Periodic Broadcasting with VBR-Encoded Video

Despina Saparilla
Dept. of Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104
saparill@eurecom.fr

Keith W. Ross
Institut EURECOM
2229, route des Crêtes
Sophia Antipolis, France
ross@eurecom.fr

Martin Reisslein
GMD FOKUS
Berlin, Germany
reisslein@fokus.gmd.de

Abstract– **We consider designing near video on demand (VoD) systems that minimize start-up latency while maintaining high image quality. Recently non-uniform segmentation has been used to develop periodic broadcasting techniques for near VoD. These techniques give significant reductions in start-up latency as compared with more conventional uniform segmentation. All of these schemes assume, however, that the videos are CBR-encoded. Since a CBR-encoded video has a larger average rate than an open-loop VBR encoding with the same image quality, there is potential to obtain further performance improvements by using VBR video. In this paper we develop a series of multiplexing schemes for the periodic broadcasting of VBR-encoded video, which are based on smoothing, server buffering and client prefetching. Two key but conflicting performance measures exist when using VBR video: latency and packet loss. By introducing small additional delays in our multiplexing schemes, our traced-based numerical work shows that the schemes can achieve nearly 100% link utilization with negligible packet loss. When the ratio of the CBR rate to the VBR average rate is a modest 1.8, start-up latency can be reduced by a factor of four or more for common scenarios.**

I. INTRODUCTION

True Video on Demand (VoD) services permit subscribers to schedule an arbitrary starting time for a video of their choice. With true VoD, clients can select a video from a large number of video files stored on central video servers. Requested videos are transmitted to a large population of clients through a network (e.g., cable, ADSL, or a LAN), and a distinct stream is dedicated to each user. True VoD is referred to as user centered since server and network bandwidth are strictly divided among the system's users [1] [2]. As the number of users increases, server and network bandwidth is quickly depleted. Consequently, true VoD is often considered inefficient and too costly to offer as a service.

To provide scalable VoD, various techniques based on a data centered approach have been developed, in which the server divides its bandwidth among distinct video objects and each video file is broadcast to the receivers. Broadcasting allows many clients to share a single server stream and, thus, achieves efficient utilization of both network bandwidth and server capacity [3] [4]. Techniques in which many clients share a common server stream provide near VoD. With near VoD, users experience a delay of the order of seconds to tens of minutes before the commencement of the video of their choice. In some near VoD techniques, this start-up latency is due to a delay at the server during which requests for the same object are batched and served together using a single server stream [5]. In another set of near VoD techniques, the server periodically broadcasts each video object at fixed time intervals and clients must wait until the beginning of the broadcast session before viewing the video of their choice. Techniques of this latter type are referred to as periodic broadcasting schemes [1] [6] [7] [2].

When the number of users is large, periodic broadcasting can be an efficient means to distribute stored video. Periodic broadcasting scales nicely, as the start-up latency is completely independent of the number of clients. The start-up latency depends, however, on the particular periodic broadcasting scheme and the number of videos that that are broadcast. In broad terms, the fewer the number of videos, the greater the number of copies of each video that can be broadcast, and the lower the initial start-up latency. Fortunately, for movies on demand, a large fraction of the demand is typically for the 10-20 most popular movies.

One simple periodic broadcasting scheme is to broadcast multiple entire copies of each video, with a new copy broadcast every fixed interval of time (e.g., a new copy of Star Wars broadcast every 20 minutes) [5]. We refer to such schemes as periodic broadcasting with uniform segmentation. In such a scheme, the maximum start-up latency experienced by a user is equal to the length of the video divided by the number of copies broadcast. This latency can be long for full-length MPEG-2 encoded movies sent over channels on the order of 100 Mbps. For example, when ten movies, each encoded at 3 Mbps and each two-hours-long, are broadcast over a 100 Mbps channel, the maximum start-up latency is 40 minutes. Beginning with the seminal paper [2], a number of non-uniform segmentation schemes have recently been proposed [1] [6] [7] [2]. Loosely speaking, these schemes reduce the initial start-up latency by broadcasting the earlier portions of the video more frequently and the latter portions less frequently.

All of the existing work on near VoD systems with periodic broadcasting is based on the assumption that the videos are Constant Bit Rate (CBR) encoded [1] [6] [7] [2]. The CBR encoding technique modifies the quantization scale during compression, which causes quality degradation in the encoded video. (With CBR encoding, the bit-rate of resulting encoded video actually fluctuates around the target CBR rate; but the video can be transmitted at the CBR rate and a small smoothing buffer at the client ensures continuity [8] [9].) For open-loop VBR encoding, the quantization scale remains constant throughout the encoding process, which often produces highly variable bit rates. Digital video distribution systems using satellite and cable have avoided using VBR video due to its burstiness. Nevertheless, for a given movie or sporting event and for the same quality level, the average bit rate for CBR video is typically 2 times or more the average bit rate of VBR video [9] [10]. Therefore with VBR video there is potential for increased system efficiency.

Although non-uniform segmentation can greatly reduce start-up latency, for many practical circumstances the start-up latency remains unacceptably high for CBR-encoded video. When a near VoD system broadcasts full-length MPEG-2 encoded movies over channels on the order of 100 Mbps, the initial latencies can be large. For example, when ten movies, each encoded at 3 Mbps and each two-hours- long, are broadcast over a 100 Mbps channel, the maximum initial start-up latency is more than 17 minutes. In this paper we develop non-uniform segmentation schemes with VBR-encoded video that significantly reduce the initial start-up latency without appreciably degrading image quality. In particular, for situations of practical interest, as the one described above, the start up latency can be reduced by a factor of 4 or more when the CBR/VBR average bit-rate ratio is a modest 1.8.

In order to obtain dramatic reductions in start-up latency with VBR- encoded video, we must allow for some small fraction of packet loss (due to link buffer overflow). The loss, however, will not be noticeable if it is extremely rare. Therefore, the challenge is develop a near VoD scheme that uses VBR-encoded video and yet has low packet loss, on the order of 10^{-6} or less. (Such losses can be effectively hidden by the use of error concealment techniques [11].)

This paper is organized as follows. In Section 2 we show how non-uniform segmentation can be combined with bufferless statistical multiplexing [12] to create a near VoD scheme using VBR-encoded video. We present a methodology that explores the trade-off between start-up latency and packet loss. In Section 3 we study a specific segmentation scheme and present results from trace-driven simulations for bufferless multiplexing. The bufferless multiplexing scheme does not provide sufficiently low loss probabilities, but it does set the stage for a series of more sophisticated schemes described in Section 4, which increasingly offer higher performance. The first of these schemes combines smoothing with bufferless multiplexing, providing significant performance gains. The second scheme uses server-buffer multiplexing, further increasing performance. The third scheme uses client prefetching [13] [14] [15], and leads to yet further improvements. We provide extensive numerical examples that show that the aforementioned schemes can lead to dramatic reductions in initial start-up latency while keeping the loss probability negligible. In Section 5 we show that our VBR multiplexing schemes can dramatically reduce the CBR start-up latency.

II. Near VoD with VBR-Encoded Video

We now present the key components of the general periodic broadcasting technique for VBR-encoded video. Let M be the number of encoded videos to be broadcast and let $N(m)$ be the number of frames in the m^{th} video. All videos are VBR-encoded. In order to keep the presentation simple, we assume that each video has a frame rate of F frames per second. The trace sequence of each prerecorded video is fully known; let $x_n(m), n = 1, \ldots, N(m), m = 1, \ldots, M$ indicate the number of bits in the n^{th} encoded frame of the m^{th} video. Finally, we denote the shared bandwidth between server and clients by C Mbps. All video streams sent by the server share the C Mbps. (The shared channel could be a cable or a digital satellite channel, for example.)

Our basic periodic broadcasting scheme for VBR traffic works as follows. Each video is divided into K segments prior to broadcasting. The server broadcasts MK simultaneous video streams, each of which repeatedly sends a single segment of a video. Frames from the MK streams are statistically multiplexed into the broadcast channel without buffering. Bits are lost whenever the broadcast rate exceeds the channel rate. The server broadcasts each video stream at rate F frames per second, the consumption rate of the videos. A client that wishes to see a particular video tunes to the stream that is repeatedly broadcasting the first segment of that video. The user then waits until the beginning of the segment starts to arrive. We refer to the maximum delay experienced by the user as the start-up latency. At the next broadcast of the first segment the client begins to receive and concurrently display frames from the beginning of the segment. As with the CBR schemes, the client downloads the remaining segments of the video according to a specific download strategy [7] [6]. The choice of download strategy depends on the ability of the client to employ pipelining, i.e., on its ability to receive frames from a number of video streams simultaneously. The download strategy is specified by q, the number of simultaneous streams from which the client can download frames at any time. For example, for $q = 4$, the client downloads segments at their next occurrence for at most four streams at a time.

A central characteristic of a periodic broadcasting scheme (CBR and VBR) is the manner in which the videos are segmented. In general, each video is divided into K segments according to a series of terms referred to as broadcast series [1] [6] [7] [2]. Let $[e_1, e_2, \ldots, e_{K-1}, e_K]$ be a general broadcast series. The series specifies that the first segment of each video consists of e_1 units, the second segment of e_2 units, etc. Without any loss of generality, set $e_1 = 1$. Let $N_i(m)$ indicate the number of frames in the i^{th} segment of the m^{th} video The broadcast series implies that successive segment sizes are related by $N_i(m) = e_i N_1(m)$, $i = 2, \ldots, K$ The size of the first video segment is determined by the equation

$$N_1(m) = \frac{N(m)}{(e_1 + e_2 + \ldots + e_{K-1} + e_K)}. \qquad (1)$$

Thus, a broadcast series $[e_1, e_2, \ldots, e_{K-1}, e_K]$ and video length $N(m)$ completely specifies all segment sizes.

An important requirement of a periodic broadcasting scheme is that it must allow the delivery of the video in a continuous and timely fashion. In other words, the delivery scheme must permit the display of the decoded video at the client without interruptions. This requirement is referred to as the continuity condition. Whether a certain scheme satisfies the continuity condition, given F and $N(m)$, depends on the value of q and the specific form of broadcast series used. For example, consider $q = 1$, i.e., the client can download frames from only one stream at a time. In this case, the uniform broadcast series $[1, 1, \ldots, 1]$ is the only type of series that results in a feasible delivery scheme. When $q = 2$, the series of increasing terms given in [6] and [7] both satisfy the continuity requirement. Finally, in the extreme case when $q = K$, a larger set of broadcast series results in feasible delivery schemes. For example the geometric series $[1, 2, \ldots, 2^{K-1}]$ meets the continuity condition.

As a specific example, consider the continuity condition for $[1, 2, \ldots, 2^{K-1}]$. The broadcasting strategy for this series is illustrated in Figure 1 for $K = 4$ and just a single video. Since $q = K$, the client can download frames from all video streams simultaneously. As a result, each of the four segments can be received at its next broadcast. The following argument shows

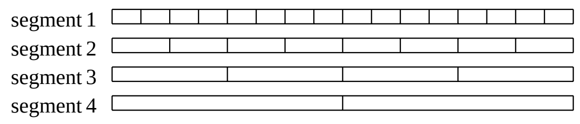

Fig. 1. Broadcasting strategy for geometric series with $e_k = 2^{k-1}$.

that the continuity condition is indeed satisfied for this broadcast series and q combination. Consider two successive video segments of sizes $N_i(m)$ and $N_{i+1}(m)$. The continuity condition is satisfied if the second segment becomes available before or at the time the broadcast of the previous segment ends. Since the sizes of the two segments are related by $N_{i+1} = 2 \cdot N_i(m)$, the broadcasts of the segments either begin or end at the same time. In the case when the broadcasts begin simultaneously, segment $i+1$ becomes available early and can be downloaded and stored by the client in the playback buffer. In the case when the broadcasts end at the same time, a broadcast of segment $i+1$ immediately follows that of segment i, and continuity is maintained.

Start-up latency is defined as the maximum delay experienced by a user before the commencement of a video. This latency is equal to the maximum access time of the first segment of the video, which equals the broadcast duration of the segment. We let $L(m)$ indicate the start-up latency for the m^{th} video. We have $L(m) = \frac{N_1(m)}{F}$. For a general broadcast series $[e_1, e_2, \ldots, e_{K-1}, e_K]$ where $e_1 = 1$, the start-up latency is given by:

$$L(m) = \frac{N(m)}{F \cdot \sum_{i=1}^{K} e_i}. \tag{2}$$

As seen from (2), the start-up latency associated with a periodic broadcasting scheme is decreased for higher values of K. Additionally, for a given value of K, the start-up latency is decreased when a fast growing broadcast series is utilized.

With VBR video, there are two performance measures, start-up latency and loss probability. As we shall see, there is a trade-off between these two measures. We define the probability of loss to be the long-run fraction of bits lost from the video streams during broadcasting. To determine this fraction, we index each video stream by a tuple (m,k), where m indicates the video and k the specific video segment that is sent by the stream. Loss of bits occurs when the aggregate bit rate of the traffic (i.e., from all MK streams) exceeds the link's capacity, C. Let $y_t(m,k)$ denote the number of bits sent by stream (m,k) during frame time t. Then, $y_t(m,k)$ can be expressed as a function of the trace sequence $x_m(n)$ as follows:

$$y_t(m,k) = x_m(j) \tag{3}$$

where j is given by

$$j = \sum_{i=1}^{k-1} N_i(m) + \text{remainder}\left(\frac{t}{N_k(m)}\right). \tag{4}$$

Note that j represents the index for the frame of the m^{th} video (i.e., $j = 1, \ldots, N$) that is sent during frame time t. Observe also that the value of j depends on the resulting segment sizes after division of the video. We next determine y_t, the total number of bits that reach the link during frame time t. Given $y_t(m,k)$ for $m = 1, \ldots, M$, $k = 1, \ldots, K$, y_t is computed as $y_t = \sum_{m=1}^{M} \sum_{k=1}^{K} y_t(m,k)$.

In our bufferless model, bits are lost from the video streams if the aggregate amount of traffic that arrives at the link during frame time t exceeds the link's capacity. Thus, loss occurs in frame time t if $y_t > \frac{C}{F}$.

We express the long-run fraction of traffic lost by P_{loss}. We have:

$$\begin{aligned} P_{\text{loss}} &= \lim_{T\to\infty} \frac{\text{\# of bits lost up to frame time } T}{\text{total \# of bits sent up to frame time } T} \\ &= \lim_{T\to\infty} \frac{\sum_{t=1}^{T} (y_t - C/F)^+}{\sum_{t=1}^{T} y_t}. \end{aligned} \tag{5}$$

Small start-up latency and small loss probability are conflicting objectives. The start-up latency is minimized for high values of K. On the other hand, the aggregate amount of traffic that reaches the link in a frame time increases with K in approximately a linear fashion. Thus, the fraction of bits lost in the long-run also increases with K. In the next section, we present numerical results from the simulation of a specific periodic broadcasting scheme that illustrate the tradeoff between start-up latency and loss probability.

III. Numerical Example: Geometric Series

To illustrate the use of VBR video with periodic broadcasting, we focus our numerical work on one download strategy and broadcast series. The techniques developed in this paper can be applied to an arbitrary download strategy and broadcast series (as long as the continuity condition holds). Specifically, we use $q = K$ and the geometric series $[1, 2, \ldots, 2^{K-1}]$ to segment the videos. As a consequence, the client can download each of the K segments at their next occurrence. Recall from Section 2 that a periodic broadcasting scheme utilizing the above broadcast series and pipelining combination satisfies the continuity condition. In our numerical example, we also make the assumption that receiver storage is not a constraining factor. In other words, we assume that playback buffers at the clients are large enough to receive and store all incoming segments without loss. When receiver storage is a constraint, an approach presented in [6], in which segment sizes are restricted to a maximum value W, can instead be employed.

We obtained 7 MPEG encoded movies from the public domain [16] [17] [18]. The trace of each movie gives the number of bits in each frame. The seven encoded movies were used to create 10 "pseudo traces" each 160,000 frames long. Table 1 summarizes statistics associated with the resulting traces. The 10 traces used in the numerical study were created from the 7 movies in the following manner. The first five traces were created using the encoded movies in [18]. Since each of the original movies is 40,000 frames long, the trace sequence of each movie was repeated four times. Each resulting trace sequence of 160,000 frames was then multiplied by a constant to bring the average bit rate to 2 Mbps. The sixth trace was created by repeating four times the first 40,000 frames of the MPEG encoding obtained from [17] and then manipulated such that its

	Frames		GoPs	
Trace	peak/mean	st. dev (Mbits)	peak/mean	st. dev (Mbits)
bond	10.1	2.11	4.2	1.10
lamps	18.4	3.06	3.3	1.57
mr. bean	13	2.34	5.0	0.48
soccer	6.9	1.91	3.7	1.87
terminator	7.3	1.86	2.8	2.13
wiz. of oz	8.4	2.48	3.2	3.48
star wars 1	10.9	2.45	3.7	2.37
star wars 2	13.2	2.34	4.4	2.57
star wars 3	12	2.31	3.1	2.77
star wars 4	8.5	2.14	3.2	2.96

TABLE I
TRACE STATISTICS

average bit rate is 2 Mbps. Finally, the MPEG encoding obtained from [16] was first divided into four parts 40,000 frames each. The resulting movie segments were repeated four times to create 4 different trace sequences of 160,000 frames. The trace sequences were finally multiplied by constants to create the last four traces illustrated in Table 1 with average bit rates equal to 2 Mbps. Although the ten pseudo traces are not traces of actual movies, we believe that they reflect the characteristics of MPEG-2 encoded movies (highly bursty, long-range scene dependence, average rate about 2 Mbps).

In summary, we have $M = 10$ VBR-encoded videos ($F = 25$ frames/sec) each of which has 160,000 frames, i.e., $N(m) = N = 160,000$ frames, and a length of approximately 107 minutes. As illustrated in Table 1, the traces used in the numerical study have high peak/mean ratios and standard deviations.

Our numerical study of the periodic broadcasting scheme focuses on start-up latency and probability of loss. The start-up latency is computed according to (2) for a general periodic broadcasting scheme. Using the sum of a geometric series with $e_k = 2^{k-1}$ as the sum of the generic broadcast series yields

$$L = \frac{N}{F \cdot (2^K - 1)} \qquad (6)$$

for each video. The probability of loss is expressed by the expected fraction of bits lost from the video streams during broadcasting as shown in (5). In general, to obtain an accurate approximation of P_{loss}, it is necessary to perform a simulation of the periodic broadcasting scheme over a large number of frame times. In our case, however, the use of geometric series $[1, 2, \ldots, 2^{K-1}]$ for segmenting the videos introduces a periodicity in the aggregate traffic pattern y_t. The same aggregate traffic pattern repeats every N_K frame times where N_K is the size of the largest segment of each video. As a result, we can determine P_{loss} by simulating the system for the first N_K frame times. Thus, we compute the loss probability P_{loss} as follows:

$$P_{\text{loss}} = \frac{\sum_{t=1}^{N_K} (y_t - C/F)^+}{\sum_{t=1}^{N_K} y_t}. \qquad (7)$$

To obtain confidence intervals on the loss probability, we perform multiple independent replications of the broadcasting in which the broadcast traffic pattern varies. To allow for replications using the available set of traces, we introduce a random shift in what we consider the starting point of each trace. In each replication of the broadcasting scheme, 10 random numbers are drawn from a uniform distribution between $[1, 160000]$ to determine random starting points for each of the traces. Starting from the random points, a perturbed trace of 160,000 frames is obtained for each of the videos by wrapping each trace around until the original starting point is reached. In each replication, the segmentation of the videos is performed in the manner specified by the broadcast series using new perturbed traces.

A. *Bufferless Statistical Multiplexing*

In this subsection, we study the performance of the periodic broadcasting scheme when the original VBR traffic is statistically multiplexed over a bufferless link. The results illustrate the start- up latency and probability of loss levels achieved by different values of K. We only consider values of K that generate start-up latencies in the range 0-16 minutes and loss probabilities below 0.1. The start-up latencies resulting from different K values are independent of the link's capacity. Recall that when the geometric series is used to segment the videos, start-up latency decreases exponentially with K. It is easily seen from (6) that for start-up latencies below 2 minutes, K must be at least 6. Start-up latencies below half a minute result from values of K that are at least 9. As the number of segments K increases, the probability that bits will be lost from the video streams also becomes higher. Our results specify this tradeoff between start-up latency and the probability of loss.

In Figure 2, the probability of loss, P_{loss}, is plotted against start-up latency for three levels of link capacity. The figure shows the average loss probability and 90% confidence intervals for the average, whose length is smaller than 10% of the estimated mean. As the figure illustrates, the loss probability associated with each start-up latency value, or equivalently with each value of K, varies according to link capacity. Each point on the curve for a single bandwidth level corresponds to a specific value of K. Depending on the available link capacity, feasible K values range between 3 and 10. Lower K values result in start-up latencies that exceed 16 minutes while higher values generate loss probabilities above 0.1. Figure 2 illustrates the tradeoff between P_{loss} and start-up latency with varying K. For start-up latency values below 2 minutes, the probability of loss is high (i.e., in the order of 10^{-4} or higher) for all the levels of link bandwidth examined. While this loss-latency tradeoff is a key characteristic of periodic broadcasting with VBR-encoded video, it is not an issue for broadcasting CBR-encoded video. The constant rate traffic in the CBR schemes allows the channel's bandwidth to be allocated among the videos in a manner that guarantees no loss due to channel overflow.

The results obtained for bufferless statistical multiplexing of the VBR-encoded streams indicate that if it is desirable to achieve latency values below 2 minutes, high probabilities of loss will be incurred resulting in unacceptable levels of quality degradation in the decoded video. This has motivated us to refine our multiplexing schemes to improve performance. In the following sections, we examine three different methods for limiting loss. The first is GoP smoothing on the VBR traces prior to broadcasting. The second is buffered statistical multiplexing by the addition of a finite buffer at the server link. The third uses prefetching of video frames during periods of time when the shared link's bandwidth is under utilized.

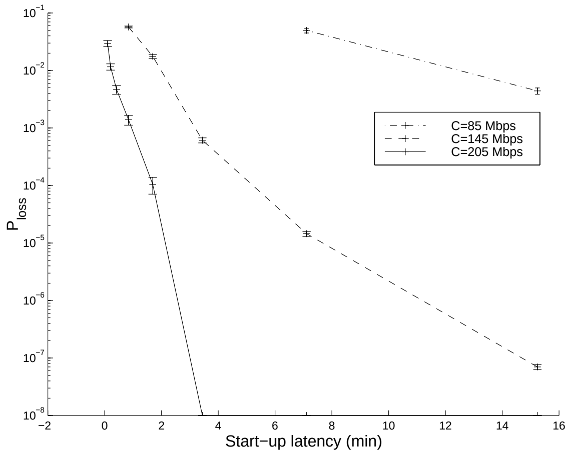

Fig. 2. Bufferless statistical multiplexing.

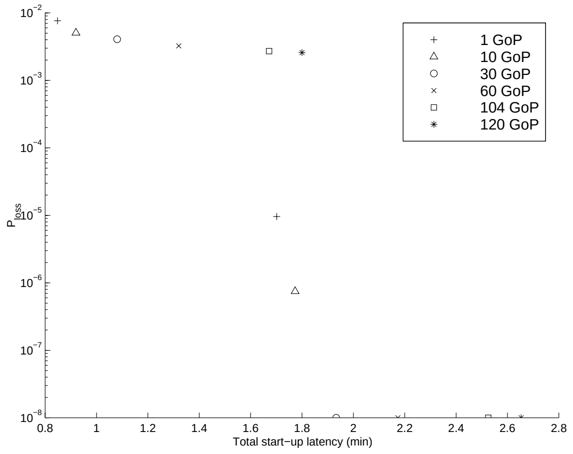

Fig. 4. Smoothing over many GoP periods ($C = 145$ Mbps).

IV. Improving Performance

A. GoP Smoothing

We investigate the effect of GoP smoothing on the performance of the broadcasting scheme. We first obtain results for the case when the video traces are smoothed over each GoP period for link capacities ranging from 85 to 205 Mbps. The results are plotted in Figure 3. The total start-up latency shown in the plot is the sum of the maximum access time for the first video segment and the delay introduced due to smoothing over one GoP period. This additional delay is equal to the length of a GoP period. For example, when the GoP size is equal to 12 frames and the broadcast rate is 25 frames per second, the additional start-up delay introduced due to smoothing equals 0.48 seconds.

We observe a significant improvement in the loss probability due to GoP smoothing for all three link capacities studied. Note that improvement in P_{loss} occurs at the expense of only a small increase in the total playback delay, i.e., an additional delay of 0.48 seconds.

We now focus on the case when $C = 145$ Mbps. Our numerical study aims at examining the effect of further smoothing on the loss probability. Instead of smoothing the video traces over each GoP period, we employ smoothing of each trace over intervals that consist of a larger number of GoP periods. The results are shown in Figure 4. We concentrate on values of K equal to 6 and 7, since for smaller K P_{loss} is zero for all smoothing policies. Let us first consider the cluster of points in the 1.5-2.8 minute interval on the latency scale. These points correspond to different smoothing policies for $K = 6$. Clearly, smoothing over intervals of 10 or 30 GoP periods, results in a considerable decrease in P_{loss}. Smoothing over intervals of 60 to 120 GoP periods introduces a longer smoothing delay without affecting P_{loss}. Thus, when $K = 6$, smoothing over intervals longer than 30 GoP periods is not only unnecessary but also undesirable. We refer to points that correspond to longer total start-up latencies with no further improvement in P_{loss} as *dominated*. Now consider the leftmost cluster of points in the 0.5-2 minute latency interval for which $K = 7$. These points are non-dominated in the sense there is always an improvement in P_{loss} with increasing latency. We observe however that the decrease in P_{loss} achieved by smoothing over longer periods is not significant relative to the added delay introduced by smoothing.

Finally, observe that smoothing over very long intervals (i.e., intervals of 120 GoP periods), results in additional delays which are significant enough to also cause dominance between clusters of points that correspond to different K values. In particular, division of the video files into 7 segments when smoothing over intervals of 120 GoP periods is implemented, results in higher latency than division into 6 segments with smoothing over 1 GoP period. In conclusion, smoothing over a higher number of GoP periods does not have an adverse effect when low start-up latencies are desirable.

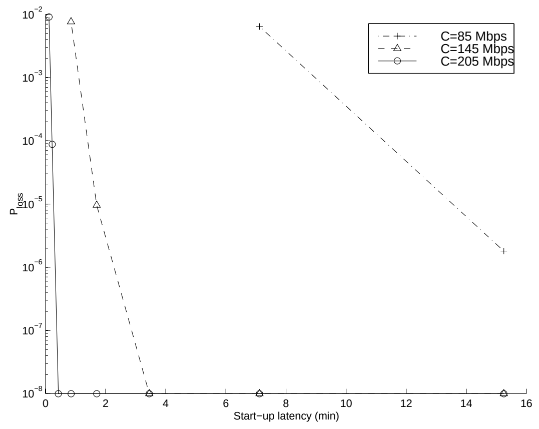

Fig. 3. Bufferless multiplexing with smoothing over each GoP period.

B. Buffered Statistical Multiplexing

In this section we consider buffered statistical multiplexing (with no smoothing) of the video streams by introducing a finite buffer of size B at the server link. We vary B in the range 72 to 8700 Mbits, corresponding to an additional start-up delay of 0.5 to 60 sec. (The added delay is equal to B/C seconds, which is the maximum possible delay that can be introduced due to buffering.) The results are shown in Figure 5 for $C = 145$ Mbps. The total start-up latency in the buffered case is the sum of the maximum access time for receiving the first video segment plus an added delay due to the buffer. To facilitate comparison of results, we include the case of bufferless statistical multiplexing.

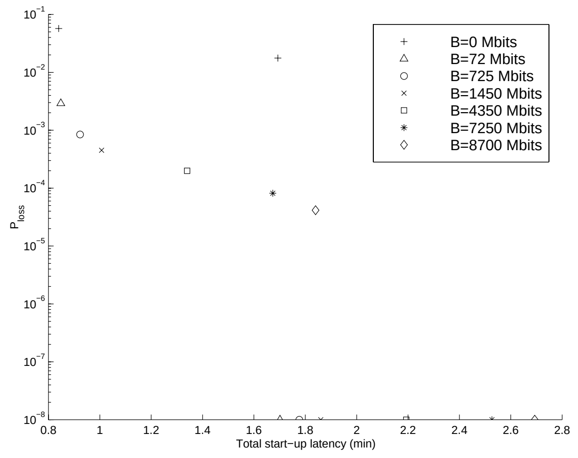

Fig. 5. Buffered statistical mutliplexing ($C = 145$ Mbps).

Figure 5 shows results for $K = 6, 7$. We observe that a buffer of size 72 Mbits has a significant positive effect on the loss probability in comparison to the case of bufferless statistical multiplexing with no smoothing. For instance, a start-up latency of approximately 1.7 minutes ($K = 6$) can be achieved with P_{loss} equal to zero. Bufferless statistical multiplexing, on the other hand, results in P_{loss} in the order of 10^{-2}. Increasing the size of the buffer to values higher than 72 Mbits, when $K = 6$, results in dominated points. When $K = 7$, increasing the size of the buffer achieves consistent improvement in P_{loss}. Note however, that the improvement in P_{loss} becomes less significant as the buffer sizes increase. The most dramatic improvement occurs for an increase of the buffer size from 0 to 72 Mbits. Increasing the buffer size from 4350 to 8700 Mbits, however, results in a significant added delay of 0.5 minutes for a more moderate improvement in P_{loss}. Using a buffer of 8700 Mbits when $K = 7$ generates a dominated point. Thus, in this example, utilizing buffers that introduce (maximum) delays longer than 30 seconds is not desirable. To limit loss it is instead preferable to use a smaller K.

C. Join-the-Shortest Queue Prefetching

The Join-the-Shortest-Queue prefetching protocol was originally developed for the client centered streaming of VBR-encoded video over a shared bufferless link [13]. Before we discuss how the JSQ protocol can be applied to the data centered near VoD systems studied in this paper, we briefly discuss its underlying idea. The JSQ protocol is based on the observation that due to the VBR nature of the multiplexed video streams there are frequent periods of time during which the shared links' bandwidth is under utilized. During these periods the server can prefetch video frames from any of the ongoing video streams and send the prefetched frames to the buffers in the appropriate clients. As a result, many of the clients will typically have some prefetched reserve in their buffers. The JSQ protocol also specifies the policy for selecting the prefetched frames. According to the JSQ policy, within each frame period the server repeatedly selects frames from the connections that have the smallest number of prefetched frames in their client buffers. Empirical work with MPEG-1 traces in [13] indicates that prefetching combined with the JSQ policy gives dramatic reductions in loss. In particular, if each client dedicates a small buffer to the video streaming application, JSQ prefetching allows for almost 100% utilization on the shared link with negligible loss. For a detailed discussion of the JSQ protocol for client centered video streaming we refer the reader to [13].

We proceed to explain how to apply the JSQ protocol to the data centered near VoD system. We introduce the concept of virtual buffers. We assign to each stream (m, k) a virtual buffer which tracks the buffer contents of a client that is tuned in to stream (m, k) at all times and repeatetly displays segment (m, k). Note that the system of MK virtual buffers receiving MK distinct video streams constitutes a client centered video streaming system. The JSQ protocol of [13] can therefore readily be applied to the system of virtual buffers. We first describe in detail the protocols' operation in the system of virtual buffers. We then explain how the system of virtual buffers relates to the actual near VoD system.

In explaining the details of the JSQ protocol we divide time into slots of length $1/F$. Let $p_t(m, k)$ denote the number of prefetched frames in virtual buffer (m, k) at the beginning of slot t. Let $\Delta_t(m, k)$ denote the number of frames that arrive to virtual buffer (m, k) during slot t. At the end of each slot one frame is removed and displayed, provided the virtual buffer holds one or more frames. Thus

$$p_{t+1}(m, k) = [p_t(m, k) + \Delta_t(m, k) - 1]^+. \qquad (8)$$

If the frame scheduled to be displayed at the end of the slot does not arrive in time, the virtual buffer is starved and the frame is considered lost. The server skips the transmission of a frame that will not meet its deadline at the virtual buffer. For each of the MK virtual buffers the server keeps track of the buffer contents $p_t(m, k)$ through (8).

During each slot of length $1/F$ seconds the server decides which frames to transmit from the MK ongoing streams. This is done according to the JSQ prefetch policy. The maximum number of bits that can be transmitted in a slot is C/F. The JSQ prefetch policy attempts to balance the number of prefetched frames across all virtual buffers. In describing the policy we drop the subscript t. Let z be a variable that keeps track of the total number of bits sent within a slot. At the beginning of the each slot the server determines the stream (m^*, k^*) with the smallest $p(m, k)$ and checks whether

$$z + x_{\sigma(m^*,k^*)}(m^*) \leq C/F, \qquad (9)$$

where $\sigma(m^*, k^*)$ is the frame of video m^* considered for transmission. If (9) holds, we transmit the frame, increment $p(m^*, k^*)$ and update z. If (9) is violated, we remove the stream (m^*, k^*) from consideration and find a new stream (m^*, k^*) that minimizes $p(m, k)$. If (9) holds for the frame of the new stream (m^*, k^*), we transmit the frame and update $p(m^*, k^*)$ and z. We then continue the procedure of transmitting frames from the streams that minimize the $p(m, k)$'s. Whenever a frame violates (9) we skip the corresponding stream and find a new stream (m^*, k^*). Once all streams have been skipped, we set $p(m, k) = [p(m, k) - 1]^+$ for all $m = 1, \ldots, M$ and $k = 1, \ldots, K$ and move on to the next slot.

To employ the JSQ protocol in the near VoD system, the server needs to schedule the broadcast of the frames of the MK

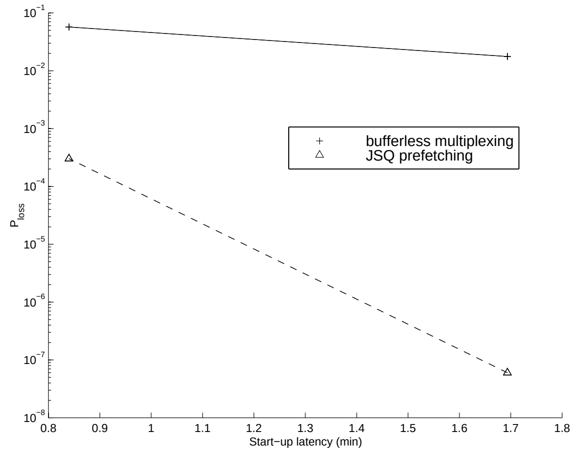

Fig. 6. Comparison of JSQ prefetching and bufferless statistical mutliplexing.

video streams as if they were being sent to the MK distinct virtual buffers. The clients tune in to a video and store the frames of the currently displayed segment that arrive early while retrieving and displaying each frame as its deadline arrives. Note that prefetching does not interfere with the continuity condition.

How do the virtual buffers relate to the buffers in the clients of the near VoD system? When the geometric broadcast series $e_k = 2^{k-1}$ is used, the lengths of any two successive segments (m, k) and $(m, k+1)$ satisfy $N_{k+1}(m) = 2 \cdot N_k(m)$. Hence, each two segments either begin or end at the same time. First, consider the case when the two segments end at the same time, i.e., the ends of segments (m, k) and $(m, k+1)$ coincide. In this case the client starts to receive and display the next broadcast of segment $(m, k+1)$ immediately after segment (m, k) has ended. During the broadcast of segment $(m, k+1)$, the buffer contents of the client in the near VoD system and the buffer contents of the virtual buffer in the virtual buffer system are exactly the same. Hence, whenever loss occurs in the virtual buffer system, the near VoD system suffers exactly the same loss. In the case when the segments begin at the same time, the client receives and displays segment (m, k) while segment $(m, k+1)$ is being received and stored. When segment (m, k) ends, the beginning of segment $(m, k+1)$ is displayed. In other words, the display of segment $(m, k+1)$ is delayed by $N_k(m)$ frame periods by the client. This implies that whenever frame starvation occurs at the virtual buffer, loss is detected at the client $N_k(m)$ frame periods later. The long-run loss probability is therefore the same for both systems.

In Figure 6 we plot the results of a simulation study of the JSQ protocol with $C = 145$ Mbps. We compare the JSQ results with the results obtained for bufferless statistical multiplexing in Section 3-A. Note that the JSQ protocol runs over a bufferless link. We observe that the JSQ protocol brings significant improvement over simply multiplexing the video streams onto the bufferless link. For $K = 7$ (i.e., a start-up latency of 50.4 seconds) the loss probability drops from roughly $6 \cdot 10^{-2}$ to approximately $3 \cdot 10^{-4}$ with JSQ prefetching.

We next develop a refinement of the JSQ protocol, which allows the virtual buffers and clients in the near VoD system to build up a reserve of frames over a certain period of time. We refer to the length of the period of time during which frames are prefetched but not consumed as the prefetch delay, denoted by d_{pre} frame periods. We refer to the refined JSQ protocol as JSQ prefetching with prefetch delay. The total start-up latency in this case is $L = (N_1 + d_{\text{pre}})/F$.

JSQ prefetching with prefetch delay of segments generated according to the geometric broadcast series $e_k = 2^{k-1}$ satisfies the continuity condition. Consider any two successive segments (m, k) and $(m, k+1)$. Recall that the segments either begin or end at the same time. In the case when the segments begin at the same time, prefetching for the segments also starts at the same time, i.e., (d_{pre} frame periods before the display of segment (m, k) starts). The start of segment $(m, k+1)$ is hence already stored in the client buffer when the display of segment (m, k) ends. When the segments (m, k) and $(m, k+1)$ end at the same time, the server starts to prefetch frames for the next broadcast of segment $(m, k+1)$ d_{pre} frame periods before the current broadcast of that segment ends. This ensures that the start of segment $(m, k+1)$ is available when segment (m, k) ends.

Figure 7 shows the results of a simulation of JSQ prefetching with prefetch delay for $C = 145$ Mbps and $K = 7$. The introduction of the prefetch delay improves the loss probability significantly. For a prefetch delay of 10 sec, the loss probability decreases from $3 \cdot 10^{-4}$ to $9 \cdot 10^{-5}$. Increasing the prefetch delay to 50 sec results in a total start-up latency of 100.4 sec and a lower loss probability of $6.6 \cdot 10^{-6}$. Observe from Figure 6, however, that JSQ prefetching without prefetch delay results in a total start-up latency of 100.7 sec and a loss probability equal to $6 \cdot 10^{-8}$ when $K = 6$. This indicates that to achieve a low loss probability, it is preferable to use a smaller K rather that a long prefetch delay. This observation parallels the conclusion of Section 4-B on buffered multiplexing which indicated that smaller K gives better performance than the use of very large buffers.

We now compare the performance of buffered multiplexing, GoP smoothing and JSQ prefetching in terms of their effectiveness in limiting the loss probability. We generate the dominance curves corresponding to Figures 4, 5 and 7. The dominance curves specify the non-dominated points generated by each technique for different levels of the key parameters (i.e., different buffer sizes, smoothing intervals and prefetching delays). The results, illustrated in Figure 8, indicate that for simi-

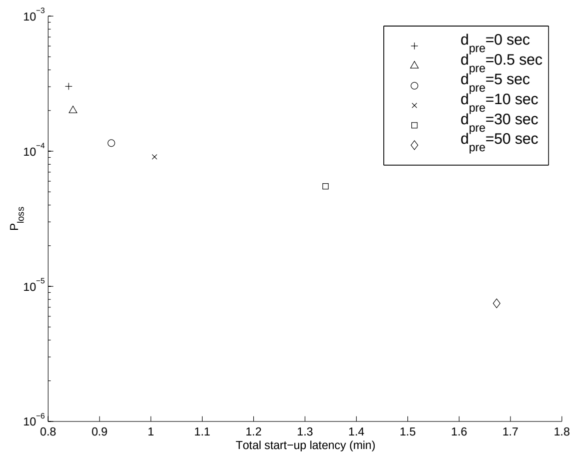

Fig. 7. JSQ prefteching with pretech delay.

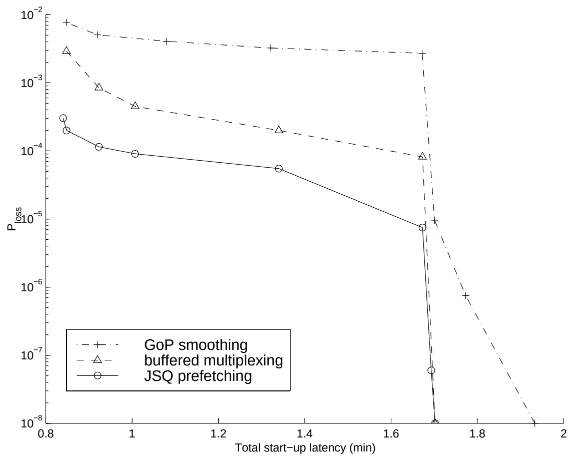

Fig. 8. Dominance curves for GoP smoothing, buffered multiplexing and JSQ prefetching.

lar latencies, JSQ prefetching gives the lowest loss probabilities. We observe that the loss probabilities associated with buffered multiplexing can be an order of magnitude higher than the ones generated by JSQ prefetching. We note, however, that buffered multiplexing attains the performance of JSQ prefetching for extremely low levels of loss in this example. Finally, as it is clearly seen in Figure 8, JSQ prefetching and buffered multiplexing is more effective in terms of limiting the loss probability than GoP smoothing.

V. VBR AND CBR COMPARED

Having shown how to design high-performance periodic broadcasting schemes for VBR-encoded video, we now compare the latency performance of CBR and VBR encoded video. In order to make a true comparison, we need to know how much CBR bandwidth is needed to achieve the image quality of open-loop VBR encoding. Unfortunately, we do not have this information for the traces in this paper. However, recent studies have shown that for movies and sporting events, the ratio of the average rate for CBR encoding to the average rate for VBR encoding is in the 2.0 range if not greater [9] [10]. Therefore, to compare VBR with CBR we will make the conservative assumption that the ratio is 1.8, i.e., CBR has an average rate 80% higher than VBR encoding for each of the traces. Since each of our VBR traces has an average bit rate of 2 Mbps, each of the CBR videos has a bit rate of 3.6 Mbps. With a known CBR rate and channel rate, it is easy to determine the start-up latency for CBR for the case $q = K$ and a geometric broadcast series [7].

The CBR start-up latencies are given in Table 2 for three link capacities. In Table 2 we also present the start-up latencies for buffered multiplexing of VBR-encoded video. (We use buffered multiplexing instead of JSQ prefetching because it requires less time for simulation; JSQ prefetching can give even better performance.) For the buffered multiplexing, we chose the K value and buffer size combination which gives the lowest delay while having a loss probability less than 10^{-7} (essentially a negligible loss probability). We see that for each of the link capacities, our VBR multiplexing scheme has reduced the start-up latency by more than a factor of 4.

The dramatic reduction in start-up latency is primarily due to

C(Mbps)	Latency of CBR	Latency of VBR
85	35.6	7.3
145	7.1	1.7
205	3.4	0.2

TABLE II
LATENCY IN MINUTES OF CBR AND VBR VIDEO

the fact that the latency decreases exponentially fast with K, the number of segments in the broadcast series. The lower average rate of VBR allows us to increase K, and thereby obtain significant reductions in start-up latency.

ACKNOWLEDGMENTS

This work has been supported partially by NSF grant NCR97-06495. Eurecom's research is partially supported by its industrial partners: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

REFERENCES

[1] C. C. Aggarwal J. L. Wolf P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," in *Proc. of the IEEE Int'l Conf. on Multimedia Systems*, Hiroshima, Japan, June 1996.

[2] S. Viswanathan T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Systems*, vol. 4, no. 4, pp. 197–208, August 1996.

[3] K. Almeroth M. H. Ammar, "The use of multicast delivery to provide a scalable and intercative video-on-demand service," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 6, pp. 1110–1122, 1996.

[4] W. D. Sincoski, "System architecture for a large scale video on demand service," *Computer Networks and ISDN systems*, vol. 22, 1991.

[5] D. Sitaram P. Shahabuddin A. Dan, "Scheduling policies for an on-demand video server with batching," in *Proc. of ACM Multimedia*, San Francisco, California, October 1994, pp. 15–23.

[6] K. A. Hua S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on demand systems," in *Proc. of the ACM SIGCOMM*, Cannes, France, September 1997.

[7] K. A. Hua Y. Cai S. Sheu, "A Client-Centric Approach to designing periodic broadcast schemes," Tech. Rep. CS-TR-98-02, School of Computer Science, University of Central Florida, Orlando, Florida, January 1998.

[8] T. V. Lakshman A. Ortega A. R. Reibman, "VBR Video: Trade-offs and potentials," in *Proceedings of the IEEE*, May 1998, vol. 86, pp. 952–973.

[9] I. Dalgic F. A. Tobagi, "Characterization of quality and traffic for various video encoding schemes and various encoder control schemes," Tech. Rep. CSL-TR-96-701, Departments of Electrical Engineering and Computer Science, Stanford University, August 1996.

[10] W. S. Tan N. Duong J. Princen, "A comparison study of variable bit rate versus fixed bit rate video transmission," in *Australian Broadband Switching and Services Symposium*, 1991, pp. 134–141.

[11] W. Luo M. El Zarki, "Analysis of error concealment schemes for MPEG-2 video transmission over ATM based networks," in *Proceedings of SPIE Visual Communications and Image Processing*, Taiwan, May 1995.

[12] M. Reisslein K. W. Ross, "Call Admission for prerecorded sources with packet loss," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, pp. 1167–1180, August 1997.

[13] M. Reisslein K. W. Ross, "A join-the-shortest-queue prefetching protocol for VBR video on demand," in *IEEE International Conference on Network Protocols*, Atlanta, GA, October 1997.

[14] M. Reisslein K. W. Ross V. Verillotte, "A decentralized prefetching protocol for VBR video on demand," in *Multimedia Applications, Services and Techniques-ECMAST (Lecture Notes in Computer Science)*, D. Hutchison R. Schafer, Ed., vol. 1425, pp. 388–401. Springer Verlag, Berlin, Germany, May 1998.

[15] M. Reisslein K. W. Ross, "High-Performance Prefetching Protocols for VBR Prerecorded Videoxb," *IEEE Network*, vol. 12, no. 6, Nov/Dec 1998.

[16] M. W. Garret A. Fernandez, "Variable bit rate video bandwidth trace using MPEG code," Nov 1994.

[17] M. Krunz R. Sass H. Hughes, "Statistical characteristics and multiplexing of MPEG streams," in *Proceedings of the IEEE INFOCOM*, April 1995, pp. 455–462.

[18] O.Rose, "Statistical properties of MPEG video traffic and their impact on traffic modelling in ATM systems," Tech. Rep. 101, University of Wuerzburg, Institute of Computer Science, Am Hubland, 97074 Wuerzburg, Germany, February 1995, ftp://ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG/.