in the decoder buffer at playback time t. We call $B^f(t)$ and $B^b(t)$ the *trace of buffer contents*. These can be easily obtained by simulating the video transmission under specific channel model. We assume that compared to the channel rate R_c from server to client, the rate R_p from proxy to client is very fast, that is $R_p >> R_c$. Thus the transmission time from proxy to client is very small and is neglected in this paper for simplicity.

We now examine the buffer contents traces before and after caching, see Fig. 3. Denote

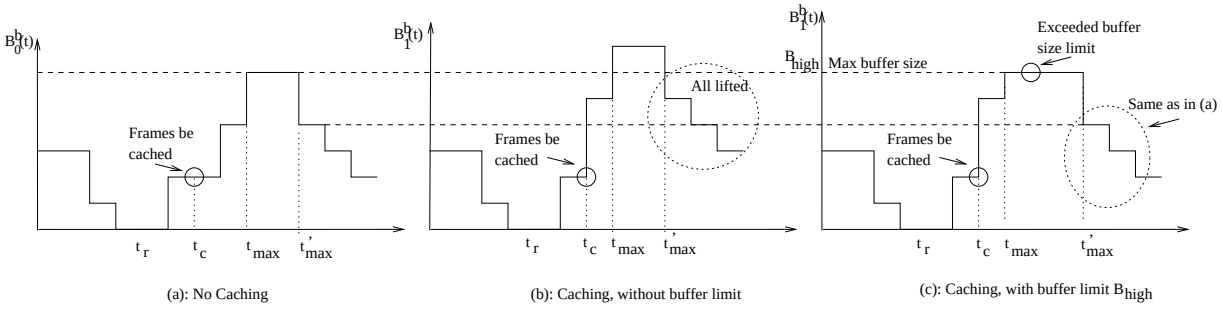

Figure 3: Buffer traces with/without caching

$B_0^f(t), B_0^b(t)$ as the traces without any caching, and $B_1^f(t), B_1^b(t)$ as the traces after caching a frame. In what follows, we use the sub-index i to indicate the number of frames that has been cached. Assume the frame be transmitted at time t_c is cached at the proxy, it will be transmitted to client from proxy instantly at time t_c, and the proxy will request the server to skip this frame and transmit the next one, so the trace $B_0^b(t)$ after t_c will be "lifted" by R_{t_c}, the size of frame F_{t_c}, or

$$B_1^b(t) = \begin{cases} B_0^b(t) & \text{if } t < t_c \\ B_0^b(t + \delta t) + R_{t_c} & \text{if } t \geq t_c \end{cases} \tag{5}$$

Where $\delta t = \lceil R_{t_c}/R_{ch} \rceil$ is the transmission time of frame F_{t_c} from server to client. Equation (5) holds when trace $B_1^b(t)$ does not exceed the maximum size of the decoder buffer B_{high} after t_c, or

$$B_1^b(t) = B_0^b(t + \delta t) + R_{t_c} < B_{high} \qquad \text{for } t > t_c \tag{6}$$

Note that $B_0^b(t)$ will never exceed B_{high} since it has been generated with a rate control that enforces the constraint of no buffer overflow.

If, after "lifting", equation (6) does not hold for some time $t_{max} > t_c$, i.e., $B_1^b(t)$ would exceed B_{high}, then the user has to stop receiving data, and equation (5) becomes:

$$B_1^b(t) = \begin{cases} B_0^b(t) & \text{if } t < t_c \\ B_0^b(t + \delta t) + R_{t_c} & \text{if } t_c \leq t < t_{max} \\ B_{high} & \text{if } t_{max} \leq t < t'_{max} \\ B_0^b(t) & \text{if } t > t'_{max} \end{cases} \tag{7}$$

Where (t_{max}, t'_{max}) is the interval during which the buffer is full. See Fig. 3(c).

The traces are obtained from the simulation under a given deterministic channel rate, e.g., a CBR channel. However, when transmitting over a real network, congestion may occur in a non-deterministic fashion, resulting in delay of the video packets. If a packet arrived later than the time it is scheduled to be displayed (i.e. the frame "time out"), it will be considered as a lost packet. When the decoder buffer has many frames already received, the next frame (packet) being transmitted can have a longer delay before time out, since the decoder can playback the frames already in the buffer while waiting for next frame. Thus at a time when there are many frames in the decoder buffer, the system will be more *robust* against network congestion. Conversely, at times with very few frames are in the buffer it is very likely for decoder to survive from buffer underflow. Thus we define a measure of robustness of a video stream U_m as

$$U_m = \min_t \{B^f(t)\}, \tag{8}$$

that is, the minimum value of the trace in terms of number of frames, and we call this minimum value a *trough*, with the corresponding time being referred to as a "risky time" t_r, i.e.,

$$t_r = \arg\min_t \{B^f(t)\}, \tag{9}$$

Similarly, "risky frames" are the frames scheduled to be displayed around risky time. There might be many risky times/frames in one trace. The larger $B^f(t)$ is, the more robust this video stream gets.

Obviously there are alternative ways to define robustness such as for instance the average number of frames in the buffer:

$$U_a = \sum_t B^f(t). \tag{10}$$

In this case, caching any frames can increase U_a. Given these two measures of robustness, U_m and U_a, when selecting which frames to be cached we can do so based on these two alternative optimization criteria. The *MaxMin* criterion will attempt to *maximize* U_m (and ensure the worst case is the least "risky"), while the *MaxAverage* criterion will require us to *maximize* U_a (and aim at improving overall or average robustness). For most scenarios in this paper, we use the MaxMin criterion for robustness, and we will use the MaxAverage criterion only to break the tie among multiple choices that all improve U_m in the same way.

We also define the *maximum peak* as the points at which trace $B^b(t)$ reaches the maximum buffer size limit, and *maximum peak time*, t_{max}, as the time when $B^b(t)$ reaches the maximum peak, that is

$$B^b(t_{max}) = B_{high} \tag{11}$$

As seen in Fig. 4, while the size in bits of each frame is different (due the VBR video data), the shapes of $B^f(t)$ and $B^b(t)$ tend to be similar (due the average over time). Therefore for simplicity, we could also use the $B^b(t)$ to locate the trough time t_r.

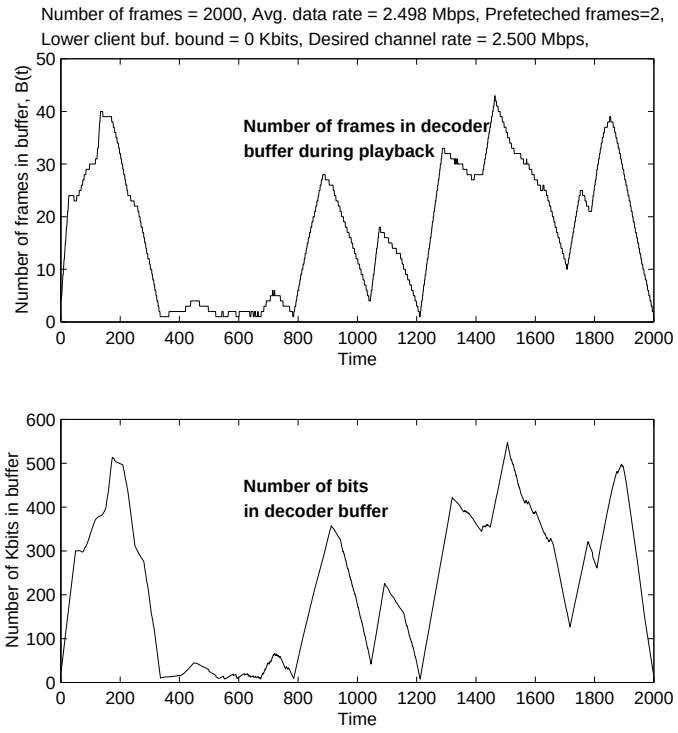

Figure 4: Trace of buffer size in number of frames and bits. We can see that two traces have similar shapes. Thus, even though the number of bits per frame is not constant, it is reasonable to make the approximation that the extrema of both curves will happen at roughly the same times.

3.2 Approach I: caching initial segment

For a proxy with limited cache size, one approach is to cache the *required* initial segment first (S_{req}, which has to be buffered before playback). In the case when there exists a fast link from proxy to the client, the physical delay required is much lower than if the required initial segment (S_{req}) has to be loaded from the server. This idea is also proposed in [11].

The next question is to determine, if more space is left on proxy after caching S_{req}, what portion of the video segment should be cached next? One simple solution is keep on caching the frames immediately after S_{req} until there is no space on proxy. We now analyze the benefits and

constraints of this approach.

In fact, if the user is willing to endure a long initial delay, it is possible to pre-fetch an *additional* segment (S_{add}) after fetching S_{req}. S_{add} (frames immediately follows S_{req}) can further increase the playback robustness against network congestion, as shown in Fig. 5 and Fig. 6(a). This long initial latency can ensure more robustness (without losing quality as achieved in compression stage with rate control).

If a proxy is caching both S_{req} and S_{add}, the latency can be significantly reduced. Thus the proxy can continue caching the frames after the S_{req} of the video stream, as the space on proxy permits. The more additional segments is cached, the more robust the playback will be for the user.

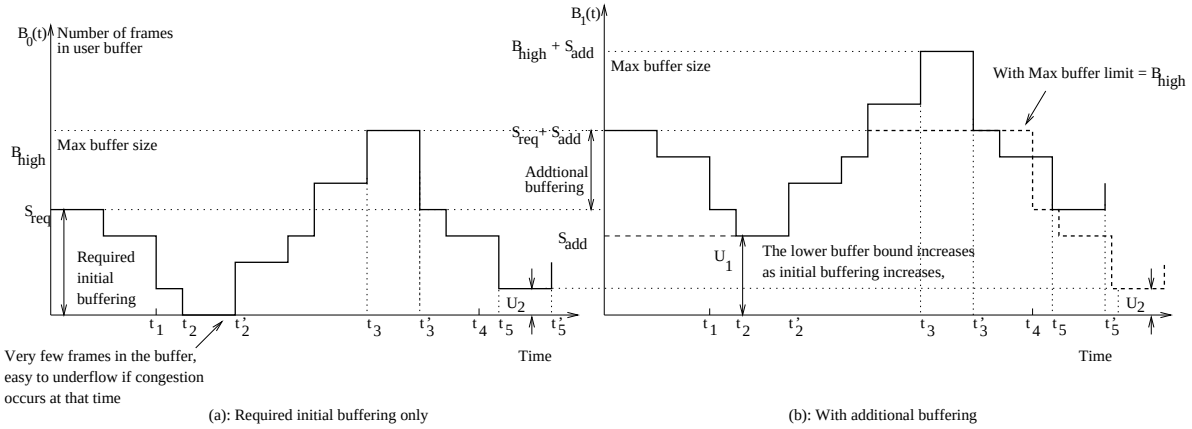

Figure 5: (a): Trace where only the required initial segment is cached, the robustness is $U_m = 0$ at this time (because of the trough at t_2). (b): After caching additional segment S_{add}, if there is no buffer limit, robustness can be increased to U_1. But if there is a maximum buffer size limit, robustness will drop to U_2 at tail of the stream.

However, arbitrarily increasing the length of the initial segment that is cached has the following problems, i) prefetching all cached frames will require some time, even if the link between proxy and user is fast, ii) The size of decoder buffer will have to be accordingly large. As in Fig. 5, the maximum user buffer occupancy is also increased when prefetching additional initial buffering (this is because the additional amount is not counted in the constraint set C with rate control). From equation (5) we know that the maximum decoder buffer size will reach to $B_{high} + S_{add}$, where S_{add} is the size of additional segment, (here $t_c = 0$).

If the decoder buffer does not meet to $B_{high} + S_{add}$, the trace $B(t)$ will not be "lifted" up

after the peak of buffer occupancy, as showed in equation (7). Thus a client with small buffer will not get any benefit when playing back the "tail" of video stream after t_4, since that will be fetched directly from server. The trace $B(t)$ is not lifted after the peak time, thus the robustness for the "tail" of video stream still remains the same as when no caching is involved. See Fig. 5 for more details.

3.3 Approach II: caching intermediate frames (selective caching)

We introduce another caching approach for streaming video, namely, *selective caching*, or selective buffering. In this approach, the proxy does not cache the frames immediately following the initial segment. Instead, it selects "intermediate frames" to be cached, as shown in Fig. 6.

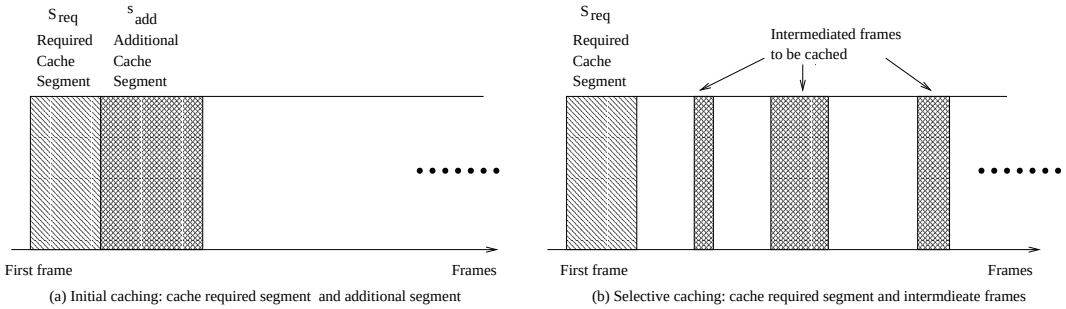

Figure 6: Caching initial segment and selective caching

Selective caching has two potential benefits over caching only the initial segment, namely, i) it can overcome the problem of decoder buffer limitation encountered with initial caching (described in previous section), ii) it can support added functionalities, such as fast forward, backward, which will be described in Section 3.5.

From equations (7) and Fig. 3(c), if a frame or frames are cached at time t_c, the traces $B^b(t)$ and $B^f(t)$ will be lifted in the time interval (t_c, t_{max}). So if $t_r \in (t_c, t_{max})$, we can easily increase the robustness U_m (with MaxMin criterion). If there are multiple troughs in $B^f(t)$, and they are interleaved by maximum peaks, caching frames at only one place will not help to increase U_m. For example, in Fig. 3(c), caching frames at t_c (or earlier) does not increase $B^b(t)$ (and $B^f(t)$) after t'_{max}. This is also why initial caching strategy (Approach I) does not benefit the "tail" of video stream after the maximum peak time.

In this case we can use a *selective caching* approach to increase the robustness U_m under the constraint of limited decoder buffer size. For example, in Fig.7(a), the trough period was at

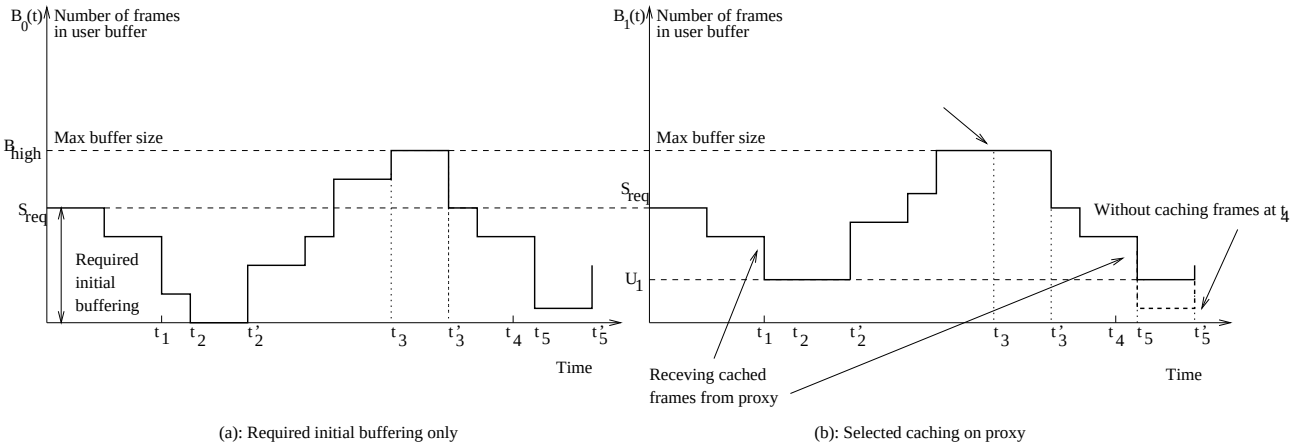

Figure 7: (a): Trace where only the required initial segment is cached and a trough occurs at time (t_2, t_2'), with the next trough occurring at (t_5, t_5'). (b): Selective caching. First select frames around t_1, but (t_5, t_5') still remains the same, due to the maximum peak at (t_3, t_3'), drawn in dotted line. Next select frames around t_4 to increase robustness for (t_5, t_5').

(t_2, t_2'), in order to increase $B^f(t)$ for robustness in this period, we can choose to cache frames around time t_1. But as $B^b(t)$ grows it may tend to exceed buffer limit B_{high} at maximum peak time (t_3, t_3'), resulting in no "lifting" around time (t_5, t_5'), which becomes the new trough after caching on t_1. Therefor it will be necessary to cache frames around t_4 in the next iteration.

Note that for the MaxMin criteria, many choices can be made to select frames to be cached. For example, to "lift" the first trough at t_2, we actually can select any frame to be cached between zero and t_2, in which case we can select (randomly) the frames around t_1. Similarly, for the trough t_5, the frames can be chosen between (t_3', t_5) to increase the robustness at that trough (where we decided for t_4). In Fig. 7(b), t_1 and t_4 are randomly selected. We can develop further constraints for selecting frames be cached, according to different benefit we want. In next section, we formalize our algorithm to choose frames, and using the MaxAverage criteria as a new constraint.

3.4 Algorithms to choose frames for selective caching

We assume there is limited space (S_p bits) on the proxy which can only hold part of video streams. Based on the analysis in section 3.3, we outline the approach of selective caching to increase the robustness under the constraint of decoder buffer size limit.

For a given pre-encoded video stream, compressed with rate control under constraint set

$C = \{R_{ch}, S_{req}, B_{high}, B_{low}\}$, consisting of frames $\{F_1, F_2, ..., F_T\}$. Define the frame index set $\mathbf{A} = [a_1, a_2, ..., a_N]$ such that frame F_{a_i} ($a_i \in \mathbf{A}$) is cached onto proxy, where $\mathbf{A}$ is a vector. Denote $B^f_{\mathbf{A}}(t)$ as the trace of buffer contents where all frames F_{a_i} (with size of R_{a_i}) are cached. We now formalize the problem as follows.

Find $\mathbf{A}^*$ (with N dimension) such that robustness $U_m = \min_t\{B^f_{\mathbf{A}}(t)\}$ is maximized (MaxMin criterion), or

$$\mathbf{A}^* = \arg\max_{\mathbf{A}} \Big\{ \min_t \{B^f_{\mathbf{A}}(t)\} \Big\}, \tag{12}$$

satisfying

$$\max_t \Big\{ B^f_{\mathbf{A}^*}(t) \Big\} < B_{high}, \qquad B_{high} \in C, \tag{13}$$

$$\sum_{i=0}^{N} R_{a_i} \le S_p, \qquad a_i \in \mathbf{A}. \tag{14}$$

For multiple choices of $\mathbf{A}^*$, break the tie by maximizing $U_a = \sum_t B^f_{\mathbf{A}}(t)$, or equivalently, using an additional criterion (MaxAverage)

$$\mathbf{A}^* = \arg\max_{\mathbf{A}} \Big\{ \sum_{t=0}^{T} B^f_{\mathbf{A}}(t) \Big\}. \tag{15}$$

A near optimal algorithm for the solution of this problem is summarized in the following steps, which will find all the frames to be cached, with their index forming $\mathbf{A}^*$, the solution for (12).

Step 1. Cache the required initial segment S_{req}, since the user has to prefetch it before playback to meet the constraints in rate control. Set $S_p \Leftarrow S_p - S_{req}$. Then start the first iteration, $i \Leftarrow 0$. Here for simplification, we use $B^f_i(t)$ to denote the trace after i^{th} iteration.

Step 2. If more space left ($S_p > 0$, or (14)), examine the current trace of decoder buffer contents, $B^f_i(t)$, find the most risky time t_{min} that satisfies $B^f_i(t_{min}) = \min\{B^f_i(t)\}$, see equation (9). At that time t_{min}, the decoder buffer will be most empty, e.g., time t_2 in Fig. 7(a). If there are multiple t_{min}, choose the first one. In this step, we use the MaxMin criteria.

Step 3. Find the *nearest* maximum peak time t_{max} before t_{min}, which is obtained by Step 2 (e.g., t'_3 for risky time t_4 in Fig. 7(b)). If no maximum peak exists before t_{min}, set t_{max} to zero (start time) (e.g., time 0 for t_2 in Fig. 7(a)). Note that t_{max} is obtained from trace $B^b_i(t)$.

Step 4. Select *one* frame F_{a_i} which is right after t_{max} obtained in Step 3. Since there are multiple choices for selecting, e.g. frames between t_{max} and t_{min} are all available for caching to increase $B_i^f(t_{min})$, we use MaxAverage criteria for robustness, and select the frame that is furthest away from the trough (but after the nearest previous peak). This provides the largest increase in average robustness U_a.

Step 5. Set F_{a_i} to be cached onto proxy. Update the trace $B_i^f(t)$ after F_{a_i} is cached. Note to obtain $B_i^f(t)$, we have to simulate $B_i^b(t)$ first, which is the buffer contents in terms of bits, since there might be buffer overflow of B_{high}, which is measured in number of bits. Set $S_p \Leftarrow S_p - R_{a_i}$, where R_{a_i} is the size of frame F_{a_i}.

Now we find a new element a_i for the solution, $\mathbf{A}^*$, of (12). If $S_p < 0$, there is not enough space left on proxy, stop the process. Otherwise, increase iteration index by 1, $i \Leftarrow i + 1$, go to Step 2.

The implementation and experimental results are shown in Section 4.2. Here we must point out that, with this algorithm, Approach I (caching additional initial segment) actually becomes a special case of Approach II (selective caching). This is explained as follows. Consider a buffer trace $B(t)$ with its unique maximum peak t_{max} occurs at the later part of the video stream, and the tail of $B(t)$ after t_{max} is very high compared to the trace before t_{max}, this algorithm will keep on selecting frames at the beginning of the video stream (of course, after the required initial segment, which is already cached). This is equivalent to Approach I, which simply continues caching those frames after required initial segment.

3.5 Added functionalities by selective caching

Another benefit of caching intermediate frames is to allow added functionalities. For example, in an MPEG encoded stream, one can place as many of the of the **I** frames as possible in the cache. Then, since the **I** frames are encoded in intra-frame mode it will be possible to perform rewind or fast forward without requiring access to other frames for decoding, and starting playback at an arbitrary frame position will be more efficient.

This can also be achieved by setting up additional criteria instead of the robustness metric we have used so far, especially for the MaxAverage criterion. This criterion will tend to select frames that are evenly spread over the entire stream.

4 Experimental results

4.1 Frame delay simulation with channel congestion

In order to verify that the risky time (frames) detected in step 3 of section 3.4 under CBR channel is also risky for channel with loss probability (packet lost or delay by congestion), we performed the following simulation, see Fig. 8. First we run the simulation with CBR channel (no loss), to get $B^f(t)$. It is used as the horizontal axis in Fig. 8, and each frame can be "tagged" with a number specifying the current buffer contents when transmitting this frame.

Then we run the simulation with some channel loss probability P_{loss}. The loss probability of each frame is calculated after performing 1000 runs. Note that we assume that when buffer underflow occurs, due to channel losses, the decoder delays playback of the following frame, so that losses do *not* propagate.

We can see that frame loss/delay probability in a probabilistic channel is higher when the buffer contents $B^f(t)$ are lower, where $B^f(t)$ is calculated based on a deterministic (CBR) channel model. The result supports our assumption in step 3 of section 3.4.

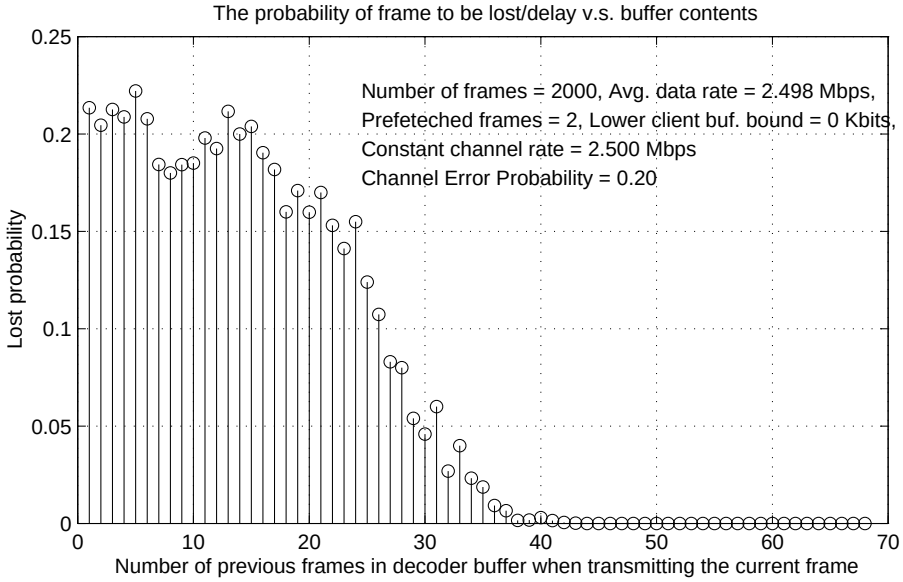

Figure 8: Lost/Delay probability v.s. buffer contents. The video sequence has 2000 frames. The real channel rate at any time is under modeled as a Binomial distribution, which drops to zero with P_{loss} (channel error probability), and keeps constant with probability $1 - P_{loss}$.

4.2 Frame selection

We implement our algorithm for selective caching described in Section 3.4. The results is shown in Fig. 9. Both traces are calculated during each iteration. $B^f(t)$ is used to select t_{min} in Step 2, while $B^b(t)$ is used to find t_{max} in Step 3. The dotted curves are for the first iteration, without any caching (except the required initial segment). The solid curves are for the last iteration, when the proxy gets full. The selected frames for caching is indicated by a "*". For both $B^b(t)$ and $B^f(t)$, we can see the robustness, (both equation (8, 10)), is improved.

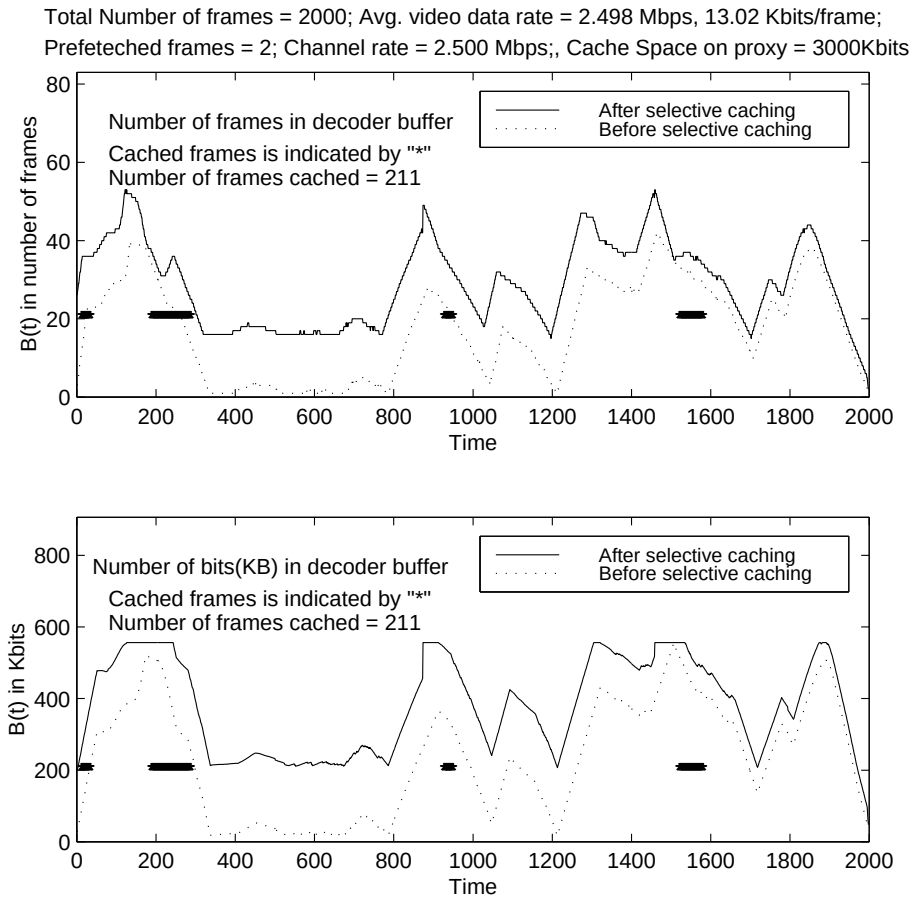

Figure 9: Experiment of selective caching

5 Conclusion

In this paper we address the issues on a proxy to cache only part of video stream object, due to large video data and limited cache space.

We first explain the role of the initial latency and initial buffering in video stream transmission and compression. The analysis and experimental results show that video stream quality can be improved with increased initial latency before playback. We denote this part of the initial segment as required initial segment, related to the compression techniques. We proposed that the required initial segment be cached first, since it is most important for a non jitter playback.

Then we continue choosing other parts of the video stream to be cached, if more space left. Two approaches are presented, additional initial caching and selective caching. Although the first approach is simple, it has the disadvantage of large decoder buffer requirement, and will not increase the robustness of the "tail" of a video stream if only with a small buffer size. The second approach, selective caching, selects the frames to be cached base on the knowledge of user buffer size and video stream properties. It tries to give the maximum benefit to the user, in terms of increasing the robustness of entire video stream against network congestion, while not violating the user buffer size limit. We also point out the Approach I can be seen as a special case of Approach II.

Finally we present the implementation and experimental results for selective caching and other issues. The results verified that our algorithm to choose frames for selective caching is correct for the criteria (equations (8, 10)) of robustness.

References

[1] A. Chankhunthod P. B. Danzig C. Neerdals M. F. Schwartz and K. J. Worrell, "A hierarchical internet object cache," in *USENIX Tech. Conf.*, 1996.

[2] Cisco Caching Engine, "http://www.cisco.com/warp/public/751/cache," .

[3] Netcache Caching Engine, "http://www.netapp.com/products/internet.html," .

[4] J. Shim P. Scheuermann and R. Vingrale, "A case for delay-conscious caching of web documents," in *Proc. Intl. WWW Conf*, Santa Clara, CA, Apr. 1997.

[5] G. Abdulla S. Williams, M. Abrams, and S. Patel, "Removal policies in network caches for world-wide web documents," in *Proc. of ACM SIGCOMM'96*, Stanford, CA, Aug. 1996.

[6] A. Ortega, F. Carignano, S. Ayer, and M. Vetterli., "Soft caching: Web cache management for images," in *IEEE Signal Processing Society Workshop on Multimedia*, Princeton, NJ, June 1997.

[7] Soft Caching Project Page, "http://sipi.usc.edu/ortega/softcaching/," .

[8] X. Yang and K. Ramchandran, "An optimal and efficient soft caching algorithm for network image retrieval," in *Proc. of ICIP*, Chicago, IL, Oct. 1998.

[9] J. Kangasharju, Y. Kwon, A. Ortega, X. Yang, and K. Ramchandran, "Implementation of optimized cache replenishment algorithms in a soft caching system," in *IEEE Signal Processing Society Workshop on Multimedia*, Redondo Beach, CA, Dec. 1998.

[10] S. Acharya, *Techniques for improving multimedia communication over wide area networks*, Ph.D. thesis, Cornell University, 1999.

[11] S.Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *IEEE Infocom99*, New York, USA, March 1999.

[12] Reza Rejaie, Mark Handley, Haobo Yu, and Deborah Estrin, "Proxy caching mechanism for multimedia playback streams in the internet," in *Submitted for review*, Jan. 1999.

[13] J. Mitchell, W. Pennebaker, C. Fogg, and D. LeGall, *MPEG Video Compression Standard*, 1996.

[14] D. Saparilla, K.W. Ross, and M. Reisslein, "Periodic broadcasting with vbr-encoded video," in *IEEE INFOCOMM*, 1999.

[15] M. Reisslein and K.W. Ross, "Join-the-shortest-queue prefetching for vbr encoded video on demand," in *1997 International Conference on Networking Protocols*, Atlanta, October 1999.

[16] J. Salehi, Z. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," in *IEEE/ACM Trans. Networking*, September 1998.

[17] Real Networks, "http://www.real.com," .

[18] C. Y. Hsu, A Ortega, and M. Khansari, "Rate control for robust video transmission over burst-error wireless channels," *IEEE JSAC, Special Issue On Multimedia Network Radios*, 1999.

[19] A. R. Reibman and B. G. Haskell, "Constraints on variable bit-rate video for ATM networks," *IEEE Trans. on Circ. and Sys.*, vol. 2, pp. 361–372, Dec. 1992.

[20] Z. Miao and A. Ortega, "Rate control algorithms for video storage on disk based video servers," in *32nd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, November 1998.

[21] A. Ortega, "Optimal rate allocation under multiple rate constraints," in *Data Compression Conference*, Snowbird, Utah, Mar. 1996.