Playback Restart in Interactive Streaming Video Applications

J. DEY, S. SEN, J. KUROSE,
D. TOWSLEY and J. SALEHI

CMPSCI Technical Report 97-24

Playback restart in interactive streaming video applications*

Jayanta K. Dey Subhabrata Sen James F. Kurose Don Towsley James D. Salehi

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Abstract

Low latency is crucial in networked multimedia applications such as on-demand streaming audio and video. In this paper, we consider the problem of restarting or resuming playback following an interactive operation such as fast forward, rewind, or indexed jump in stored on-demand video.

We present two approaches to restart playback after an interactive operation, and develop algorithms to compute the latencies incurred in restarting playback. Using long MPEG-1 traces, with an optimal smoothing technique [22] to transmit data for regular playback, we demonstrate that the latencies incurred under our approaches are very small. We next examine the latencies incurred when restarting playback in a video server under two policies for sharing its bandwidth: (a) one in which only regular playback bandwidth is available to restart playback, and (b) one in which additional free server bandwidth (if any) is accessible. We find playback restart latencies to be similarly low under both policies, suggesting that the simpler approach (a) is sufficient.

1 Introduction

Until recently, a majority of research in the area of video on demand has focussed on providing continuous delivery of stored video from a server to a client through a network(see [10, 23] for surveys). More recently, there has been increased emphasis on supporting additional streaming video applications (such as WWW-based video, distance learning, training) and on supporting *interactive* operations such as fast forward, rewind, pausing and indexed jumps within the media [5, 6].

In all of these applications, just as it is crucial that the client experience low latencies when initiating video playback, an important consideration with interactive operations is that of *restart* or *resume latency* – the user-perceived delay between the completion of such an interactive operation (e.g., a rewind or indexed jump) and the resumption of "normal" video playout at the client.

*This work supported in part under National Science Foundation grants NCR-9508274 and CDA-9502639. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

In this paper we examine two different approaches for resuming video playout at a client after an interactive operation. The approaches differ in the manner in which they buildup the buffer of received data before resuming playout. We present computational algorithms for computing the restart latency associated with each approach, for any given video trace, and compare the performance of these algorithms using long VBR MPEG-1 encoded video traces. We also assess how resource considerations such as client buffer sizes and network bandwidth for playback affect the performance of these two approaches. Our evaluations indicate that our approaches result in low playback restart latencies. For example, when playback transmission is smoothed [22] to a client with 1 MB client buffer, the average delay in restarting playback is of the order of 1 second. We also find that the restart latencies can be decreased significantly when the bandwidth available for restarting playback is increased only slightly.

We then examine the impact of our approaches on the client latencies under two bandwidth allocation models in a server. Playback being the the primary functionality in a video delivery system, we require that every client should be guaranteed sufficient bandwidth by the server to ensure discontinuity-free and lossless playback. The bandwidth allocation models for restarting playback that we consider are: a *Fixed allocation* model where the client has a fixed amount of bandwidth available to it throughout its session, and a *Variable allocation* model, in which the playback bandwidth is fixed and guaranteed, but a client can use additional bandwidth, if available, when restarting playback. We find that the playback restart latencies are low under both models. This indicates that when our playback restarting algorithms are used in combination with an appropriate smoothed playback transmission schedule, the *Fixed allocation* model can provide video service with low bandwidth requirements for playback as well as low playback restart latencies after an interactive operation.

For video playback, a number of researchers have developed transmission techniques that exploit client buffering capabilities to reduce the peak bandwidth requirements of VBR compressed videos [8, 17, 18, 19, 20, 22]. In particular, the algorithm developed in [22] produces a transmission schedule that minimizes the peak rate as well the rate variability of any video, for a given client buffer size. Because it optimally reduces the peak rate of a video, we use this algorithm as the playback transmission technique throughout this paper. For a discussion on these various smoothing algorithms, see [28] and [7]. We note that although our numerical results are for video traces which are transmitted according to this specific smoothed playback algorithm, the playback restart algorithms and the computational approaches for evaluating these restart algorithms are completely general.

This paper is organized as follows. Section 2 describes the problem setting, formulates the problem and discusses related work on interactivity in video servers. Section 3 describes two playback restarting algorithms as well as the mechanisms to compute the latencies incurred in restarting playback when using these algorithms. Section 4 evaluates the performance of the algorithms on MPEG-1 traces. The performance

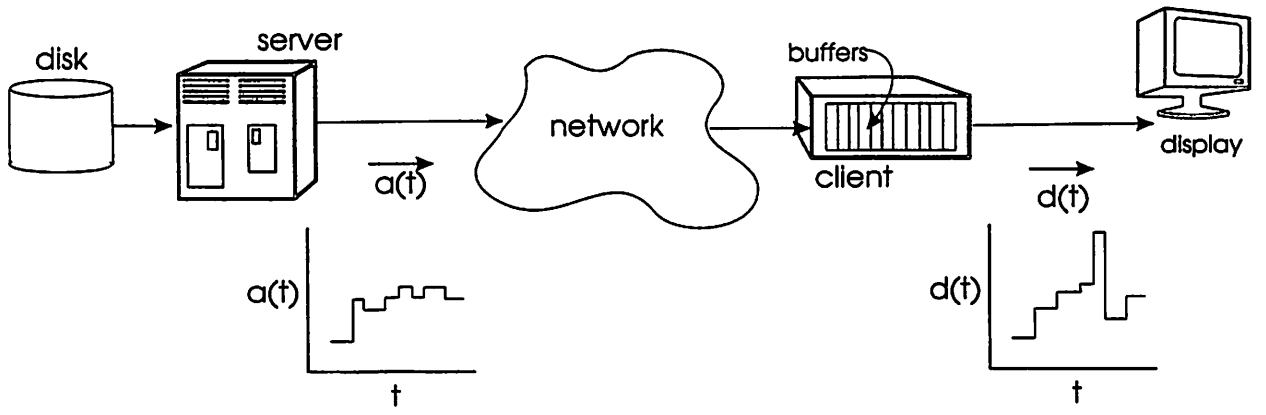


Figure 1: **Video-on-Demand system model**

of the algorithms under the two server bandwidth sharing policies is evaluated in Sections 5 and 6, and Section 7 concludes the paper.

2 Interactive video-on-demand

2.1 Problem Setting

Consider Figure 1, which depicts an end-to-end interactive video delivery system in which a server transmits stored video to a client through a network. Since our discussion in this section will focus on a specific client/server pair, we show only a single client and a single server, although several clients may be interacting with the server. We first consider a system with no interactive operations (i.e., a system in which a client initiates the playback of video that is played through completion).

In the server, data is retrieved from the disk subsystem into its memory buffers, from where it is transmitted into the network, according to some *transmission schedule*. In this paper, we make no assumptions about *how* this transmission schedule is computed. We do note, however, that data must be transmitted in such a manner that *(i)* it is available in the client buffer before the time at which is it scheduled to be displayed by the client (i.e., the client will not starve for data), and *(ii)* the client buffer does not overflow with yet-to-be-displayed data. We assume that the transmission schedule exists at the outset of video playback.

In order to more precisely define a transmission schedule in a system without interactive operations (such as pause, index, fast forward, rewind), we consider, without loss of generality, a discrete-time model with granularity at the video frame level. Thus, for a video, time t is defined as $t \in \{1, \ldots, N\}$, where N is the length of the video in frames. We define $a(t)$ as the amount of data sent by the server during time slot t. The transmission schedule A is defined by $\{a(1), a(2), \ldots, a(N)\}$. Let $A(t)$ represent the cumulative amount of data transmitted by the server during time interval $[1, t]$: $A(t) = \sum_{i=1}^{t} a(i)$.

b	:	Client buffer capacity for storing undisplayed frames.
N	:	Number of frames in a video.
$d(t)$	:	Amount of data consumed by the client in time slot t (size of the t^{th} frame).
$D(t)$	:	Cumulative data consumed by the client over $[1,t]$: $\sum_{i=1}^{t} d(i)$.
$a(t)$	:	Amount of data sent by server in time slot t.
$A(t)$	:	Cumulative data sent by the server during the interval $[1,t]$: $\sum_{i=1}^{t} a(i)$.
$R(t)$	:	Amount of data present in client buffer in time slot t: $A(t) - D(t)$.
$B(t)$	:	Maximum cumulative data that can be received by the client during time interval $[1,t]$, without buffer overflow: $B(t) = \min\{D(t-1)+b, D(N)\}$ for $t = 2,\ldots,N, B(1) = b, B(0) = 0$.

Table 1: **Summary of notation used in this paper.**

Once the data is transmitted though the network, it arrives at the client. The received data is placed in a buffer, from which it is displayed and removed according to the schedule $d(t)$, where $d(t)$ indicates the amount of data displayed during time slot t. We note that $d(t)$ is determined solely by the manner in which the video has been encoded; that is, once a video has been encoded, its display schedule is invariant. $D(t)$ represents the cumulative amount of data displayed by the client during time interval $[1,t] : D(t) = \sum_{i=1}^{t} d(i)$. Therefore $R(t) = A(t) - D(t)$, represents the amount of data present in the client buffer during time slot t.

Finally, the client buffer size is denoted by b. $B(t)$ denotes the maximum cumulative amount of data that can be received by the client over $[1,t]$ without buffer overflow. For a client with fixed buffer size b, $B(t) = \min\{D(t-1)+b, D(N)\}$ for $t = 2,\ldots,N$, with $B(1) = b$, and $B(0) = 0$.

Formally, any transmission schedule $A = [a(1),\ldots,a(N)]$ is a *feasible* video transmission schedule over $[1,N]$, if the client buffer does not starve or overflow during video playback over the time period, i.e., if $D(t) \leq A(t) \leq B(t), \forall t = 1,\ldots,N$. Table 1 summarizes this notation.

2.2 Interactive Operations and restarting Playback

Let us now examine the considerations that arise when interactive operations such as fast forward, rewind, and indexed-jumps can be made. An indexed-jump results in the playback point in the video being moved in the forward or backward direction.

Given the ubiquity of fast forward and rewind operations on videotape systems, surprisingly few researchers have considered the problem of supporting these interactive operations with video on demand. An early work on supporting fast forward and rewind service is [5], which assumed that these operations required the server to transmit the video stream at a higher rate. The focus of that work was on evaluating the number of clients that could be admitted and be provided with statistical quality of service guarantees

in such a system. Researchers have also considered mechanisms to provide fast forward and rewind service using approximately the same bandwidth as in playback [2, 3, 4, 24]. This is accomplished either by sending a smaller number of frames during this operation, or by transmitting an alternate encoding of the video with a coarser granularity and a lower resolution. A third mechanism of providing this functionality is by sending only *key frames* [27], which are representative frames for consecutive video segments. Use of key frames, which are pre-extracted for every video, achieves this functionality with very low bandwidth requirements. Feng et al. [6] evaluate a fast forward and rewind service that is limited to data present within the client buffer, for client buffer sizes of 25 MB or more. It is unclear whether their approach will be adequate for smaller client buffer sizes, e.g., WebTV with a buffer size of 2 MB [26].

Regardless of the manner in which interactive operations such as fast forward, rewind, or indexed-jumps are implemented, the result of each of these operations is the identification of a *new* position in the playback video stream from which normal playout should resume. In this paper, we are concerned with the *restart latency* that a client experiences between the *completion* of an interactive operation and the *resumption* of normal playback. In order to prepare for resumption of playback, the server must transmit video data beginning at this location in the video stream. At some point in time after the client begins receiving this data (the exact value of which will depend on the algorithm used to resume playback and the characteristics of the video itself), the client will actually begin playing out the video. The length of this time interval between the client beginning to receive data and actually beginning playback is defined as the *restart latency* of the client. We will also use the term *resume latency* synonymously. We do not include the network propagation delay as part of this latency since it is independent of any restart algorithm or video. Note that these considerations do not arise for the interactive operation of pause, which temporarily stops a video and resumes from the same position[1].

Before considering specific algorithms for restarting video playback, it is worthwhile to identify some broad goals for a video delivery system that supports interactive operations from clients. From the popularity of streaming audio and video applications such as RealAudio(TM) , VivoActive(TM) and VDOLive(TM), it is clear that clients prefer to wait as little as possible before initiating playback. Similarly, a client would prefer to wait as little as possible before playback is resumed following an interactive operation. Thus, restart latency is a performance measure of primary interest.

From the network and server point of view, however, it is desirable to have as small a change, if any, in the resource or bandwidth requirements in the server or the network to support this playback resumption. When a server is fully loaded, i.e., it is supporting the maximum possible number of clients, any additional resource requirements may cause service disruption to the other clients.

Broadly, there are two ways of resuming transmission to the client:

[1] For live broadcasts, a similar problem arises when playback is resumed after a pause operation.

- By following the original schedule. The advantage of this method is that the resources required to deliver this video do not change at the server or the network.

- An alternative approach, usable in a variable bit rate (VBR) or Renegotiated Constant Bit Rate (RCBR [12, 11]) network service setting, would be to re-calculate a smoothed transmission schedule for the video from the frame position at which playback is to restart. An advantage of this scheme is that the client's playback resume latency can be explicitly controlled in the computation of the new schedule. However the characteristics of this new transmission schedule may be different from those of the original schedule. This would imply that the server and network resource requirements of this video stream may change. If the resource requirements of this new schedule are higher than that of the original one, it may result in a reduction of server and network resource availability for other concurrent video streams.

  A more serious concern is the computation delay associated with calculating a new schedule. For example, on a 150 MHz R4400 processor having 32 MB main memory, computing the optimal smoothed transmission schedule for *Starwars* video clip (171000 frames), assuming a 1 MB client buffer, takes about 6.45 seconds. The time taken is similar for other client buffer sizes as well. This implies that recomputing a new transmission schedule "on the fly" could result in the clients experiencing large delays. Note that it may be possible to compute a partial schedule and then "graft" it onto the original playback transmission schedule, but it is an open question.

This paper thus investigates the client's perceived delays under a scheme in which the original transmission schedule is used throughout a client's session with the server. We demonstrate that the client can resume playback with a very small restart delay using this scheme. In the next section, we describe two algorithms to restart playback using the original transmission schedule.

3 Playback restart algorithms

In this section we describe two algorithms to restart playback following an interactive operation. We also present approaches to compute the restart latency to resume playback from a given frame position in a video, when either of the algorithms is used.

3.1 Algorithm 1

Figure 2 describes the sequence of events when Algorithm 1 is used to handle a client request to restart playback from frame i. Note that some actions are executed in the client while others are performed in the

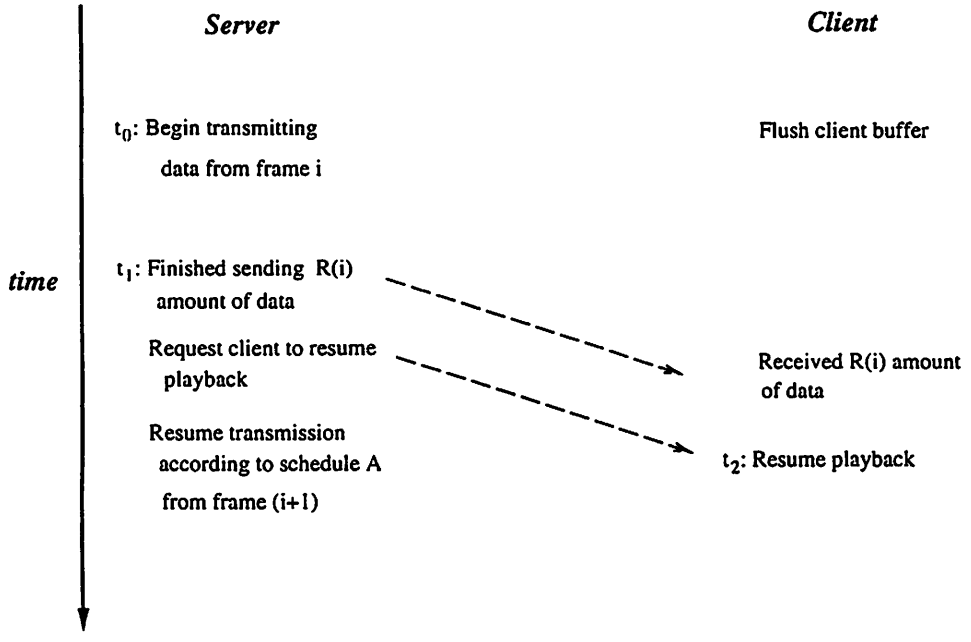


Figure 2: **Algorithm 1 to restart playback from frame position i, given a playback transmission schedule A.**

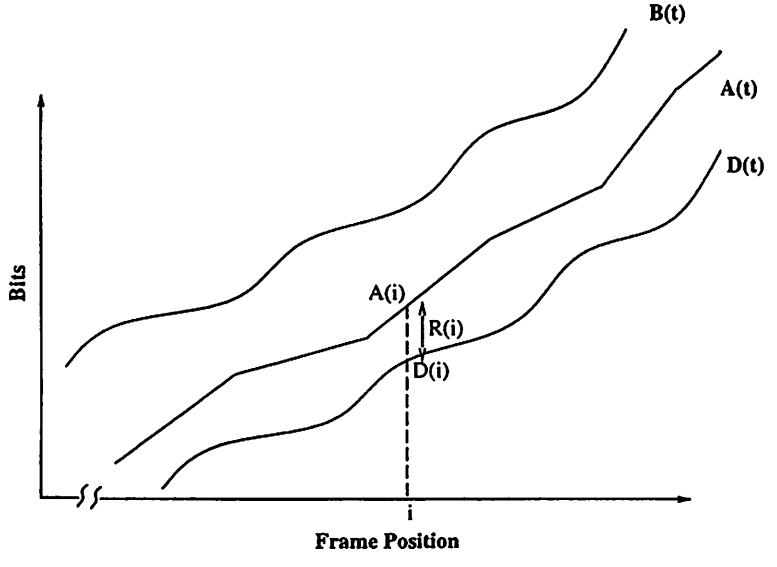


Figure 3: **Client side view of Algorithm 1**

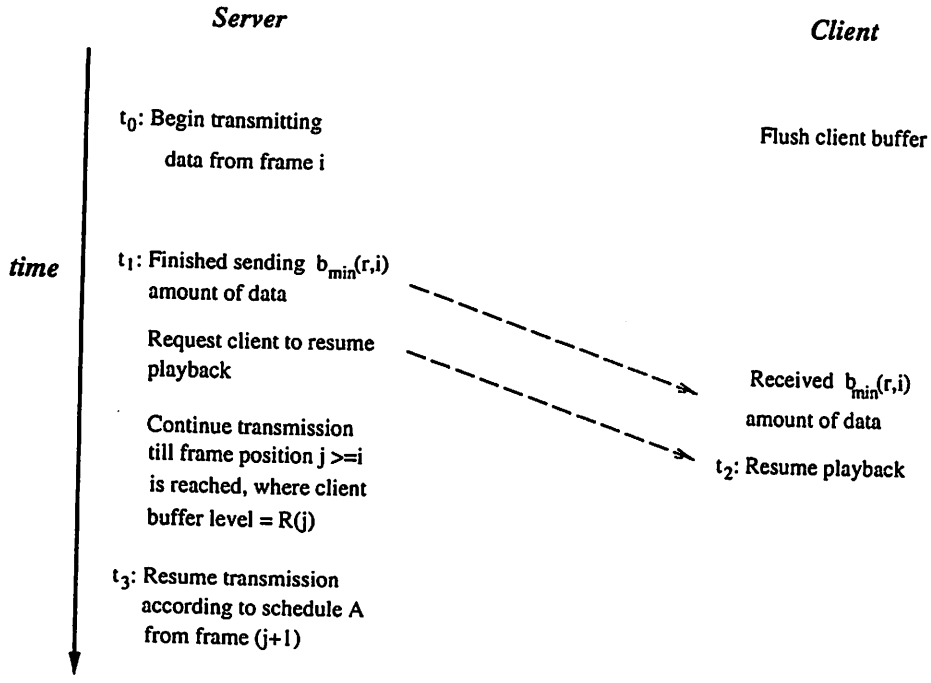


Figure 4: **Algorithm 2 to restart playback from frame position i, given a playback transmission schedule A.**

server. In the figure, time flows from the top to the bottom. The left hand column corresponds to the steps performed at the server, while the right hand column refers to steps performed at the client.

The client flushes its buffer when it requests playback to resume from frame position i. At time t_0, the server starts transmitting video data from frame position i. Recall from Section 2.1 that when playback data is transmitted following schedule A, $R(i) = A(i) - D(i)$ must be the amount of data present in the client buffer when frame i is displayed (see Figure 3 for a client side view). Thus, at time t_1, after the amount of data $R(i)$ has been transmitted to the client, the server requests the client to resume playback. The server then continues data transmission following schedule A from frame index $(i + 1)$. When the client receives the request to restart playback at time t_2, its buffer has $R(i)$ amount of data. Therefore resuming playback from frame i at time t_2, with the server transmitting according to schedule A, will ensure that playback will continue with no underflow or overflow at the client buffer.

The delay in transmitting $R(i)$ amount of data is $(t_1 - t_0)$. As mentioned earlier, the additional delay of $(t_2 - t_1)$ for the client to restart playback is the propagation delay across the network, and is independent of any algorithm used. Thus the inherent restart latency in this algorithm is the time taken to fill the client buffer up to level $R(i)$. If the rate available to fill up the buffer is r, $R(i)/r$ is the latency to resume playback[2]. In our evaluations, we consider only the inherent restart latency of the algorithm.

[2]Note that the restart signal to the client can be piggybacked along with the data transmission, allowing it to restart playback as soon as the buffer is filled up to the requisite level.

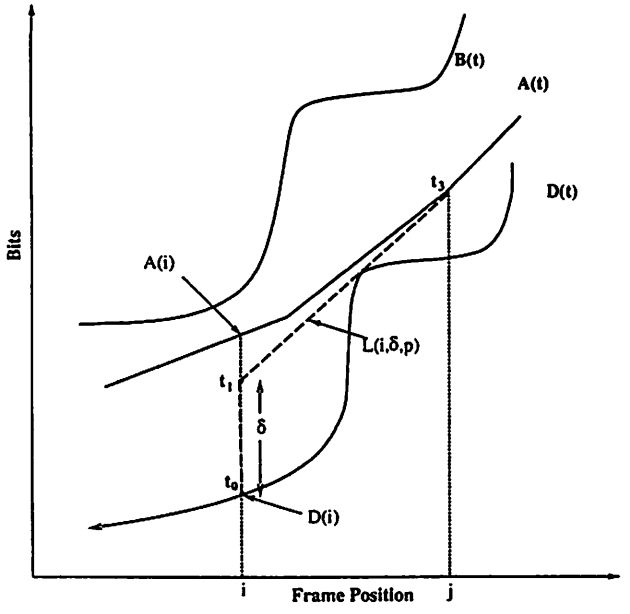


Figure 5: **Server side view of Algorithm 2**

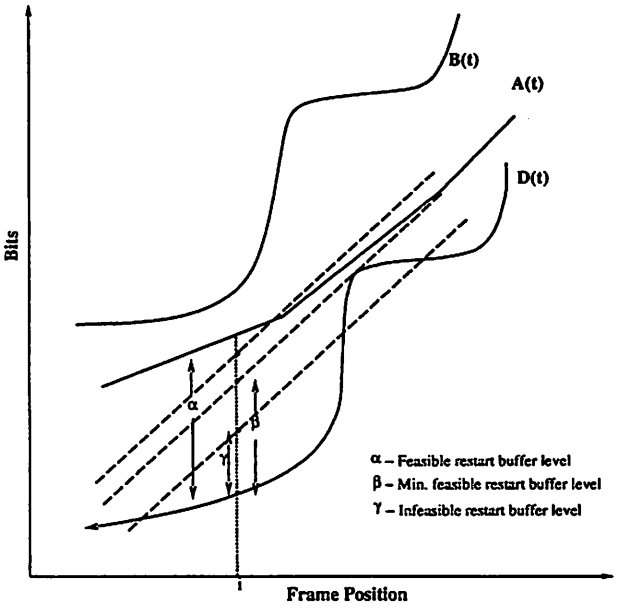


Figure 6: **Examples of feasible and infeasible ramps**

Thus the restart latency of this algorithm is dependent upon both the amount of data that needs to be sent to and received by the client before playback can begin, as well as the rate at which this amount of data is sent. Increasing the client buffer size b can increase the maximum possible value of $A(i)$ as computed for the original playback schedule (e.g., under the optimal smoothing approach [22]). This in turn, increases $R(i)$ as well as the restart latency given the same rate r. Similarly, decreasing the rate r increases the latency when the client buffer needs to be filled to the same level.

While the advantage of this scheme is its implementation simplicity, a drawback is potentially high playback restart delays, particularly when the value of $R(i)$ is large or the rate r is small. We shall use Algorithm 1 as a baseline algorithm to compare with Algorithm 2 which, as we will see, achieves lower restart latencies.

3.2 Algorithm 2

The latency to restart playback from frame position i will decrease from that incurred in Algorithm 1, if the client can resume playback before its buffer level reaches $R(i)$. In the second algorithm, described in Figure 4, *the client buffer is filled up as little as possible* before the client resumes playback, i.e., to the minimum level which avoids client buffer starvation when the server transmits at rate r until convergence with the original transmission schedule.

As in Algorithm 1, the client flushes its buffer when it requests playback to resume from frame position i. Upon receiving a request to restart playback, the server starts transmitting data from frame i at time t_0.

By time t_1, an amount of data $\delta \leq R(i)$ is transmitted at rate r. The server then requests the client to resume playback, which the client does at time t_2, after receiving the request. The latency to restart playback in this algorithm, $(t_1 - t_0)$, is δ/r. Again we do not consider the network propagation delay to be an inherent delay of the algorithm.

If the level of the client buffer is less than $R(i)$, at time t_2, the server transmission cannot yet proceed according to schedule A. Instead, the server continues to transmit at rate r, until the client buffer level equals $R(j)$ for some $j \geq i$. At this time, the server switches to playback transmission schedule A beginning with index $(j+1)$. To ensure that the server transmission can converge with the original playback transmission schedule without any starvation at the client, it is necessary that $r \geq |a(k)| \ \forall k, i \leq k \leq j$. This can be satisfied by choosing the transmission rate r to be at least as large as the peak rate of the playback transmission schedule A.

Figure 5 presents the server side view of this algorithm. The curves $B(t), A(t)$ and $D(t)$ are plotted in bits against time measured in frame periods. At time t_0, the server starts transmitting data from frame position i at rate r. At time t_1, after δ units of data are sent, the server requests the client to resume playback. Thus the client starts displaying/consuming data from position i according to the curve $D(t)$. The server continues to transmit at rate r. Defining $L(i, a, p) = D(i) + a + r \cdot (p - i)$ to be a line of slope r passing through $(i, D(i) + a)$, $L(i, \delta, p)$ denotes this server transmission schedule. As stated before, the server reverts to transmission schedule A at time t_3 from frame position $j = \min\{p \geq i \mid L(i, \delta, p) \geq A(p)\}$ (see Figure 5). Note that it is necessary for $L(i, \delta, p)$ to lie above $D(p)$ in the range $[i, j]$ in order for the client buffer not to underflow, i.e., the client sees continuous playback. The server then proceeds to continue video transmission according to schedule A from frame position $(j+1)$.

We refer to the period of time, during which the server continues to transmit at rate r prior to reverting to schedule A, as the period during which the server *ramps up* to the transmission schedule. This period is represented in Figure 5 by the projection of the line $L(i, \delta, p)$ on the Frame Position axis.

We say that a buffer level δ is a *feasible restart buffer level* given a playback restart bandwidth of r and a restart position i, if the client is able to restart playback after its buffer is filled to level δ and does not subsequently underflow or overflow. In Figure 6, α and β are feasible restart buffer levels for a given rate r, since the client can begin playback of the video at either of these buffer levels while the server ramps up to the playback transmission schedule as shown in the figure. However, γ is an infeasible restart buffer level, since, if the client begins playback from that level, while the server continues transmission at rate r, a buffer underflow occurs as shown in the figure.

Central to Algorithm 2 is computing, for every frame position i, the *minimum feasible restart level* δ, denoted by $b_{min}(r, i)$. The simplest method to compute $b_{min}(r, i)$ is to perform a binary search over the interval $[0, R(i)]$, and for each possible buffer level, check to see if a buffer underflow occurs before the

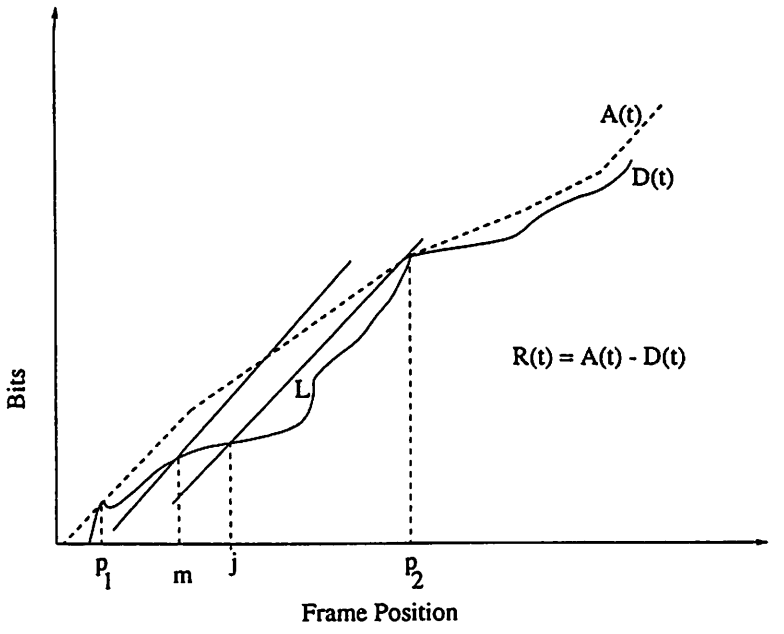


Figure 7: **Diagrammatic representation of Algorithm 2**

server ramps up to the playback transmission schedule. This checking takes time $O(N)$, while the binary search takes time $O(\log b)$. Thus computing this minimum buffer level takes time $O(N \log b)$ for every frame position i, i.e. $O(N^2 \log b)$ for the entire video.

In the next subsection, we describe a method for computing $b_{min}(r,i)$ $\forall 1 \leq i \leq N$, i.e. for the entire video in $O(N)$ time.

3.3 Computing Restart Latencies under Algorithm 2

Suppose playback is to restart from frame position i. The server fills the client buffer up to level δ, requests the client to restart playback, and continues transmission of the video at rate r. If δ is a feasible restart buffer level, then, $c(\delta, r, i)$, the video frame position at which the server will ramp up to the transmission schedule A (see Figure5), is

$$c(\delta, r, i) = \min\{j \geq i \mid A(j) \leq L(i, \delta, j)\}.$$

If instead, δ is not feasible, then, $s(\delta, r, i)$, the video frame position at which the client buffer will underflow (see Figure 6), is

$$s(\delta, r, i) = \min\{j \geq i \mid D(j) > L(i, \delta, j)\}.$$

Since $A(i) \geq D(i)$ for all frames i, the definitions of $c(\delta, r, i)$ and $s(\delta, i, r)$ imply the following simple property:

Property 1: The buffer level δ is a *feasible buffer restart level* for playback to restart from frame position i with the server ramping up at rate r iff $c(\delta, r, i) < s(\delta, i, r)$.

The proof of this property is trivial.

The *minimum feasible restart buffer level* $b_{min}(r, i)$ is then defined as:

$$b_{min}(r, i) = \min_{0 \leq \delta \leq R(i)} \{\delta | c(\delta, r, i) < s(\delta, r, i)\}.$$

To compute the minimum feasible restart buffer level and the playback restart latency, Algorithm 2 presented below relies on transmission schedules that have the client buffer empty for at least two frame positions, i.e., $R(i) = 0$ for at least two values of i. For any arbitrary feasible playback transmission schedule, we are guaranteed that the client buffer is empty at the beginning and the end of a video, i.e., $R(i) = 0$, for at least $i = 0$ and $i = N$. Note that in the optimal smoothed transmission schedule (considered in Section 4), there are typically several other frame positions i for which $R(i) = 0$.

Consider two successive frame positions p_1 and p_2, $p_1 < p_2$, where the client buffer is empty, as shown in Figure 7. Therefore $R(p_1) = R(p_2) = 0$ and $R(p) > 0 \ \forall p_1 < p < p_2$. Since $R(p_1) = R(p_2) = 0$, therefore $b_{min}(r, p_1) = b_{min}(r, p_2) = 0$. Let j be the largest frame position less than p_2 s.t. $L(p_2, 0, p)$ intersects D either at frame position j or in between frame positions $(j-1)$ and j (see Figure 7). Thus

$$j = \max\{k | D(k) > L(p_2, 0, k); k < p_2\} + 1.$$

Property 2: Let j be as defined above. For any frame k, $j \leq k < p_2$, $b_{min}(r, k) = L(p_2, 0, k) - D(k)$.

Proof: Suppose that the property were wrong, i.e. $b_{min}(r, k) = \delta < L(p_2, 0, k) - D(k)$ for some k, $j \leq k < p_2$. Construct a new line, $L(k, \delta, p)$. It follows that $L(k, \delta, p) < L(p_2, 0, p) \ \forall p$; hence $L(k, \delta, p_2) < L(p_2, 0, p_2) = D(p_2)$. This implies that $c(\delta, r, k) > s(\delta, r, k)$. From Property 1, this is a contradiction. ∎

The above property can be generalized as follows.

Property 3: Assume that position l is such that $b_{min}(r, l) = 0$, and $b_{min}(r, k) > 0$ for $k = j, j+1, \ldots, l-1$ for some $j < l$. Then, for position $(j-1)$,

$$b_{min}(r, j-1) = \max\{0, L(l, 0, j-1) - D(j-1)\}.$$

Proof: Suppose the property were wrong, i.e. $b_{min}(r, j-1) > 0$ but $b_{min}(r, j-1) = \delta < L(l, 0, j-1) - D(j-1)$. Construct a new line, $L(j-1, \delta, p)$. It follows that $L(j-1, \delta, p) < L(l, 0, p) \ \forall p$; hence $L(j-1, \delta, l) < L(l, 0, l) = D(l)$ which implies that $c(\delta, r, j-1) > s(\delta, r, j-1)$. From Property 1, this is

```
PROCEDURE ComputeB(A, D, r)
BEGINPROC
1.  Going through A,
2.  PUSH(stack, i) ∀ i s.t. A(i) = D(i).
3.  p1 = pop(stack)
4.  WHILE (stack not empty)
5.  BEGIN
6.    p2 = pop(stack)
7.    k= p1
8.    WHILE (k > p2)
9.    BEGIN
10.     IF ((Δ = D(k) − r) ≤ D(k − 1))
11.     BEGIN
12.       b_min(r, k) = 0; k = k − 1;
13.     ENDIF
14.     ELSE
15.     BEGIN
16.       /* Generate b_min for all appropriate points */
17.       b_min(r, k) = 0;
18.       FOR (l = k − 1; Δ >= D(l); l = l − 1)
19.       BEGIN
20.         b_min(r, l) = (Δ − D(l));
21.         Δ = Δ − r;
22.       ENDFOR
23.       k = l;
24.     ENDELSE
25.   ENDWHILE
26.   p2 = p1;
27. ENDWHILE
ENDPROC
```

Figure 8: **Algorithm to compute $b_{min}()$ for every frame position for a video given D, A, and a restart rate r.**

a contradiction. ∎

Exploiting the above recursive property gives us an efficient way to compute the playback restart latency for all frames k, starting from the last frame position N where $b_{min}(r, N) = R(N) = 0$, and iterating backwards. Consider frame position m, s.t. $b_{min}(r, m) = 0$. Draw line $L(m, 0, k)$. If $L(m, 0, m-1) > D(m-1)$, then from Property 2 we have $b_{min}(r, m-1) = L(m, 0, m-1) - D(m-1)$. Now Property 3 can be utilized iteratively to $b_{min}(r, k)$ for all appropriate frame positions k.

This computation is detailed in Figure 8. In lines 1-2, each of the frame indices j where $R(j) = 0$ are pushed into a stack. Thus lines 3 and 6 extract frame positions p_1 and p_2, $p_1 > p_2$, such that $R(p_1) = R(p_2) = 0$, and $R(j) > 0$, $\forall j$, $p_2 < j < p_1$. This allows Properties 2 and 3 to be applied within the *WHILE* loop from lines 8 through 25.

Line 10 checks to see whether $D(k-1)$ lies above line $L(k, 0, k-1)$. If so, we assign $b_{min}(r, k-1) = 0$.

If this condition does not hold, Property 3 is applied from lines 18 through 24. Line $L(k, 0, p)$ of slope r is drawn starting from frame position k (line 21) and is extended backwards until it hits D (termination condition of *FOR* loop in line 18), calculating $b_{min}(r, l)$ for every frame position l in between (line 20).

In Line 26, p_2 is made the endpoint of the next segment considered, and the entire process is repeated. Since this computation needs only $O(1)$ computation for every frame position, computing $b_{min}(r, k)$ takes time $O(N)$ for all $1 \leq k \leq N$.

Note that this minimum restart latency exhibits another interesting property. In Algorithm 1, the restart latency is inversely proportional to rate r. In this case, if a rate $r' > r$ is available for playback restart, the line with slope r' drawn through $(p_2, D(p_2))$ in Figure 7 would meet D at frame position $j' > j$. Thus there would be several frame positions k, $j < k < j'$, which had $b_{min}(r, k) > 0$, but will have $b_{min}(r', k) = 0$. Additionally, for frame positions k' such that $j' < k' < p_2$ and $b_{min}(r', k') > 0$, not only will the value of $b_{min}(r', k')$ be less than $b_{min}(r, k')$, but also this smaller buffer amount will be filled at a higher transmission rate r'. Hence this decrease in restart latency will be more than the corresponding decrease in Algorithm 1, where the restart latency is proportional to rate r.

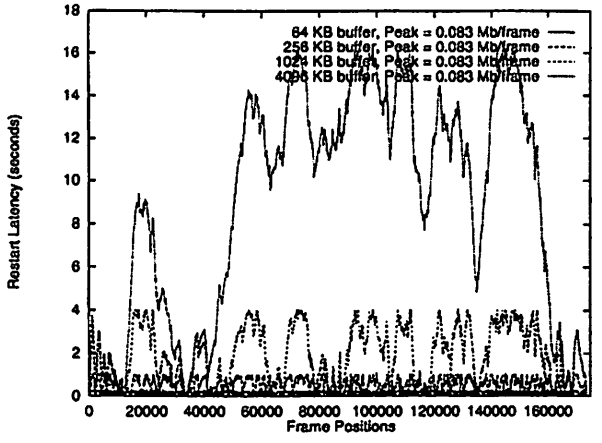
Thus in this algorithm, when the available rate r for playback restart increases to r', the number of frames that have a zero restart latency increases. Also, for the frame positions whose latencies remain non-zero, the reduction in restart latency is caused by two factors: a decrease in the value of $b_{min}(r', k)$, and an increased available rate r' to reach this buffer level. The effect of this is demonstrated in Section 4.

4 Empirical evaluation

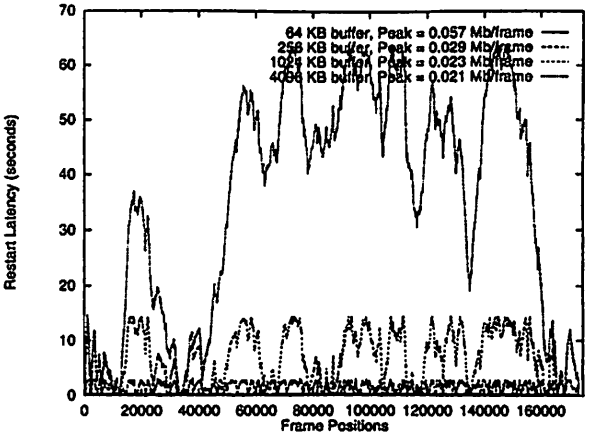
We have evaluated the restart delays seen under Algorithms 1 and 2 using VBR MPEG-1 traces of *Beauty and the Beast* [15], *Jurassic Park* [21], *Advertisements* [15], *Wizard of Oz* [16] and *Star Wars* [9]. In this section we describe the restart delay seen under the two algorithms, on two representative traces. *The Wizard of Oz* trace is 23 minutes in length with a mean rate of 1.25 Mb/s, while the *Star Wars* trace is 119 minutes long with a mean rate of 0.374 Mb/s.

To investigate the performance under the algorithms, we assume that the videos are transmitted using the optimal smoothing algorithm [22], since it minimizes the peak rate as well as the rate variability. We study the effects when the bandwidth available to the algorithms for playback restart is the peak rate of the corresponding smoothed playback transmission schedule. We also explore the benefits of additional bandwidth being available for restarting playback for Algorithm 2. Additionally, it is assumed that when playback is resumed, it occurs from the nearest previous I frame, unless the frame itself is an I frame.

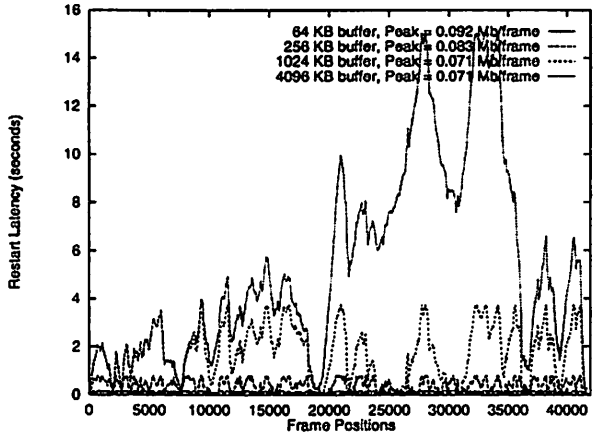
It is worthwhile to distinguish between the terms *playback initiation latency* and *playback restart*



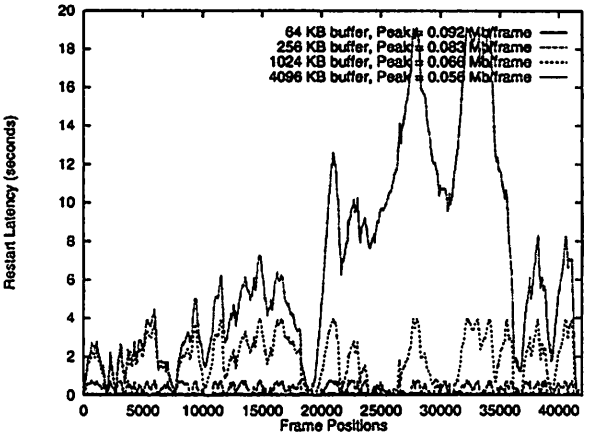
(a) *Starwars*: Initiation latency 0 frames



(b) *Starwars*: Initiation latency 10 frames



(c) *Wizard*: Initiation latency 0 frames



(d) *Wizard*: Initiation latency 10 frames

Figure 9: **Algorithm 1 performance on *Starwars* and *Wizard of Oz*.**

(resume) latency. In the optimal smoothing algorithm, a playback transmission schedule is said to have a *playback initiation latency* of s frames if the client starts playback s frame periods after receiving the first frame, which due to the MPEG compression technique is typically much larger than the immediate subsequent frames. Thus this playback initiation latency enables smoothing or amortizing the transmission of the first frame over s frame periods, reducing the transmission rate of the initial segment which can otherwise be quite large [22]. This latency is referred to as the playback initiation latency in the rest of this paper and is to be distinguished from the playback restart latency which is incurred when restarting playback following an interactive operation.

4.1 Performance of Algorithm 1

Figure 9 illustrates the performance of Algorithm 1 on *Starwars* and *Wizard of Oz*, smoothed for playback initiation latencies of 0 and 10 frame periods, and for client buffer sizes of 64KB, 256 KB, 1MB and 4MB. Each point in a curve represents the resume latency incurred when playback is restarted from the corresponding frame number. There are several interesting points to note in these graphs.

In all the curves, there are segments with negative slopes where the latencies decrease with increasing frame positions (sometimes all the way down to 0). They correspond to sections in the transmission schedule where R decreases with increasing frame position. Similarly, the segments in the curves with positive slopes correspond to portions in the transmission schedule where R increases. This behavior results from the optimal transmission schedule attempting to minimize the rate variability. The maximum latency is experienced by a client at a frame position where the value of R equals the client buffer size. We expect to see this behavior when most other reasonable smoothing algorithms are used for playback transmission, e.g. [8], as they attempt to take advantage of the client buffer to smooth the variability of a VBR video.

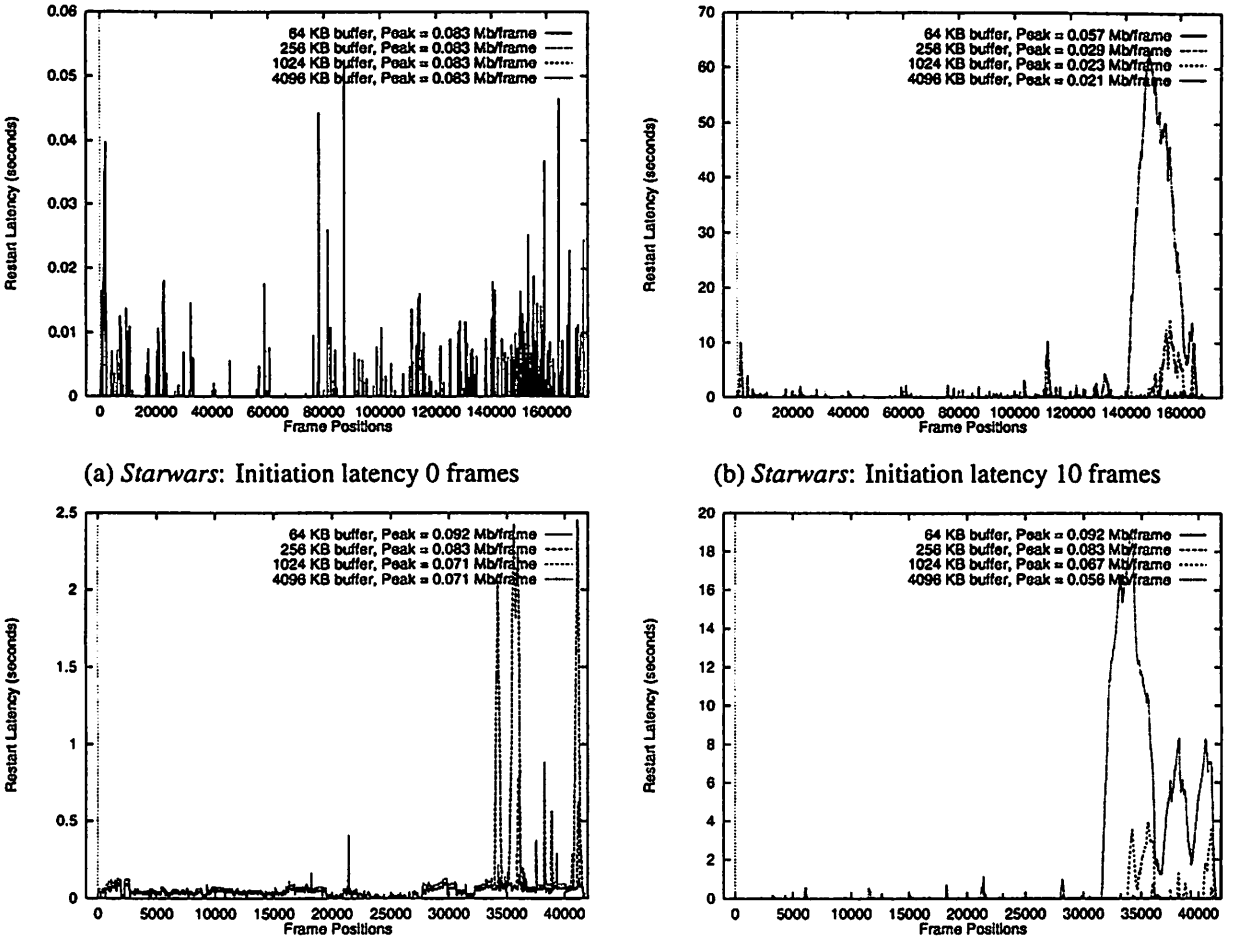
Figure 9(a) plots the performance of this algorithm on the *Starwars* trace smoothed for 0 playback initiation latency. The peak rate of the smoothed schedules for each of the 4 client buffer sizes, caused by the size of the first frame [22], is 0.083 Mb/frame.

First, we note that the restart latencies experienced by the clients are vastly different from each other and are roughly proportional to the size of the client buffer. The maximum restart latency seen by a 64KB client is 0.254 s, while those seen by a 1MB and 4MB client are 4.03 s and 16.12 s respectively. The corresponding average restart latencies are 0.126 s, 2.03 s and 9.05 s.

In figure 9(a) , the shape of the curves corresponding to the four client buffer sizes look very similar to each other. Owing to the size of the first I frame of *Starwars*, the optimal smoothed schedule for a playback initiation latency of 0 produces the same peak rate for all the client buffer sizes [22]. As per the discussion in Section 3.1, the differences between the four latency curves in figure 9(a) is primarily owing to the difference in the buffer sizes, resulting in their similar contours[3].

A playback initiation latency of 10 frame periods (417 ms for 24 fps *Star Wars*) smoothes out the effects of the first frame. Thus, with increasing client buffers, the optimal schedules become smoother and the peak rates smaller [22] as illustrated in Figure 9(b). The peak rate for the 4 MB-smoothed schedule is only 0.021 Mb/frame period, whereas it is 0.057 Mb/frame for the 64 KB-smoothed schedule. Low bit rates are clearly desirable for video playback both by the server and the network. However, in the 4 MB-smoothed schedule, a low peak bandwidth availability, combined with a large client buffer, causes the playback restart latencies

[3]Note that the smoothed schedules are not simple multiples of schedules corresponding to smaller client buffer sizes, even with identical peak rates.



(a) *Starwars*: Initiation latency 0 frames

(b) *Starwars*: Initiation latency 10 frames

(c) *Wizard*: Initiation latency 0 frames

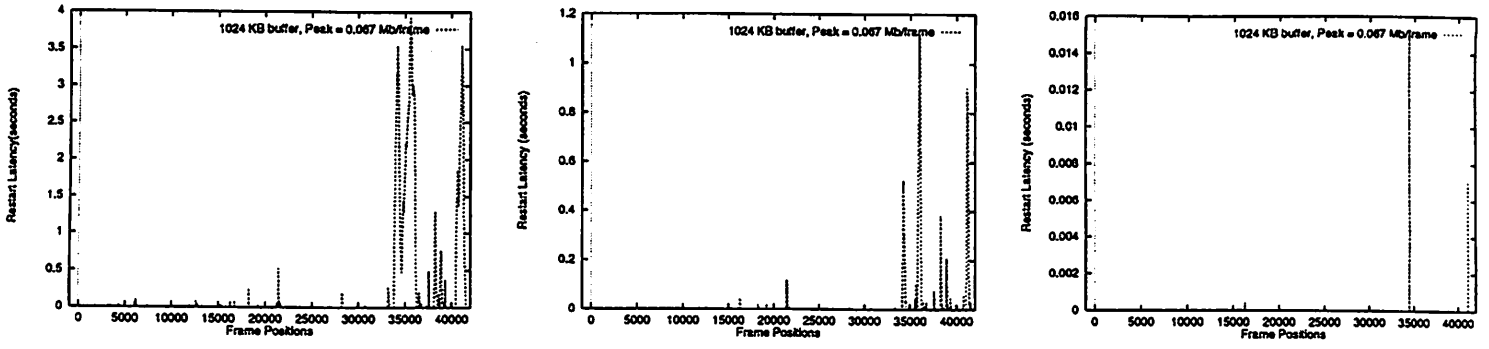
(d) *Wizard*: Initiation latency 10 frames

Figure 10: **Algorithm 2 performance on *Starwars* and *Wizard***

to increase dramatically, resulting in latencies in the 60-65 s range (maximum 63.69 s, average 35.79 s). Users resuming playback from the corresponding frames will find these latencies unacceptably high.

The marginal reduction in the peak rate of the smoothed video decreases with increasing buffer size [22]. In Figure 9(b), the peak rate of the 1 MB-smoothed stream is 0.023 Mb/frame compared to 0.021 Mb/frame peak rate for the 4 MB-smoothed stream. But since the buffer size is 75% smaller, the latencies are in the much more tolerable 10-16 s range (maximum 15.91s, average 7.29 s). Latencies for both the 256 KB- and the 64-KB smoothed schedules are much lower (maximum 2.98 s and 0.37 s, average 1.54 s and 0.19 s), made possible by the dual effect of both much smaller buffers and higher peak rate availability.

Figures 9(c) and 9(d) graph the playback resumption latencies for the movie *Wizard* for schedules smoothed for playback with initiation latencies of 0 and 10 frame periods (0.333 s for 30 fps *Wizard*) respectively. Note that for *Wizard*, unlike *Starwars*, the peak bandwidth of the smoothed schedules for different sized client buffers are different. This has two notable implications:



(a) Bandwidth = Playback bandwidth (b) Bandwidth = 1.2× Playback bandwidth (c) 2× Playback bandwidth

Figure 11: **Effect of additional bandwidth in Algorithm 2**

- Even for schedules smoothed for playback initiation latencies of 0, the playback restart latencies from a frame i for different buffer sizes is dependent on both the client buffer size and the peak rate, and not on the buffer size alone as in *Starwars*.

- Unlike *Starwars*, the peak rate of the smoothed schedules changes by a much smaller amount from Figure 9(c) to Figure 9(d) for all four buffer sizes. This implies that the resume latencies also do not change significantly, as shown in the figures. For example, the worst playback restart latency for the 4 MB-smoothed schedule for the 0 frame and 1 frame playback initiation latency cases are 15.05 s and 19.18 s respectively.

Wizard, encoded at a higher bit rate than *Starwars*, has relatively higher peak rates for the smoothed schedules. Hence higher bandwidth is available for playback resumption, leading to lower restart times for similar buffer sizes (compare Figures 9(b) and 9(d)). But the latencies experienced by a 4 MB-smoothed client can be higher than 14 s, as illustrated in both Figures 9(c) and 9(d). The latencies experienced by a 1 MB- smoothed client are much more tolerable, being less than 10 s. Of course, the latencies for the 256 KB- and 64 KB-smoothed clients are much lower, but they come at the cost of requiring a higher playback bandwidth from the network, and higher resource requirements from the server.

In summary, Figure 9 illustrates that while a server and the network might benefit from larger client buffers resulting in lower peak rates of the smoothed playback transmission schedules, a client using Algorithm 1 would have to pay the price of having a large buffer by incurring high latencies for resuming playback after an interactive operation. *Noting the diminishing marginal peak rates with increasing client buffer sizes, a reasonable compromise with Algorithm 1 would be to find a buffer size for which most of the benefits of peak rate reduction in the optimal smoothed schedule is achieved, and for which the client restart latencies remain within a tolerable threshold.* In both the videos above, a 1 MB client buffer seems to be the appropriate choice, when combined with an optimal smoothed schedule for a 10 frame period playback

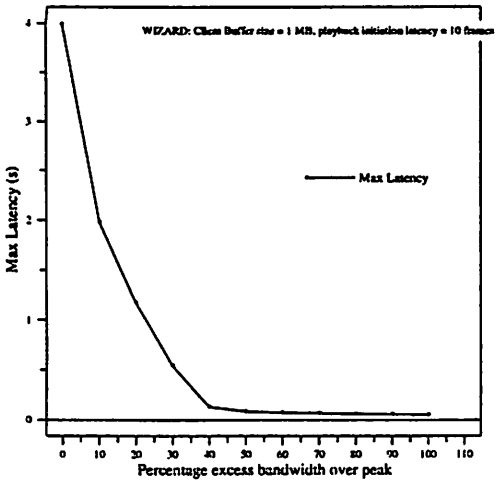


Figure 12: **Max. Latency vs. excess bandwidth**

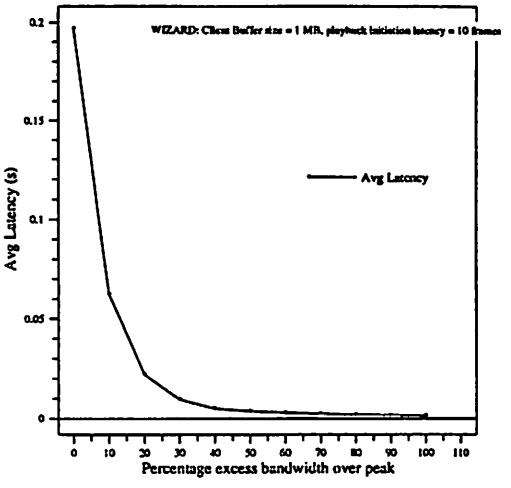


Figure 13: **Avg. Latency vs. excess bandwidth**

initiation latency.

4.2 Performance of Algorithm 2

Figure 10 illustrates the performance of Algorithm 2 on *Starwars* and *Wizard of Oz*, smoothed for playback initiation latencies of 0 and 10 frame periods, and for client buffer sizes equal to 64KB, 256 KB, 1MB and 4MB. Figure 10(a) analyses *Starwars* for the four client buffer sizes for smoothed schedules designed for playback initiation latency of 0 frame period. Figure 10(b) shows the client restart latencies for playback initiation latency of 10 frame periods. Figures 10(c) and 10(d) are the analysis for *Wizard* for playback initiation latencies of 0 and 10 frame periods respectively.

Each curve plots the playback resumption latency as a function of the frame positions of a video, for a schedule optimally smoothed for a particular client buffer size and a particular playback initiation latency. Each point in a curve represents the resume latency incurred when playback is restarted from the corresponding frame number. Note that the restart bandwidths available to Algorithm 2 are identical to the corresponding restart bandwidths available to Algorithm 1 in the previous subsection.

The most salient feature in all the curves in all four graphs is the predominance of zero and "near-zero" latency points. Users incur very little delay when resuming playback under Algorithm 2. Also, under Algorithm 2, when an interactive request is made, the probability of enduring a high latency to resume playback is small. *Thus we can expect very low average latencies over a video session when this scheme is used to resume playback. The large density of low latency points also clearly demonstrates the superiority of Algorithm 2 over Algorithm 1.*

Under Algorithm 2, the curves for a 1 MB buffer client in Figures 10 (a), (b), (c) and (d) have maximum

latencies of 0.056 s, 14.39 s, 2.45 s and 3.99 s respectively. The corresponding average latencies are 7.5×10^{-5} s, 0.55 s, 0.09 s and 0.20 s respectively.

Thus, as in the case of Algorithm 1, the dual goals of low bandwidth for playback and low latency for playback restart can be achieved by sizing the client buffer to 1 MB.

We now highlight characteristics of the four graphs individually. In Figure 10(a), the restart latencies for each video frame is negligible (< 0.06 s) for all four curves. This is due to the identical peak rates (0.083 Mb/frame period) for all the client buffer sizes.

In Figure 10(b), the restart latency in the case of the 4 MB-smoothed schedule can be as high as 63.69 s, the same as the corresponding maximum for Algorithm 1. In fact, the downward sloping region immediately following frame number 148459 (where the maximum latency occurs in both curves in both Figures 9(b) and 10(b)), is similar in Figures 9(b) and 10(b). This region corresponds to the peak rate segment (0.021 Mb/frame period) of the smoothed schedule A. In this segment, the only way to ramp up to A with a bandwidth equal to the peak rate, is by making the client wait until R amount of data is present in the buffer. Hence algorithm 2 and algorithm 1 perform identically in this region. In the optimal smoothed schedule, the peak rate segment starts when the client buffer is full (maximum latency point), and ends when the client buffer is empty (zero latency point) [22]. This explains the similarity of the downward sloping section following frame number 148459.

A 10 frame period playback initiation latency reduces the peak smoothed bandwidth for client buffer sizes of 1 MB and 4 MB in Figure 10(d). The maximum restart latency of the 4 MB curve is 19.18 s, which can be considered to be beyond the tolerable threshold. Once again, the maximum latency corresponds to the start of the peak rate segment of the optimal smoothed curve and the immediately following downward sloping region is similar to the corresponding region in Figure 9(d).

The behavior described above is not seen in the graphs for the smoothed schedules computed for playback initiation latency of 0 for the following reason. For a initiation latency of 0, the peak rate occurs in the first frame, and lasts for exactly one frame. For playback to start, $D(1)$ must equal $A(1)$, i.e., the first frame must be completely present in the client buffer.

Finally, as described in Section 3.2, the latency to restart playback from any frame position reduces dramatically when the bandwidth available *during the restart period only* is increased slightly. Figure 11 illustrates this characteristic by comparing the resume latencies in *Wizard* for a transmission schedule with a 10 frame period playback initiation latency, smoothed for a client with a 1 MB buffer. In Figure 11(a), the bandwidth available for playback restart is the peak bandwidth of the 1 MB-smoothed curve (0.067 Mb/frame), while the bandwidths in Figures 11(b) and 11(c) are 1.2 and 2 times this peak bandwidth. We focus on two specific characteristics evidenced in the graphs.

First, as explained in Section 3.3, note that the peak as well as the average latencies are dramatically reduced as additional bandwidth is available for restarting playback. Adding a mere 20% extra bandwidth brings the peak latency down by a factor of 3.41 (from 3.99 s to 1.17 s), while providing 100% extra bandwidth reduces the peak by a factor of 79.80 (from 3.99 s to 0.05 s). Similarly, the average bandwidths go down from 0.20 s to 0.022 s and 0.001 s with an additional 20% and 100% bandwidth respectively.

Second, as explained in Section 3.3, the number of frame indices, from where the restart latency is zero, are increased significantly with increased bandwidth. An extra 20% bandwidth increases the fraction of zero-latency indices from 61.36% to 74.02%, while adding 100% extra bandwidth increases the fraction to 91.31%.

Thus the combined effect is that the non-zero latency regions of the graph are compressed both horizontally and vertically, as we go from Figure 11(a) to 11(c).

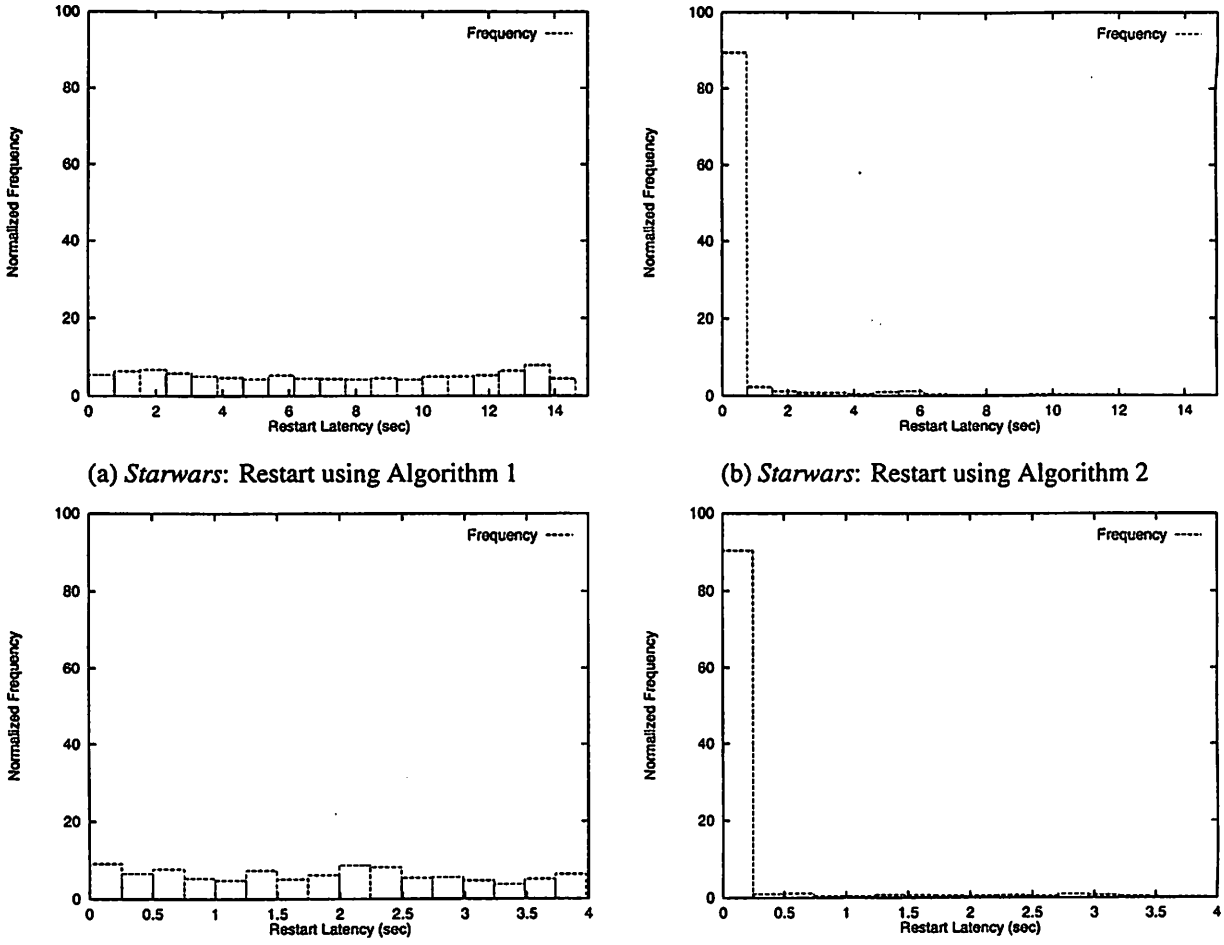
Figures 12 and 13 plot the maximum and average restart latency seen as the bandwidth available during restarting playback is increased from the peak playback transmission bandwidth to twice that amount. The video chosen is *Wizard* transmitted to a 1-MB client buffer with a 10 frame period playback initiation latency. In both the graphs, the decrease in latency is dramatic when the bandwidth is increased slightly above the peak transmission bandwidth, e.g., going from 0% to 10% excess bandwidth. Then the effects are less marked, resulting in both curves being convex. These indicate that in order to realize most of the benefit of this super linear decrease in latency of using Algorithm 2, only a small fractional amount of bandwidth over the peak (smoothed) playback bandwidth needs to be provided to restart playback.

4.3 Comparing Algorithms 1 and 2

We conclude this section by comparing the performance of Algorithm 1 and 2 using the histograms in Figure 14. The figures plot the normalized frequency (as a percentage) as a function of the restart latencies. In this figure, we focus on *Starwars* and *Wizard* transmitted to 1 MB-smoothed client buffer with a 10 frame period playback initiation latency, since it is most suitable for low playback bandwidth as well as low playback restart delay.

The benefit of Algorithm 2 is obvious when we compare Figure 14(a) to Figure 14(b). Using Algorithm 2, a user will incur a restart latency less than 0.77 s in 90% of the frame positions. Additionally, in 94% of the frame positions, the restart latency is less than 4 s. On the other hand, the restart latencies under Algorithm 2 are more or less evenly spread over the range 0–14.6 s. Recall that the maximum latency seen under both algorithms are the same.

Similarly for *Wizard*, as depicted in Figure 14(c) and Figure 14(d), for 90% (93%) of the frames positions, the restart latency is less than 0.25 s (1 s) when Algorithm 2 is used. Once again, using Algorithm



(a) *Starwars*: Restart using Algorithm 1

(b) *Starwars*: Restart using Algorithm 2

(c) *Wizard*: Restart using Algorithm 1

(d) *Wizard*: Restart using Algorithm 2

Figure 14: **Comparison between Algorithm 1 and Algorithm 2**

1, results in restart latencies that are roughly evenly spread over 0–4 s.

5 System with multiple video sessions

In the previous sections we have established that Algorithms 1 and 2 (in particular Algorithm 2) permit quick restart of playback service after completion of an interactive function. In the next two sections, we focus on a server that serves multiple clients simultaneously. This allows us to examine the playback-resume latencies seen by clients when Algorithm 2 is used in a multi-session setting. In this section, we outline the model of the server we have used for our study.

We consider a video server that has multiple resources, e.g., I/O subsystem bandwidth, network access bandwidth, system bus bandwidth, memory bandwidth etc. The bottleneck resources among these is the one that limits the throughput of the system. In our system, this determines the maximum number of users

that can be simultaneously accommodated by the server. Effectively, the bandwidth of the server is the bandwidth of this bottleneck resource.

Since playback is the primary functionality desired by clients, we require that the server should provide adequate bandwidth to each client to ensure continuous playback throughout its session.

Inferring from the previous sections that additional bandwidth for restarting playback can reduce the playback restart times, we consider and examine two bandwidth allocation schemes:

- The *Fixed allocation* (FIX) scheme: In this approach, each video session or user has exclusive access only to the bandwidth that is allocated for it. In this scheme, the rate allocated to the session is the peak rate of the smoothed playback transmission schedule of the corresponding video. The advantage of this scheme is the simplicity of its implementation. Also, this scheme can easily be mapped onto the Constant Bit Rate (CBR) network service model for transport of the video data across the network. This is often the default implementation of a video server.

  Under CBR, the network bandwidth is negotiated by an end-host (in our case, the server or the client) at connection setup time, which is fixed for the entire session. Studying this service model allows us to estimate the behavior of the system in a setting where no additional bandwidth is available for interactive operations, thus establishing a lower bound on the performance.

- The second server bandwidth allocation scheme we consider is the *Variable allocation* (VAR) scheme. Under VAR service, to ensure continuous playback, the minimum bandwidth requested by a video session is the peak rate of its smoothed playback transmission schedule. When a user requests playback resumption after an interactive operation, the server allocates the instantaneous excess bandwidth available to it, which is the sum of the available bandwidths from all the sessions that are not transmitting at their peak rate[4]. When the restart operation of multiple sessions overlap, this excess bandwidth is divided equally among all of them. The rationale behind considering this service is to exploit the dramatic restart latency reduction possibilities of Algorithm 2. This model also gives us an indication of the best performance of Algorithm 2 in a multi-user setting, since the bandwidth allocation for restarting playback is maximal. Note that if no additional bandwidth were available under VAR, the performance would be identical to that under FIX.

  In evaluating the performance under VAR, we assume stringent operating conditions. We assume that the server is operating under full load, i.e., it has the maximum number of users that it can admit. Thus our results provide a lower bound on the performance of VAR under other loads.

[4]Note that this excess bandwidth can change while the playback restart operation is in progress. However, the session's own bandwidth is always available to itself for the duration of the operation

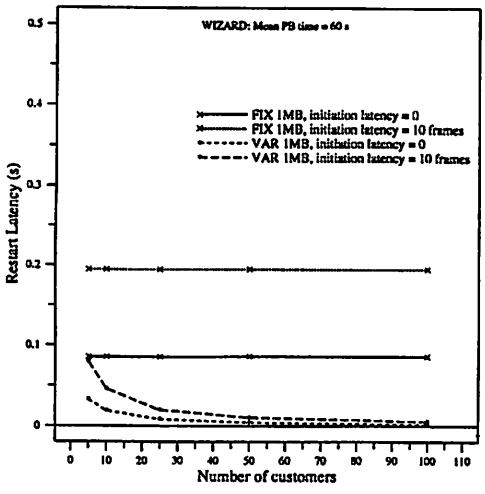


Figure 15: **Avg Latency: Mean PB Time = 60 s**

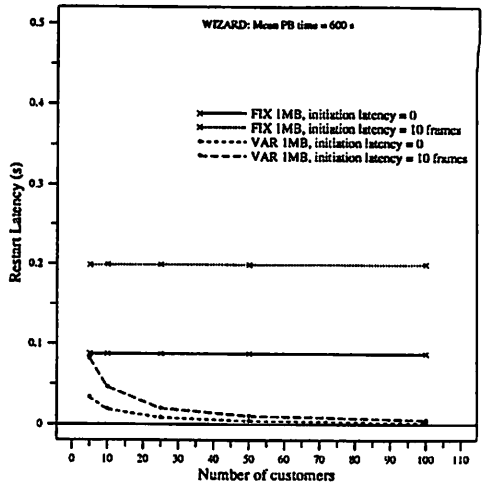


Figure 16: **Avg Latency: Mean PB Time = 600 s**

We note that while the VAR model is attractive in theory, it is difficult to implement in practice. Also, it is difficult to map this model onto a network service model. A possible match for a network service model is the (Available Bit Rate) ABR model under ATM [1, 13, 14, 25]. In that case, the MCR (Minimum Cell Rate) for every stream is the peak bandwidth of the transmission schedule, while the PCR (Peak Cell Rate) is the entire server bandwidth, since it is possible for every stream to have access to the entire server bandwidth during a playback restart operation. However, the ATM network may not be always able to provide the PCR when needed. Also an end-to-end network connection may not have the capacity equal to the excess server bandwidth.

6 Evaluation using Trace driven Simulation

We now proceed to evaluate the performance of Algorithm 2 under the two restart bandwidth allocation schemes in a system with multiple users. We consider a server that has the capacity to support simultaneous playback of at most n independent video sessions. As before, the playback transmission schedule is determined by the optimal smoothing algorithm [22]. A user, in our simulation, spends an exponentially distributed amount of time in playback mode before requesting an index operation. Immediately after the index operation, the user requests playback resumption. The index operation is in the forward or the reverse direction with probability 0.5 each. The new frame position, from where playback resumes, is equally likely to be any frame of the video. Algorithm 2 is used to restart playback. To measure performance under frequent as well as relatively rare interactive activities, the mean time between interactive requests is assumed to be 60 s and 600 s respectively (the 600 s number corresponds to 2 interactive operations on average while watching a 30 minute video segment). Every data point presented is averaged over 15

independent simulation runs, each consisting of 100000 interactive operations. The 95% confidence interval half widths are below 8% for all data presented and discussed in this section.

The evaluation uses the VBR MPEG-1 video traces described in Section 4. In the simulation, the starting frame of each stream is equally likely to be any one of the frames in the corresponding video. This is equivalent to assuming that the phase difference between the starting time of the n sessions is random. The playback restart latency is measured every time playback is resumed. When a video stream reaches its last frame, it is wrapped around to ensure that the number of streams remains identical throughout each simulation.

We describe some representative results in the rest of this section. Given the inferences made in Section 4, we focus primarily on the results for 1 MB sized client buffers.

The first QoS (Quality of Service) metric of interest is the average restart latency. Figures 15 and 16 depict the mean restart latency for 1 MB clients as a function of the number of customers (i.e., the maximum capacity of the server), which is varied from 5 to 100. The playback transmission schedule is smoothed for playback initiation latencies of 0 and 10 frames. Figure 15 corresponds to a mean time between index operations of 60 s while Figure 16 corresponds to a mean time of 600 s. Several aspects of the graphs are notable.

First we note that the mean latencies are very low under both the FIX and VAR service models. When the playback initiation latency is 0, the FIX rate (and the minimum VAR rate) negotiated is 0.071 Mb/frame period. The mean restart latency using this rate is less than 0.1 s. Even when playback transmission has a playback initiation latency of 10 frames, thus requiring a lower peak rate of 0.066 Mb/frame, the mean restart latency remains lower than 0.2 s. Under FIX, since each user/session has exclusive access only to its own resource, the mean restart latency does not change with the number of users. We conclude that using a FIX service model combined with Algorithm 2 results in very tolerable playback restart latencies.

In both Figures 15 and 16, availability of additional bandwidth under the VAR service model results in mean restart latencies that are at most half of that under FIX. In Figure 15, the mean restart latency is only 0.03 s (0.08 s) when the server can accommodate 5 Wizard playback sessions transmitted with 0 (10 frame period) playback initiation latency. Since this is under the condition that the server bandwidth is completely utilized by the 5 sessions, the latencies would be even lower when additional bandwidth is available. For a server that can accommodate a larger number of sessions, statistically the bandwidth available for a playback restart operation is higher. From Figure 13 we know that the marginal decrease of latency reduces as the available bandwidth increase. This explains the shape of the VAR curves in Figures 15 and 16. This also explains the intersection of the two VAR curves for systems of capacity 100 in both figures. Since the peak rate for 0 playback initiation latency is higher than the peak rate for 10 frame period playback initiation latency, the capacity of the first server is higher. Thus statistically more bandwidth is leftover in the first

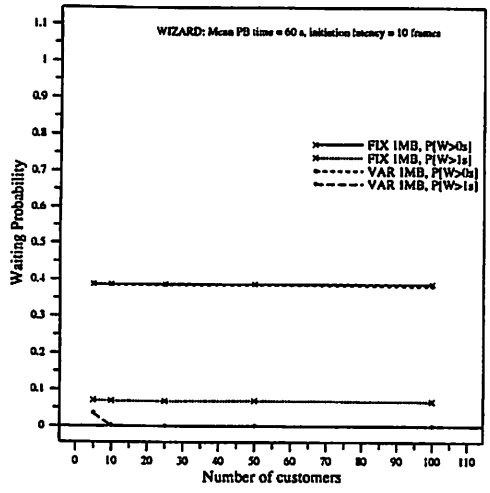


Figure 17: **Tail probabilities: Mean PB Time = 60 s**

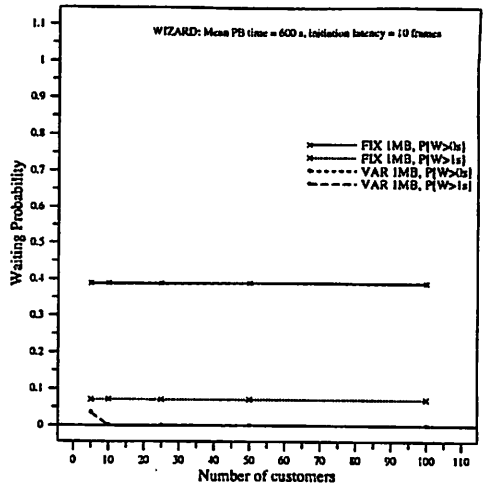


Figure 18: **Tail Probabilities: Mean PB Time = 600 s**

(*Wiz*,*Star*)	$P[W > 0]$	$P[W > 0.25s]$
(63,3)	0.40	0.09
(60,14)	0.41	0.10
(50,50)	0.42	0.10
(40,85)	0.42	0.10
(30,121)	0.41	0.09
(20,157)	0.40	0.09
(10,193)	0.40	0.09
(0,229)	0.41	0.09

Table 2: **FIX: Results for heterogeneous streams. Client buffer sizes = 1 MB, playback initiation latency = 10 frame periods, mean playback time = 60 s. The server capacity is that of an OC-3 link.**

server for restarting playback, leading to more rapid decrease of average restart latency as the server capacity increases from 5 to 100 (i.e. the slope of the curve is higher), causing an eventual intersection.

Finally, we note that the change in average restart latency is relatively insensitive to the frequency of interactive operations. Each of the four corresponding curves in Figures 15 and 16 have very similar values.

We now turn to evaluating the system using a second QoS metric: $P[W > t]$, the probability that the restart latency W exceeds t seconds.

Figures 17 and 18 graph $P[W > t]$ for 1 MB sized clients for mean playback times of 60 s and 600 s respectively. We focus only on playback transmission with playback initiation latency of 10 frame periods.

In Figure 17, we find that $P[W > 0]$ is virtually identical for FIX and VAR. This corroborates our inference that using the traditional FIX server bandwidth allocation model gives very satisfactory performance to a user, and its performance is close to the performance under the more aggressive VAR service model. The value of $P[W > 0]$ is less than 0.4, indicating that a user does not have to wait 60% of

(*Wiz*,*Star*)	$P[W > 0]$	$P[W > 0.25s]$
(63,3)	0.39	0.0
(60,14)	0.40	0.0
(50,50)	0.42	0.0
(40,85)	0.42	2e-05
(30,121)	0.40	6e-05
(20,157)	0.40	0.00045
(10,193)	0.40	0.00085
(0,229)	0.40	0.00233

Table 3: **VAR: Results for heterogeneous streams. Client buffer sizes = 1 MB, playback initiation latency = 10 frame periods, mean playback time = 60 s. The server capacity is that of an OC-3 link.**

the time. We are also encouraged to note that $P[W > 1s]$ is less than 0.1 for FIX, indicating that the restart latency for users would be less than 1 s 90% of the time. The value of $P[W > 1s]$ under VAR starts out at less than 0.05 for a server capacity of 5 customers and decreases sharply initially and then more slowly as the server capacity increases. The rationale for this shape is the same as that for the corresponding $E[W]$ curves. Once again, we note that the results are very similar across Figures 17 and 18, i.e., the results stay the same whether an interactive operation is invoked every minute or every 10 minutes on the average.

Based on our results, we thus conclude that our playback restart scheme works very well under both FIX and VAR server bandwidth allocation models. The restart latencies for playback are very low. Under FIX, no additional bandwidth is necessary either at the server or at the client to provide quick restart. Given the implementation simplicity of FIX, the ubiquity of CBR service availability, and the consequent simplification of admission control procedures in both the network and server, it is the service model of choice.

Finally, we consider the performance of our approach when heterogeneous streams are present in a server. Specifically, we are interested in the sensitivity of the performance to changes in the stream mix. We assume that the server is connected to an OC-3 link, and that the bandwidth of the server is the same as that of the link, i.e., it has a maximum payload capacity of 127.155 Mb/s. The video streams played out of the server are a mix of *Starwars* and *Wizard*. Once again we measure the performance when the server is fully loaded, i.e., when no additional *Wizard* or *Starwars* streams can be played out from it.

Table 2 summarizes the results under FIX for 1 MB sized clients when the playback initiation latency is 10 frame periods. The mean playback time between interactive requests is 60 s. The 2-tuple in the first column of the table is the number of *Wizard* and *Starwars* used to compute the result. The second and the third column are the values of $P[W > 0]$ and $P[W > 0.25s]$. We find that the tail probabilities do change as the stream mixture changes, but the changes are very minor, i.e., the server performance does not vary significantly.

The results under VAR are summarized in Table 3. From this table, we find that the values of $P[W > 0]$

are very similar under both VAR and FIX. However, under VAR, a user has potential access to the entire OC-3 bandwidth when restarting playback, and this results in extremely low values of $P[W > 0.25s]$, i.e., a user rarely waits, if at all, more than 0.25 s to resume playback. Also, under VAR, the results seem to be more sensitive to the stream mix. The OC-3 bandwidth not being an exact multiple of the bandwidth of either *Starwars* or *Wizard*, the actual amount of free bandwidth varies depending on the actual combination of the videos, leading to this difference.

7 Conclusions and Future Work

In this paper, we have considered the problem of restarting playback following an interactive operation in networked multimedia applications such as streaming audio and video. We have proposed two approaches to restart playback for stored video, and developed algorithms to compute the latencies incurred in restarting playback under the two approaches. To transmit video for regular playback, we have used an optimal smoothing algorithm that takes advantage of the available client buffer to reduce the rate variability of compressed video. Using long MPEG-1 traces, we have demonstrated that our approaches result in low latencies in restarting playback.

Although the peak playback bandwidth requirements for optimally smooth transmission decrease, the restart latencies increase, with increasing client buffer sizes. Noting that the marginal decrease in playback bandwidth requirement diminishes with increasing client buffer size, we are able to identify client buffer sizes that achieve the dual goals of low bandwidth requirement for regular playback and low restart latencies after an interactive operation, without using any bandwidth beyond that allocated for playback. In particular, for the traces used in our simulations, client buffer sizes of 1 MB achieved these goals.

Finally, we examined the latencies involved in restarting playback in a video server that supports interactivity, under two policies for sharing its bandwidth: (a) one in which only regular playback bandwidth is available to restart playback, and (b) one in which additional free server bandwidth, if any, is accessible. Under both policies, we find that our approach results in very low playback restart latencies. This result is of particular interest since the first approach can be easily mapped onto the ubiquitous Constant Bit Rate (CBR) network service, with consequent simplification of implementation and admission control procedures in both the network and the server. Although we have established our results using only optimal smoothing for playback transmission, our approaches will work for other transmission schedules, smoothed or non-smoothed.

In our present work we have not made any assumptions about the contents of the client buffer. It may be possible to reduce the client restart latency further if any portion of the client buffer contents can be reused. We plan to explore this in the future.

Acknowledgments

We would like to thank Zhi-Li Zhang for many helpful comments and insightful discussions.

References

[1] ALLES, A. ATM internetworking. available as http://www.cisco.com/warp/public/614/12.html. May 1995.

[2] BUDDHIKOT, M., AND PARULKAR, G. Distributed data layout, scheduling and playout control in a large scale storage server. In *Proceedings of the Sixth International Workshop on Packet Video* (Portland, OR, 1994).

[3] BUDDHIKOT, M., PARULKAR, G., AND COX, J. R. J. Design of a large scale multimedia storage server. *Journal of Computer Networks and ISDN Systems* (1995), 504–517.

[4] CHEN, M.-S., KANDLUR, D., AND YU, P. Support for fully interactive playout in disk-array-based video server. In *Proceedings of ACM International Conference on Multimedia* (San Fransisco, CA, Oct. 1994), ACM, pp. 391–398.

[5] DEY-SIRCAR, J. K., SALEHI, J., KUROSE, J. F., AND TOWSLEY, D. Providing VCR Capabilities in Large-Scale Video Servers. In *Proceedings of ACM International Conference on Multimedia* (San Fransisco, CA, Oct. 1994), ACM, pp. 25–32.

[6] FENG, W.-C., JAHANIAN, F., AND SECHREST, S. Providing VCR functionality in a constant quality video-on-demand transportation service. In *IEEE Multimedia* (Hiroshima, Japan, June 1996), IEEE.

[7] FENG, W.-C., AND REXFORD, J. A comparison of bandwidth smoothing techniques for the transmission of prerecorded compresses video. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)* (Kobe,Japan, Apr. 1997), IEEE.

[8] FENG, W.-C., AND SECHREST, S. Smoothing and buffering for delivery of prerecorded compressed video. In *IS&T/SPIE Multimedia Computing and Networking* (San Jose, CA, Feb. 1995), IS&T/SPIE, pp. 234–242.

[9] GARRETT, M., AND WILLINGER, W. Analysis, modeling and generation of self-similar VBR traffic. In *SIGCOMM Symposium on Communications Architectures and Protocols* (London, UK, Aug. 1994), ACM, pp. 269–280.

[10] GEMMELL, D. J., VIN, H., KANDLUR, D., AND RANGAN, P. V. Multimedia storage servers: A tutorial. *IEEE Computer 28*, 5 (May 1995).

[11] GROSSGLAUSER, M., KESHAV, S., AND TSE, D. RCBR: A simple and efficient service for multiple time-sclae traffic. In *SIGCOMM Symposium on Communications Architectures and Protocols* (Boston, MA, Aug. 1995), ACM, pp. 219–230.

[12] GROSSGLAUSER, M., KESHAV, S., AND TSE, D. The case against variable bit rate services. In *Proc. 5^{th} Workshop on Network and Operating Systems Support for Digital Audio and Video* (Durham, NH, Apr. 1995).

[13] JAIN, R., KALYANARAMAN, S., GOYAL, R., AND FAHMY, S. Source behavior for ATM ABR traffic management: An explanation. *IEEE Communications Magazine* (Nov. 1996).

[14] KALYANARAMAN, S., JAIN, R., GOYAL, R., FAHMY, S., AND JIANG, J. Performance of TCP over ABR on ATM backbone and with various VBR traffic patterns. In *Submitted to ICC 1997* (Montreal, Canada, June 1997), IEEE.

[15] KNIGHTLY, E., WREGE, D., LIEBEHERR, J., AND ZHANG, H. Fundamental limits and tradeoffs of providing deterministic guarantees to VBR video traffic. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems* (Ottawa, Canada, May 1995), ACM, pp. 98–107.

[16] KRUNZ, M., AND HUGHES, H. A traffic model for MPEG-coded VBR streams. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems* (Ottawa, Canada, May 1995), ACM, pp. 47–55.

[17] McManus, J. M., and Ross, K. W. Prerecorded VBR sources in ATM networks: Piecewise-constant-rate transmission and transport. Tech. rep., Manuscript, Sept. 1995.

[18] McManus, J. M., and Ross, K. W. Video-on-demand over ATM: Constant rate transmission and transport. *IEEE Journal on Selected Areas in Communication 14*, 6 (August 1996), 1087–1098.

[19] Reibman, A. R., and Berger, A. On VBR videoconferencing over ATM networks. In *Proc. IEEE GLOBECOMM* (1992), IEEE, pp. 314–319.

[20] Reibman, A. R., and Berger, A. Traffic descriptors for VBR video teleconferencing over ATM networks. *IEEE/ACM Transactions on Networking 3*, 3 (June 1997), 329–339.

[21] Rose, O. Statistical properties of mpeg video traffic and their impact on traffic modeling in atm systems. Tech. Rep. 101, University of Witzburg Institute of Computer Science, Feb 1995.

[22] Salehi, J., Zhang, Z., Kurose, J., and Towsley, D. Supporting stored video: Reducing rate variablity and end-to-end resource requirements through optimal smoothing. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems* (Philadelphia, PA, May 1996), ACM.

[23] Shenoy, P., Goyal, P., and Vin, H. M. Issues in multimedia server design. *ACM Computing Surveys (Special Issue: Symposium on Multimedia Systems* (Dec. 1995).

[24] Shenoy, P., and Vin, H. M. Efficient support for scan operations in video servers. In *ACM Conference on Multimedia* (San Francisco, CA, Nov. 1996), ACM.

[25] Siu, K.-S., and Jain, R. A brief overview of ATM: Protocol layers, LAN emulation, and Traffic Management. In *SIGCOMM Symposium on Communications Architectures and Protocols* (Boston, MA, Apr. 1995), ACM.

[26] WebTV Networks, Inc. WebTV technical specifications. http://www.webtv.net/HTML/home.specs.html, 1996.

[27] Zhang, H., Low, C., Smoliar, S., and Wu, J. H. Video parsing, retrieval and browsing: An integrated and content-based solution. In *Proceedings of ACM International Conference on Multimedia* (San Francisco, CA, Nov. 1995), ACM, pp. 15–23.

[28] Zhang, Z.-L., Kurose, J., Salehi, J., and Towsley, D. Smoothing, statistical multiplexing and call admission control for stored video. *IEEE Journal of Selected Areas in Communications Special Issue on Real-Time Video Services in Multimedia Networks* (To appear).