University of Alberta

An Inherently Loss–Less and Bandwidth–Efficient Periodic Broadcast Scheme for VBR Video

by

Fulu Li and Ioanis Nikolaidis

Technical Report TR 99-04
November 1999
(Revised June 2000)

DEPARTMENT OF COMPUTING SCIENCE
University of Alberta
Edmonton, Alberta, Canada

An Inherently Loss–Less and Bandwidth–Efficient Periodic Broadcast Scheme for VBR Video

Fulu Li & Ioanis Nikolaidis

November 1999
(Revised June 2000)

Abstract

A key requirement for the successful deployment of Video-on-Demand (VoD) systems is to maintain both bandwidth efficiency as well as tolerable startup latency. The startup latency is defined as the interval between the point a user "tunes–in" to (starts downloading) the video content and the point at which the uninterrupted playout of the video can start. Periodic broadcast schemes have been proposed to solve the VoD distribution problem for the set of the most requested videos. However, the current proposals for periodic broadcast schemes assume CBR encoded videos. The few existing proposals for support of the bandwidth–efficient family of VBR encoded videos are available at the cost of data loss due to overflow of the broadcast link capacity by the aggregate traffic of the simultaneously transmitted segments. In this paper we address the issue of the VBR encoded video broadcast and we propose LLBE (a *L*oss–*L*ess *B*andwidth–*E*fficient scheme), which, in contrast to previous techniques for VBR video, is an inherently loss–less broadcast distribution scheme. Given ample client storage, LLBE can satisfy any a–priori prescribed per–video startup latency. In LLBE, the bandwidth necessary for the transmission and downloading of a video is minimized through an optimization process. We observe that the minimization of the server bandwidth can be stated as a shortest path problem. Finally, example performance results of LLBE and a discussion of some issues related to its implementation are presented.

Extended version of the paper appearing in the Proceedings of ACM SIGMETRICS 2000.

1 Introduction

In this paper, we present a novel Loss–Less and Bandwidth–Efficient (LLBE) protocol requiring minimal server bandwidth while maintaining zero data loss for the periodic broadcast of VBR video in Video–on–Demand (VoD) systems. LLBE is, to the best of our knowledge, the first proposal that supports, by its design, the loss–less periodic broadcast of VBR encoded video. LLBE can achieve any arbitrary a–priori per–video startup latency (i.e., delay from "tune–in" of the set–top box to start of uninterrupted video playout). The startup latency is deterministic, that is, *exactly* w_m seconds of startup latency is required for the m–th video. In this sense, LLBE provides predictable latency for the client devices, which can be considered preferable to the bounded but random startup latency featured by other schemes.

LLBE operates by partitioning the entire video into smaller successive and non–overlapping segments. Each segment is continuously broadcast on a separate channel. Each channel is assigned different bandwidth. The client set–top box starts downloading the information of all the segments of a specific video as soon as the client "tunes–in". As soon as the first segment is downloaded completely, the uninterrupted playout of the video can begin. Because of the schedule construction employed in LLBE, subsequent segments are guaranteed to be completely downloaded just prior to their consumption by the playout process.

This paper differs from previous work in two crucial assumptions:

1. *The client–side set–top box secondary storage capacity is not considered a constraint.* Recent price reductions for secondary storage devices, such as magnetic disks, as well as the gradual availability of re–writable DVD media, indicates that commodity priced set–top boxes will likely contain storage in the range of several Gigabytes. We thus believe that in the near future any set–top box with a basic set of capabilities will be sufficiently equipped for the off–line storage of complete feature–length (approximately 2 hours) videos. Subsequently, we will assume that the client set–top box is capable to store a large percentage, up to 100%, of a video.

2. *Reception of a broadcast segment does not delay until the start of the segment transmission, but can start as soon as the client set–top box "tunes–in".* Previous schemes insist on forcing the reception of a segment to start at the exact beginning of a segment, and never *during* its transmission. The segment lengths are possibly representing several Gigabytes of information. Hence, it makes little sense to force the start–of–segment restriction, especially if the following factors are accounted for:

   (a) As a matter of design, VoD systems are to predominantly operate over uni–directional broadcast or multicast channels, over cable or satellite infrastructure. Forward Error Correction (FEC) is routinely employed in such systems for increased open–loop data integrity. FEC schemes group information in units (blocks) such that error propagation is limited and re-synchronization of sender/receiver is still possible in the presence of errors.

   (b) The information conveyed in MPEG-1 and other VBR video encodings observes its own frame–oriented or block–oriented structure. Parsing an incoming stream of MPEG-1 data allows the receiver to synchronize by identifying the boundaries between successive frames.

   (c) A possible multiplexing technology used for transporting the several different video segments is packet–(cell–)based statistical multiplexing. Thus the entire segments are split into packets and the inclusion of timing information in the header of the packets is subsequently trivial to accomplish.

   The above three factors indicate that it is, in principle, possible to start reception of a segment without waiting for its beginning, since the fragmentation of the entire segment into smaller units is a matter of error resilience, or of the nature of video traffic, or of the underlying multiplexing technique. For the sake of brevity we will assume that the reception starts at frame boundaries. For example, a segment consisting

of 10000 frames, can be completely received if 10000 consecutive frames are received from the channel on which the corresponding segment is cyclically broadcast. The time interval between the tune–in and the identification of the first frame of a segment is considered negligible compared to w_m. Note that reasonable values of w_m for near–VoD systems range typically in the range of few seconds up to a few minutes.

In this paper we do not consider VCR–like functionality, such as fast forward/backward and pause. We also consider that the underlying network can provide dedicated broadcast/multicast service without the interference from other traffic flows in the network. Thus, the presented solution is not suitable for best–effort type of service, in the sense that it does not provide the means to compensate for the jitter accrued in a best–effort packet–switched network. Nevertheless, the scheme can be applied in networks supporting bandwidth reservation.

The remaining of this paper is organized as follows: Section 2 provides a review of the related research. Section 3 provides the necessary notation and definitions used in the new proposed scheme. Section 4 outlines the periodic schedule construction process and the related algorithm. Section 5 illustrates through experimental results the properties of the new scheme. A summary and future research directions is given in Section 6.

2 Previous Work

Periodic broadcast schemes form one of the two families of options used in the distribution of VoD content. The other family is batched multicast, e.g., [1, 3]. Batched multicast collects requests over successive non–overlapping intervals of time and admits a single new multicast for each video for which it received one or more requests in the last collection interval. Blocking, i.e., rejection at the stage of call admission is possible. Although batched multicasting is appealing for its ability to satisfy multiple requests through a single multicast, it is inefficient for extremely popular ("hot–set") videos. A simple back–of–the envelope computation indicates that for 5 minute request collection intervals, and a typical popular feature movie length of 2 hours, 24 replicas of the same video will be in progress at any point in time. At the same time it is reasonable to assume that the most popular items will be the ones for which a provider wishes to minimize the startup latency. Clearly, reducing the collection interval (and hence startup latency) results in an inversely proportional increase of the concurrently transmitting replicas of the same video. Thus, batching for the hot–set videos is inefficient in terms of bandwidth.

While not dismissing the value of batched multicast for "cold–set" videos, a more efficient alternative is needed for the "hot–set". The rest of this paper deals precisely with the class of "hot–set" videos. The introduction of the periodic broadcast schemes was initially proposed in the form of staggered broadcast, e.g., by Dan et. al. [5]. Multiple entire copies of each video are repeated every fixed interval of time. The startup latency of this approach is normally large since the maximum startup latency experienced by a user is equal to the length of the video divided by the number of copies that can be broadcast using a common link capacity. For example, ten two-hour long movies, encoded at 3 Mbits/sec, broadcast on a link of 155 Mbits/sec, result in latency of 24 minutes.

The Pyramid Broadcasting (PB) technique presented by Viswanathan and Imielinski in [15] was the first to drastically reduce the service latency by dividing each video into simultaneously broadcasting segments where each segment is broadcast on a separate channel. Thus, the startup latency depends on the latency necessary to start receiving only the first segment. Subsequent segments are guaranteed to be available when needed by PB's broadcast schedule construction. Improvements to PB have been proposed, such as the Permutation-based Pyramid Broadcasting (PPB) [2], Skyscraper Broadcasting (SB) [7], the Client–Centric Approach (CCA) [8] etc. Part of the improvements of PB's variants was the reduction of the necessary client–side secondary storage. Clearly, as the introduction reveals, reduction of the client–side secondary storage is in our view the wrong priority.

All of the above periodic broadcast protocols (PB, PPB, SB, CCA) share a similar structure, that is, the same bandwidth for all channels and increasing size of segment lengths. CCA has shown the best performance by making maximum use of the client bandwidth and keeping a lower buffer space requirement. However, a

common tradeoff is that, in order to decrease the startup latency, the number of segments must be increased and the necessary bandwidth for the server increases proportionately to the number of segments. Variations of this tradeoff include consideration for the specific sizes of the individual segments. However, the size of the segments is dictated by the timing requirements. The intuition behind the segment sizes is that a segment must have begun its cyclic transmission at least once while the previously downloaded segments are being consumed. Such a relation ensures that the n–th segment has already been (at least partly) stored at the set top box just before its consumption has to start.

Another approach to periodic broadcasting schemes, that of Harmonic Broadcasting and its variants, exhibits a different structure, i.e., decreasing bandwidth for the channels and equal segment lengths. In this category fall Juhn and Tseng's Harmonic Broadcasting (HB) [9], Paris, Carter and Long's Cautious Harmonic Broadcasting (CHB) [11], Quasi-Harmonic Broadcasting (QHB) and Polyharmonic Broadcasting (PHB) [12]. These schemes reduce the start–up latency and improve the bandwidth efficiency over PB and its variants. PHB demonstrated the best performance, especially as far as bandwidth efficiency is concerned. Typical transmission bandwidths were reported to be roughly 5 times the video consumption bandwidth. What HB and its variants suggest is that it is possible to increase the number of segments without necessarily increasing proportionately the required server bandwidth.

The previously proposed periodic broadcast schemes assume CBR encoded videos. For similar video quality, CBR encoded video requires twice or more bandwidth than the average bandwidth of the corresponding VBR encoded video [4], Clearly, there are substantial gains by replacing CBR by VBR encoded videos. The current schemes that obtain further performance improvements by using VBR video are few, and rely on the same definition of segments as the CBR–based schemes. Two recent proposal extending broadcast schedules to VBR are [10] and [14]. In these schemes, it is assumed that the clients cannot start downloading and playout the video until the beginning of the first segment is encountered. While the clients consume the first segment, they also concurrently download segments whose playout is forthcoming according to a specific download strategy [7, 8]. Moreover, the server broadcasts each video segment (of the total K_m segments for the m–th video) at its nominal frame rate of F frames per second, that is, at the consumption rate of the video. Frames from the $\sum_{m=1}^{M} K_m$ streams are multiplexed into the broadcast channel without buffering. Data loss occurs whenever the aggregate rate of all broadcast segments exceeds the capacity of the shared link.

VBR–B [14] uses a geometric series of video segments. The segments are then aggregated and the data loss performance is improved by the addition of smoothing, buffered multiplexing or the Join–Shortest–Queue (JSQ) prefetch discipline. It must be noted, any scheme that improves loss performance but results in variable delays requires a compatible jitter absorption technique at the receiver. Another VBR–specific scheme, [10], proposed to use a fragmentation scheme that provides several alternative feasible fragmentations for the same startup latency. Different fragmentation schemes could lead to different aggregate traffic shape when multiplexing the segments together, resulting in very different bandwidth requirements and data losses. Although [10] improves on [14], it does so at an increased computational cost, because it explores a large set of alternative feasible schedules. At the same time, the inherent drawback of both schemes is the potential for data loss.

Summarizing, it appears that although the problem is completely deterministic in its formulation (a–priori known frame sequences for each video and per–video playout latency demands) the existing ad–hoc schemes treat the problem in a fashion that introduces the probability of data loss. Instead, we propose an algorithmic approach which produces loss–less broadcast schedules for VBR video that minimizes also the per–video bandwidth given the video sequence and a corresponding desired playout latency. The resulting scheme is called the Loss–Less Bandwidth–Efficient (LLBE) broadcast protocol.

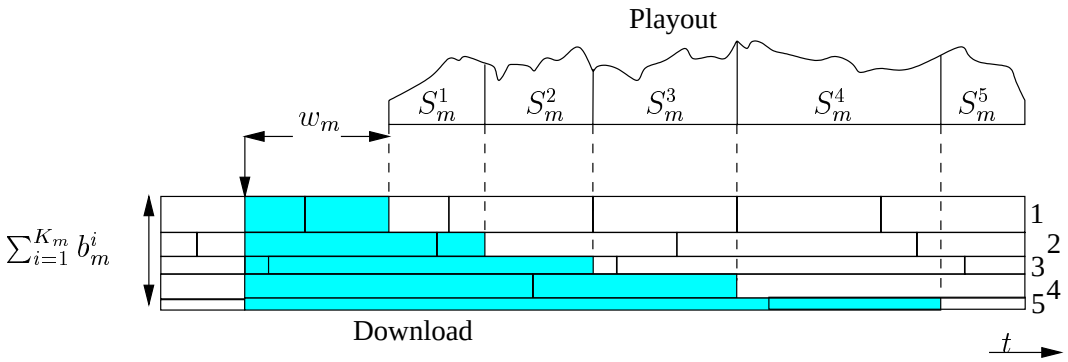

Figure 1: An example of LLBE's operation. The playout part describes the consumption of the video at the client. The download part describes the continuous concurrent periodic broadcast of the five example segments. Each of the five separate segments is periodically transmitted on a channel of bandwidth b_m^i. The shaded area in the download part corresponds to the information that is actually download by the client. Each segment is completely stored at the client just in time for its playout.

3 Definition of the LLBE Protocol

Let M denote the number of VBR videos to be broadcast. The bandwidth of the broadcast link from server to clients is B Mbits/sec. All video streams sent by the server share the B Mbits/sec. The consumption rate of each video is F frames per second. The trace sequence of each video is fully known *a priori*. Let f_m^i, $i = 1, ..., N_m$, $m = 1, ..., M$ stand for the number of bits in the i–th frame of the m–th video, where N_m denotes the total number of frames of the m–th video.

The m–th video is divided into K_m segments prior to broadcasting. The server broadcasts $\sum_{m=1}^{M} K_m$ video streams simultaneously, each stream periodically broadcasts the same segment of the same video. A central point of a periodic broadcast scheme is the way in which the videos are fragmented. The m–th video is fragmented into K_m segments of different sizes. Let S_m^i denote the set of successive frames of the i–th segment of the m–th video, we have:

$$N_m = \sum_{i=1}^{K_m} |S_m^i| \tag{1}$$

In LLBE, each of the K_m segments is transmitted on a separate channel of bandwidth b_m^i, $i = 1, \ldots, K_m$. Thus, the total bandwidth necessary to transmit the m–th video given a fragmentation $S_m = \{S_m^i, \ldots, S_m^{K_m}\}$ is

$$B_m(S_m) = \sum_{i=1}^{K_m} b_m^i(S_m) \tag{2}$$

In LLBE the client behaves in a greedy fashion, that is, it starts downloading all K_m segments that correspond to the desired video. Moreover, the download of all the segments starts at the same time point: the tune–in instant. Thus, equation (2) is both the required server and required client bandwidth. As segments get gradually consumed, the necessary client bandwidth decreases. That is, the maximum client bandwidth is necessary in the beginning of the video reception. Essentially, the required per–video server and client bandwidth are coupled. In the next section, a minimization process is employed to reduce the required server and client bandwidth on a per–video fashion. The construction of LLBE is a generalization of the GEBB scheme [6]. GEBB was restricted to CBR video but possessed a closed form solution for determining the optimal segment sizes that minimized the total bandwidth necessary for a given playout latency.

Following the construction illustrated on Figure 1, and in order to guarantee the startup latency w_m, we note that the first segment must be broadcast at a bandwidth:

$$b_m^1(S_m) = \frac{\sum_{i \in S_m^1} f_m^i}{w_m} \tag{3}$$

Equation (3) establishes that the surface of the first horizontal rectangle of the step–function for the client download in Figure 1 is the same as the surface of the area denoted by the S_m^i in the playout graph of the same figure. A similar relation is true for all subsequent segments. That is, in order for the continuous and timely playout, the next segment, S_m^i, is fully downloaded and available at the client just before the end of the playout of segment S_m^{i-1}. Thus, the bandwidth at which the i–th segment of the m–th video is broadcast is determined by:

$$b_m^i(S_m) = \frac{\sum_{i \in S_m^i} f_m^i}{w_m + \frac{\sum_{j=1}^{i-1} |S_m^j|}{F}} \tag{4}$$

Overall, after the startup latency w_m seconds, the uninterrupted playout of the video is possible. Each segment is available on local storage at the client just–in–time for its playout. The data are played out at their nominal frame rate of F frames per second in the order of $S_m^1 \bullet S_m^2 \bullet S_m^3 \ldots S_m^{K_m}$. Note that because the frames have different sizes, the playout curve of Figure 1 which represents the per–frame consumed data is also variable. From equations (3) and (4), we conclude that the server bandwidth requirement to broadcast the m–th video using LLBE for any valid transmission schedule, S_m, can be expressed by:

$$B_m(S_m) = \sum_{i=1}^{K_m} \frac{\sum_{j \in S_m^i} f_m^j}{w_m + \frac{\sum_{j=1}^{i-1} |S_m^i|}{F}} \tag{5}$$

The loss–less nature of LLBE is evident since the aggregation of the bandwidth required for all the K_m segments of the m–th video, is constant as described by equation (5).

One can observe that several feasible schedules exist, since the construction of S_m is based on fairly mild constraints. Namely, the only constraint necessary for S_m to be a feasible schedule is that all segments together cover the entire video and that they do not overlap. The bandwidth–efficiency component of LLBE comes from the selection of the feasible schedule that minimizes the required server (client) bandwidth (equation (5)). Our objective is thus to determine S_m^i's such that a broadcast schedule can be constructed which results in the least amount of bandwidth requirement B_m^*, $B_m^* = \inf\{B_m(S_m) | S_m \text{ is a feasible schedule}\}$. Essentially, we wish to determine an ordered sequence of $K_m - 1$ integers (the "boundaries" between the segments) in the range 1 to $N_m - 1$ that minimizes the value of equation (5).

4 Calculation of the Optimal S_m and B_m^*

In formulating the optimal selection for S_m to achieve the minimum required server bandwidth, we consider a discrete–time model at the frame level. That is, the segments are to consist of an integer number of frames. Let us consider the collection from the i–th to the $(j-1)$–st frame of a video. Let us define $C_m^{i,j}$ as the bandwidth necessary if the group of frames from i to $j-1$, inclusive, were to form a segment:

$$C_m^{i,j} = \frac{\sum_{t=i}^{j-1} f_m^t}{w_m + \frac{(i-1)}{F}} \tag{6}$$

Using the cost definition of equation (6) and given a sequence of video frame sizes f_m^i, we can construct a Directed Acyclic Graph (DAG) with vertices labeled from 1 to $N_m + 1$ and with directed edges from i to j where

$N_m+1 \geq j > i \geq 1$. The edge weights are as defined by the cost equation (6). A fragmentation to K_m segments (or less) amounts to finding a path from 1 to N_m+1 consisting of K_m edges (or less). Since the minimization we wish to apply corresponds to the minimization of the cost of the edges participating in the path, the optimization problem is in fact a shortest–path problem. Figure 2 illustrates the relation of the shortest path and the optimal fragmentation of a video for $K_m = 3$.

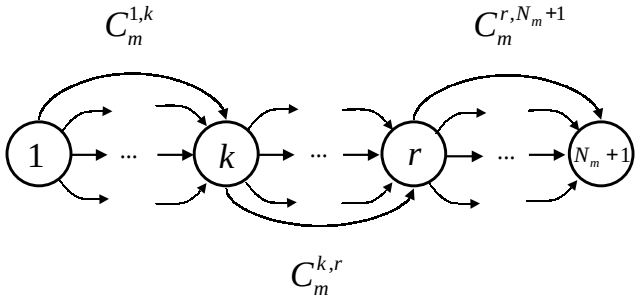

Figure 2: Example shortest path and optimal fragmentation, S_m, for $K_m = 3$ given a certain w_m. Segment S_m^1 spans frames from 1 to $k-1$. Segment S_m^2 spans from k to $r-1$. Segment S_m^3 spans from r to N_m.

It appears appealing to apply the Bellman–Ford algorithm, specialized to single–source (from vertex 1) shortest path. By terminating the algorithm after its $(K_m - 1)$–st iteration, we guarantee that all paths determined up to this point are the shortest paths with K_m *or less* edges. Thus, the resulting running time is $O(K_m(N_m)^2)$. However, a source of complications is not the running time but the need to preserve the original cost matrix $C_m = \{C_m^{i,j}\}$ in main memory. The space requirement for the C_m matrix is $O((N_m)^2)$ but because typical values of N_m are $160,000$ or more (for a two-hour video file) it is not practical to assume that the cost matrix can be stored entirely in main memory in currently available systems.

To solve the problem of the large memory requirements, we introduce a vector $A_m = \{A_m^i\}$ which contains the prefix sums of the frame size sequence, that is:

$$A_m^i = \sum_{j=1}^{i-1} f_m^j, \quad i = 1, \ldots, N_m + 1 \tag{7}$$

From equations (6) and (7), $C_m^{i,j}$ can be restated as:

$$C_m^{i,j} = \frac{A_m^j - A_m^i}{w_m + \frac{i-1}{F}} \tag{8}$$

By introducing equation (8) the memory requirements are decreased to $O(K_m N_m)$. The calculation of $C_m^{i,j}$ for any i, j pair is performed upon demand based on the A_m vector which is used to look–up the prefix sums. A_m is calculated in $O(N_m)$ once at the beginning of the run. Figure 3 presents the pseudocode of the specialized shortest–path algorithm, and it can be clearly seen that it exhibits a time complexity of $O(K_m(N_m)^2)$ and space complexity of $O(K_m N_m)$.

We note that for LLBE, the scheduling process for each video is independent from all the other videos that share the same link. In other words, once the transmission schedules for a specific video is constructed, the required bandwidth for transmitting this video is finalized and it is only dedicated for this video.

$sum = 0$;
for $i = 1, \ldots, N_m + 1$ **do**
  $A_m^i = sum$;
  $D_m^i = A_m^i$;
  $W_m^i = D_m^i$;
  $sum = sum + f_m^i$; // Note: $f_m^{N_m+1} = 0$
endfor
for $iter = 1, \ldots, (K_m - 1)$ **do**
  for $j = 2, \ldots, N_m + 1$ **do**
    for $i = 2, \ldots, (j - 1)$ **do**
      if $\left(W_m^j > D_m^i + \frac{A_m^j - A_m^i}{w_m + \frac{i-1}{F}} \right)$ **then**
        $W_m^j = D_m^i + \frac{A_m^j - A_m^i}{w_m + \frac{i-1}{F}}$;
        $r_m^{iter,j} = i$;
      endif
    endfor
  endfor
  D_m = W_m; // array–wide operation
endfor

Figure 3: Pseudocode of single–source Bellman–Ford version of the K_m–step shortest path algorithm using an $O(N_m)$ prefix sum array, A_m. W_m is a work array. $r_m^{iter,j}$ holds the intermediate vertex on the path to j at the $iter$ iteration. $r_m^{iter,j}$ is $(K_m - 1) \times N_m$ and is initialized to NULL.

5 Experimental Results

The presented experiments are based on the application of LLBE on a set of sample traffic traces coded according to the MPEG-1 standard [13]. All traces used herein are 40000 frames long, captured at 25 frames per second with a GOP of 12 frames. We selected a total of $M = 10$ videos from the set of available traces. The set contains both low–motion and high–motion content, namely it includes news broadcasts, feature movies, music videos, sporting events and animations. We do not present separate per-video results for all video traces examined since the performance results we derived lead to the same basic conclusions regardless of the video content.

First we look into the influence of K_m and w_m on the required server bandwidth for some representative video traces. We anticipate that increasing K_m or w_m decreases the required bandwidth. Moreover, a question that arises naturally by observing the slowly increasing bandwidth demand in HB and its variants is whether there exists an asymptotic bandwidth demand which, given a startup latency objective, is the limit of the required bandwidth as the number of segments K_m tends to infinity. The results presented in Figure 4 suggest that this asymptotic behavior is indeed the case. Hence, if we factor out the constraint that the number of segments is limited by technological constraints, we claim that there exists an inherent bandwidth limit for a given startup latency w_m. This is demonstrated for VBR video traffic in LLBE, and trivially, the same is true for CBR video as well. Consequently, we can claim that the experimental results suggest that for LLBE:

$$\lim_{K_m \to \infty} B_m^* = constant < \infty \tag{9}$$

An exact expression for the asymptotic bandwidth demand for a given playout latency is in fact known for

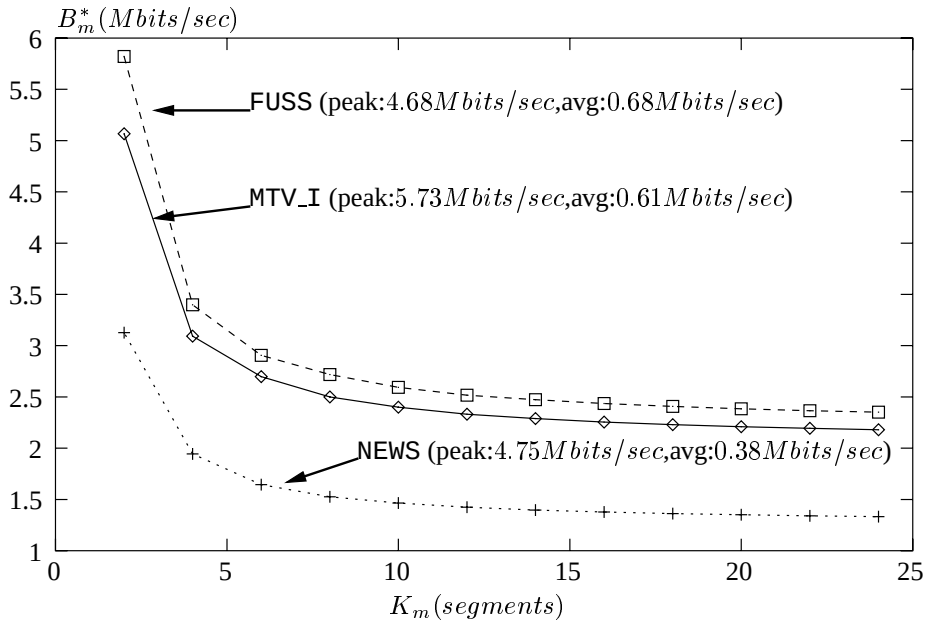

Figure 4: Optimal server bandwidth, B_m^*, demand for three sample video streams vs. variable number of segments K_m. ($w_m = 60\ sec$)

the special case of CBR video [6].

Figure 4 presents the results for three video traces for variable K_m from 2 to 24 and for a fixed startup latency $w = 60sec$. Similar curves were observed for all examined video traces. We note that the value at which B_m^* converges depends on the *exact* video trace under consideration, that is, different videos converge to different values. The convergence value does not depend only on the average frame size or any other trace–wide statistic, but instead on the *exact* sequence of frame sizes.

LLBE, by its definition, produces segments that are broadcast on channels of different bandwidths. An implementation issue is how to schedule a given broadcast medium between several channels/streams such that each one is allocated a different portion of the available bandwidth. The actual complexity is in providing the mechanism for allowing the allocation of *arbitrary* bandwidth to each channel/stream. The current literature provides already a large assortment of scheduling mechanisms, e.g., Weighted Fair Queueing (WFQ) and its approximations, to allow implementations of arbitrary bandwidth allocations. Instead of the scheduling issue, we focus on the related issue of quantized bandwidth allocation. That is, on the inefficiency of using a scheduling scheme that allows only multiples of a basic rate to be allocated. Scheduling schemes that provide quantized bandwidth allocation are, for example, the Time Division Multiplexing (TDM) schemes. Their inefficiency in terms of quantized bandwidth allocation is compensated by the straightforward multiplexing and demultiplexing mechanism.

We consider a TDM system with a basic rate of $64\ kbits/sec$ (ISDN rate). Thus, we consider this particular example as an illustration of what an implementation using conventional voice channels would require in terms of bandwidth. Namely, the total number of ISDN channels necessary for video m are:

$$C_m = \sum_{i=1}^{K_m} \lceil b_m^i / (64\ kbits/sec) \rceil \tag{10}$$

The observation we derive from experimental results for increasing K_m in Figure 5 is the disappearance of the monotonic decrease of B_m^* for increasing K_m. The reason is the *internal fragmentation* within the quanta of

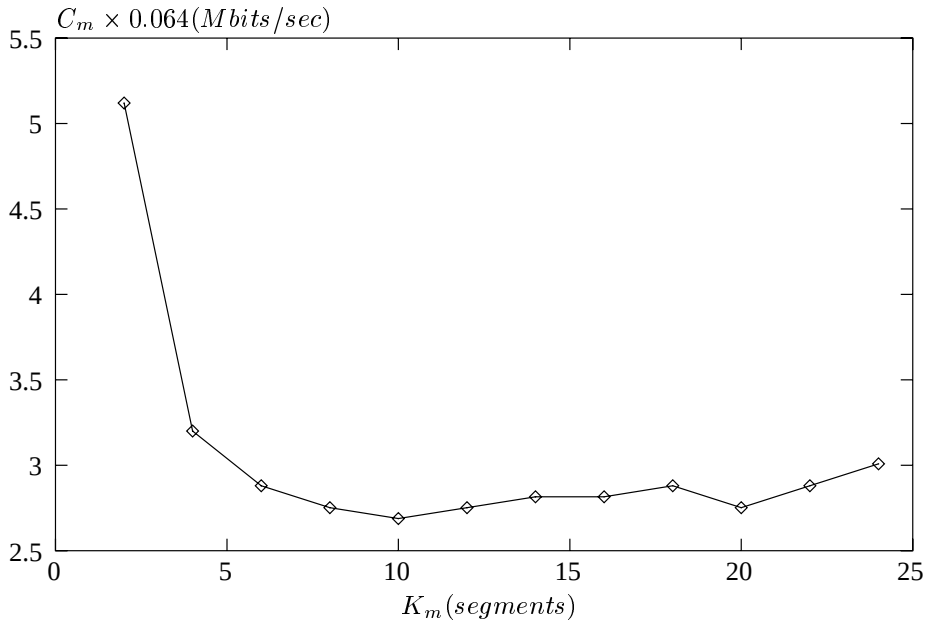

Figure 5: Optimal quantized server bandwidth, $C_m \times 0.064$ $(Mbits/sec)$, demand for a sample video vs. variable number of segments K_m. ($w_m = 60$ sec, trace `MTV_I`)

allocated bandwidth. For example, if $b_m^i = 130$ $kbits/sec$, then the corresponding channels for this video segment requires $3 \times 64kbits/sec$, leaving $3 \times 64 - 130 = 62kbits/sec$ unused. In order to provide straightforward demultiplexing, no sharing of a primary rate channel is allowed between two or more segments. Thus, as the number of segments increases, the per–channel bandwidth may decrease, leading to an increase of the level of internal fragmentation.

A comparison of Figures 4 and 5 indicates that because of the internal bandwidth fragmentation the overall bandwidth allocation can indeed be larger in the quantized case. If, instead, WFQ was used to provide an arbitrary bandwidth allocation, then a jitter absorption scheme is necessary because the implementations of WFQ can only approximate the fluid model assumed for the traffic, and service is dispensed in packet or cell quanta. Jitter absorption is also necessary for TDM systems. In TDM, the necessary jitter absorption is defined by the length of the repeating TDM cycle (each channel is allocated one or more slots within the TDM cycle, where each slot in the current example corresponds to a $64kbits/sec$ quantum).

Dealing with the internal fragmentation inefficiency in a TDM system requires the generation of segments that use bandwidth very close (but lower) to an integer multiple of the basic rate. A strategy that can be used for reducing the internal fragmentation is the following: at the DAG construction stage that we illustrated earlier, we omit all edges that correspond to bandwidths that lead to large internal fragmentation. Hence, the shortest path, once created, uses only edges (and corresponding channel bandwidths) that to not exacerbate the internal fragmentation. However, such technique is not valid in general because it can lead to a disconnected graph, prohibiting the construction of a complete path. It is therefore recommended to, instead, explore the set of allowed values of K_m for each video (which can be accomplished in one single run of the shortest path algorithm for up to the maximum K_m) and to select the value that minimizes the quantized required bandwidth upon inspection of a graph similar to Figure 5.

Next we explore the performance of the required server bandwidth B_m^* as a function of w_m. Figure 6 presents the results that are representative of the examined video traces. Namely, the required bandwidth indeed decreases with increasing startup latency. The construction of LLBE allows the following very intuitive asymptotic behavior

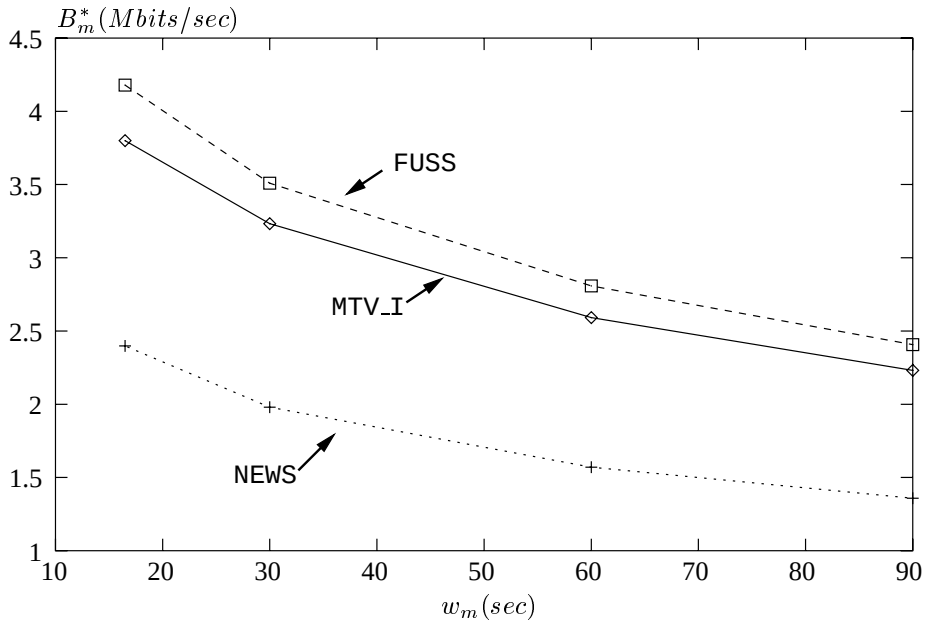

Figure 6: Optimal server bandwidth, B_m^*, demand for three sample videos vs. startup latency w_m. ($K_m = 7\ segments$)

(for a given K_m):

$$\lim_{w_m \to \infty} B_m^* = 0 \tag{11}$$

and

$$\lim_{w_m \to 0} B_m^* = \infty \tag{12}$$

That is, if the clients can wait for a long period of time, the necessary bandwidth is virtually zero (equation (11)). Inversely, instant playout startup requires essentially infinite bandwidth. What is important to observe, is the fast reduction of the necessary bandwidth for "reasonable" values of w_m. For example, for the MPEG-1 videos that we examined and $w_m = 60\ sec$ the necessary bandwidth is not more than 2.5 to $3\ Mbits/sec$ and for a latency of a few minutes, the necessary bandwidth can be equal to the average bandwidth necessary of the corresponding MPEG-1 trace. Compared to the typical commodity disk drive I/O throughput of a few $Mbytes/sec$ the necessary bandwidth is indeed well below the technological limits of low–cost set–top boxes for "reasonable" values of w_m even if we assume $N_m = 160000$ or more.

The next set of observations we make are related to the bandwidth necessary for each segment. Figure 7 illustrates the relation of the bandwidth of all the segments of the same video. For sufficiently large K_m, and for the segments in the later part of the trace the following is consistently observed:

$$b_m^i \approx b_m^{i+1} \tag{13}$$

In Figure 7 the above behavior is noticeable for $i > 12$. This observation is also in agreement with the intuition behind the construction of the schedules with equal bandwidth segments (PB and variants) and also in agreement with the results observed in GEBB [6].

For the same video trace (FUSS), Figure 8 illustrates the relation of the segment sizes. Notice the logarithmic scale used for the y–axis on Figure 8. Hence, the almost linear increase in the size of the later segments corre-

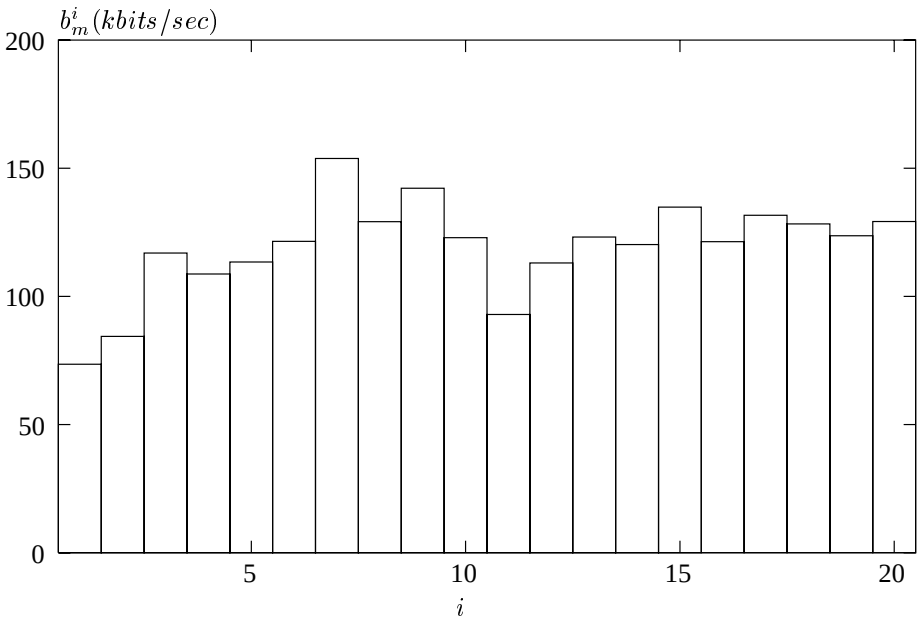

Figure 7: Per–segment server bandwidth, b_m^i, for each segment. ($K_m = 20\ segments$, $w_m = 60\ sec$, trace FUSS)

sponds to an exponential increase of successive segment sizes (for $i > 12$). Similar behavior was observed in all the other video traces as well. The observations can be summarized in the following relation:

$$\sum_{j \in S_m^{i+1}} f_m^j \approx \beta \sum_{j \in S_m^i} f_m^j \tag{14}$$

The above relation is also the intuition behind the CBR–based schemes, and the variants of PB in particular. We note however that the factor of the geometric series, β, depends on w_m, K_m and the exact frame sequence of the trace.

The later segment sizes, by accumulating a larger number of frames together, are almost equal to the product of the number of frames multiplied by the average frame size. Essentially, for large i, the corresponding segments depend on the average frame size statistics. On the other hand, for small values of i, the segments consist of a small collection of frames and they exhibit higher variability which depends on the exact collection of frames. Hence, for large i, an exponential increase of the length of the segments (in terms of numbers of frames) is also observed (for the sake of brevity, we omit presenting a corresponding figure).

A final performance comparison can be performed against the two schemes that have been proposed in the past for VBR periodic broadcast, VBR–B and TAF [10, 14]. We first note that the segments sizes in VBR–B follow a geometric sequence with a factor of 2. TAF generalizes the segment length selection. The results shown in Figures 7 and 8 indicate that the selection of the "best" segment lengths (sizes) may follow a sequence which is not necessarily monotonically increasing. This is in direct contrast to the construction of VBR–B and TAF. The ability of LLBE to guarantee the timing constraints given a non–increasing segment lengths is due to its particular "greedy" nature of the client download strategy. Furthermore, the size of segments, in terms of bits, compared to their size in terms of frames are drastically different. For example, the number of frames corresponding to the 8–th segment of Figure 8 is approximately twice the number of frames of the 9–th segment, but the size in bits of the 8–th segment is less than that of the 9–th segment.

To compare the bandwidth efficiency of multiplexing 10 videos in the proposed LLBE with that of VBR-B

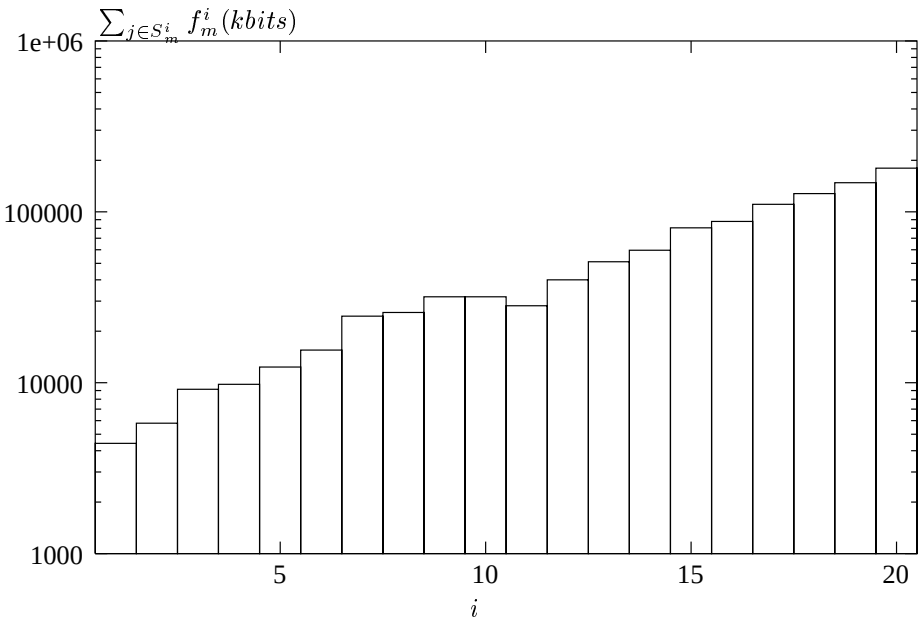

Figure 8: Segment sizes, $\sum_{j \in S_m^i} f_m^j$, for each segment. ($K_m = 20\ segments$, $w_m = 60\ sec$, trace FUSS)

[14] and TAF [10], we set the number of segments for each videos, K_m to 7. The startup latency w_m is set to 16.5 seconds. Figure 9 demonstrates the data loss rate. The bandwidth from the aggregation of the 10 videos according to LLBE is used as bandwidth for VBR–B and TAF. The results clearly indicate that under VBR–B and TAF, the resulting data loss (approximately 15 and 10 %) is high enough to render the video distribution useless in such limited bandwidth. The exception is LLBE. In order to achieve the same zero data loss rate, the required bandwidth for transmitting the 10 movies for LLBE is much lower than that for VBR-B and TAF (Figure 10). For LLBE, the required bandwidth is 33.59 Mbits/sec, while for VBR-B it is 86.96 Mbits/sec and 60.72 Mbits/sec for TAF.

Scheme	Loss Rate
LLBE	0.000
VBR-B	0.153
TAF	0.104

Figure 9: Data loss rate for three VBR broadcasting schemes for a collection of 10 videos. ($K_m = 7\ segments$, $w_m = 16.5\ sec$, $B = 33.6\ Mbits/sec$)

6 Conclusions and Future Directions

In this paper we consider the problem of loss–less periodic broadcast of VBR video for VoD systems. Given the continuous decline in prices for secondary storage devices, we relax the constraints related to the client–side secondary storage size. Moreover, because of the particular structure and distribution setup for video information, we relax the assumption that was used to force the start of downloading to coincide with the beginning of a segment. The resulting scheme, LLBE, provides enough degrees of freedom (the sizes of the individual segments)

Scheme	$\sum_{i=1}^{K_m} b_m^i$
LLBE	33.587
VBR-B	86.958
TAF	60.722

Figure 10: Required server bandwidth in Mbits/sec for zero data loss rate for three VBR broadcasting schemes for a collection of 10 videos. ($K_m = 7\ segments$, $w_m = 16.5\ sec$)

to allow its formulation as an optimization problem. We show the equivalence of the optimization problem to a shortest path problem on a DAG. The storage needs for the shortest path problem cannot be supported with commodity main memory storage resources. Hence, we refine the shortest path algorithm for the specific problem and greatly reduce its main memory requirements.

The results we derive indicate that the segment sizes and bandwidth allocation approaches in previously proposed schemes used the correct intuition but were unable to quantify the problem properly to support the case of VBR encoded video. The optimization algorithm proposed for LLBE allows us to exactly quantify the necessary parameters based on the particular stored video content and startup latency objective. Moreover, because LLBE handles the distribution of VBR video as a collection of per–segment CBR streams, it inherently avoids data losses that plagued previous schemes (even after they included additional schemes to rectify the losses – at significant implementation complexity).

Currently, in LLBE, the server and client bandwidth demands for a video are identical. Future research in this direction will focus on the reformulation of the optimization problem in a manner that will allow a–priori constraints to be enforced on the client–side I/O bandwidth separately from constraints enforced on the server–side bandwidth. We are also exploring techniques to support interactive VCR–like operations and to support potentially heterogeneous clients/set–top boxes. Another possible use is the application of the decreasing–only download bandwidth, as dictated by LLBE, to the call admission in a batched multicast or advance reservation system. One approach to this end is the generalization of LLBE, in a similar manner to the generalization of Skyscraper Broadcasting towards the Dynamic Skyscraper Broadcasting of Eager and Vernon.

References

[1] Aggarwal, C., Wolf, J., Yu, P., "On Optimal Batching Policies for Video-on-Demand Storage Servers", In Proc. of IEEE Int'l Conf. on Multimedia Systems '96.

[2] Aggarwal, C., Wolf, J., Yu, P., "A permutation-based pyramid broadcasting scheme for video-on-demand systems", In Proc. IEEE Int'l Conf. on Multimedia Systems '96.

[3] Almeroth, K., Ammar, M., "The Use of Multicast Delivery to Provide A Scalable and Interactive Video-on-Demand Service", IEEE J. Selected Areas Commun., Aug. 1996.

[4] Dalgic, I., Tobagi, F., "Characterization of quality and traffic for various video encoding schemes and various encoder control schemes," Technical Report CSL-TR-96-701, Department of Electrical Engineering and Computer Science, Stanford University, August 1996.

[5] Dan, A., Sitaram, D., Shahabuddin, P., "Scheduling Policies for an On-Demand Video Server with Batching," In Proc. of ACM Multimedia, pp. 15-23, Oct. 1994.

[6] Hu, A., Nikolaidis, I., van Beek, P., "On the Design of Efficient Video–on–Demand Broadcast Schedules," In Proc. of MASCOTS '99, pp. 262–269, Oct. 1999.

[7] Hua, K., Sheu, S., ”Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems”, In Proc. of SIGCOMM ’97, pp. 89-100.

[8] Hua, K., Cai, Y., Sheu, S., ”Exploiting Client Bandwidth for More Efficient Video Broadcast”, In Proc. IEEE Int’l Conf. on Computer, Communications and Networks ’1998.

[9] Juhn, L., Tseng, L., ”Harmonic Broadcasting for Video-on-Demand Service”, IEEE Transactions on Broadcasting, 43(3): 268-271, Sept. 1997.

[10] Li, F., Nikolaidis, I., ”Trace-Adaptive Fragmentation for Periodic Broadcast of VBR Video”, In Proc. of the 9th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV ’99), June 1999.

[11] Paris, J., Carter, S., Long, D., ”Efficient Broadcasting Protocols for Video on Demand”, In Proc. of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS ’98), PP. 127-132, July 1998.

[12] Paris, .J., Carter, S., Long, D., ”A Low Bandwidth Broadcasting Protocol for Video on Demand”, In Proc. ot IEEE Int’l Conf. on Computer Communications and Networks ’1998.

[13] Rose, O., ”Statistical Properties of MPEG Video Traffic and Their Impact on Traffic Modelling in ATM Systems”, Technical Report 101, University of Wuerzburg, Germany, Feb. 1995.

[14] Saparilla, D., Ross, K., Reisslein, M., ”Periodic Broadcasing with VBR-Encoded Video”, In Proc. of IEEE Infocom ’99, March 1999.

[15] Viswanathan, S., Imielinski, T., ”Metropolitan Area Video-on-Demand Service Using Pyramid Broadcasting”, Multimedia Systems, 4(4): 197-208, Aug. 1996.