# Trace–Adaptive Fragmentation for Periodic Broadcast of VBR Video

Fulu Li and Ioanis Nikolaidis
Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada, T6G 2H1
{fulu,yannis}@cs.ualberta.ca

## Abstract

*Periodic broadcast schemes have been proposed as a scalable solution for the implementation of Video–on–Demand (VoD) systems. Efficient periodic broadcast schemes fragment each video into a number of segments assuming that the videos are encoded using Constant Bit Rate (CBR) coders. We focus instead on the more efficient and commonplace class of Variable Bit Rate (VBR) encoded videos. The claim made in this paper is that the significant bit rate variability of VBR video requires a different fragmentation approach for the construction of periodic broadcast schedules. Rigid fragmentation techniques do not consider the particular video traces and may result in traffic which exceeds the available broadcast link capacity and can lead to increased data losses. Remarkably, all known periodic broadcast schemes use rigid fragmentation techniques. We introduce a new fragmentation scheme, Trace–Adaptive Fragmentation (TAF), which derives improved broadcast schedules for VBR video. Essentially, increased processing time for an off–line optimization process is traded off for improved data loss performance while also satisfying a prescribed maximum playout latency. Numerical results show the benefits, feasibility and flexibility of the proposed scheme.*

## 1. Introduction

In recent years, several techniques have been proposed for the implementation of scalable Video-on-Demand (VoD) systems [1, 2, 3, 6, 8, 7, 9, 10, 11, 12, 14, 15]. The techniques can be categorized into (a) *batched multicast* and (b) *periodic broadcast* schemes. Batched multicast schemes collect user requests (“batching”) over successive time intervals. The subscribers requesting the same video within the same interval will receive the video stream through a single multicast of the entire video. Periodic broadcast schemes, as the name suggests, operate by periodically broadcasting the same video. Depending on the exact scheme, the video is transmitted as one complete unit or fragmented into separate segments and each segment transmitted over a different channel. The lengths of the segments, normalized relative to the first segment, is also called the *broadcast series* of the periodic broadcast scheme.

Compared to batched multicast, periodic broadcast schemes are considered scalable because they bypass the need to process individual user requests. The transmission schedules used by periodic broadcast are determined off–line. In this sense, the periodic broadcast schemes for VoD can be seen as an evolution of earlier *teletext broadcast* schemes [4]. The computation of the schedules can account for the popularity of the videos and the maximum tolerable latency between the time a subscriber activates its set–top box (“tunes-in”) and the point at which the uninterrupted playout of the entire video can start. Clearly, an objective is the minimization of the delay between “tune-in” and start of playout, hereafter called the *playout latency*.

The periodic broadcast schemes require large bandwidth, especially if short playout latency is required. The bandwidth allocated for the transmission of all the segments comprising a video is several times the bandwidth necessary to transmit a single instance of the video from beginning to end at its nominal frame rate (also called *consumption bandwidth* of a video). In addition, the broadcast schemes impose particular synchronization relations between the different segments of the same video. Finally, the bandwidth requirements are further compounded by the fact that most periodic broadcast schemes assume Constant Bit Rate (CBR) coded video. It has been observed [5] that, for the same video quality, the average CBR bandwidth demands far exceed the average bandwidth demands of Variable Bit Rate (VBR) coded video. In addition, VBR coding schemes are becoming commonplace (MPEG-1 and MPEG-2) and extensive libraries of video material are already available in VBR form.

The technique presented in this paper is suitable for VBR as well as for CBR coded videos. The work is influenced by the CCA [7] periodic broadcast schemes and by recent advances towards the periodic broadcast support of VBR encoded videos [14]. We expand the class of CCA schemes in a way that allows us to derive a set of alternative fragmentations while still satisfying the desired playout latency constraint. The wealth of fragmentation choices provides direct benefits for VBR coded videos. Namely, different fragmentations can result in drastically different data loss performance (due to limited link capacity). We exploit this property to expand the results of [14] by adopting a per–video fragmentation scheme which results in less data loss than the standard "rigid" fragmentation employed in [14]. The selection of the optimal fragmentation scheme from all the feasible ones is the subject of an off–line optimization process. The resulting scheme, called Trace-Adaptive Fragmentation (TAF), combines and improves the properties of [7] and [14].

The rest of the paper is organized as follows: Section 2 provides a review of the related research. Section 3 outlines the scope of the paper and the assumptions made. Section 4 provides the characterization of the two important quantities of this study, the playout latency and the data loss probability. Section 5 provides in detail the proposed Trace–Adaptive Fragmentation (TAF) scheme. Section 6 summarizes our experimental results and Section 7 provides an overview and conclusions as well as areas of future research.

## 2. Related Research

Batched multicast schemes are particularly inefficient for the distribution of popular videos. Assuming a two hour long video and batching intervals of 5 minutes, a popular video may result in as many as 24 concurrent multicasts of the same video. Reducing the replicated transmissions is possible if the batching interval is increased, thus resulting in *worse* playout latency for the most popular videos.

It has been repeatedly shown in the literature that the popularities of videos follow the Zipf distribution. A typical skew factor for a Zipf distribution representing video preferences is 0.271 (see [6]). That is, most of the demand (80%) is for a few (10 to 20) very popular videos. Periodic broadcast schemes can then be used for the popular set ("hot set") of videos while a form of batched multicast can be used for the less popular set ("cold set"). Thus, batched multicast schemes are still valuable for scheduling the cold set. In the following we focus in the distribution of hot set videos using periodic broadcast.

The staggered periodic broadcast of entire videos [6] is a scheme that requires bandwidth inversely proportional to the playout latency objective. For example, on a 155 Mb/sec link, multiplexing the broadcast of ten two-hour long movies (each encoded at 3 Mb/sec) we cannot provide a playout latency better than 24 minutes. Moreover, this type of staggered broadcast demands large secondary storage capacity and high secondary storage I/O capabilities at the set–top box. The solution to these issues is the fragmentation of the complete video into a sequence of segments. The first segment can be short such that its broadcast can be repeated very often resulting in short playout latency. The complexity lies in the timing of the broadcast of subsequent segments such that no starvation of the receiver occurs. Moreover, the receiver can employ secondary storage to store (completely or partially) segments in anticipation of their playout.

The Pyramid Broadcasting (PB) presented by Viswanathan and Imielinski [15] was the first scheme to reduce the playout latency using fragmentation. In PB, each segment is broadcast on a separate channel. However, PB's drawbacks are (a) the large client buffer size which is usually more than 70% of the entire video and (b) the high disk bandwidth required to write data to disk as quickly as received. A number of efficient protocols have been proposed to address these issues. Aggarwal et. al. proposed the Permutation-based Pyramid Broadcasting (PPB) [2] which reduces the buffer size requirements down to approximately 50% of the video. Skyscraper Broadcasting (SB) presented by Hua et. al. [8] substantially reduces the disk to approximately 10% of the video using a novel fragmentation technique and a different broadcasting scheme. Recently, Hua et. al. proposed a Client-Centric Approach (CCA) [7] which incorporates the restriction on how many channels a receiver can download at any point in time.

Overall, the above periodic broadcast protocols (PB, PPB, SB, CCA) share a similar structure, that is, equal 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. The bandwidth requirements of this family of protocols, for reasonable playout latencies, is roughly 7 times the video consumption bandwidth.

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) [10], Quasi-Harmonic Broadcasting (QHB) and Polyharmonic Broadcasting (PHB) [11]. These schemes aim to reduce the start–up latency and improve the bandwidth efficiency. PHB has given the best performance, especially as far as bandwidth efficiency is concerned. Typical storage requirements are near 40% of the video size and transmission band-

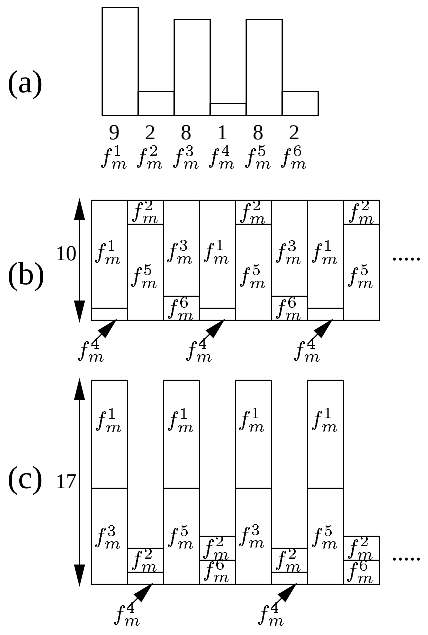

**Figure 1. An example of a video frame sequence, (a), (the numbers are the frame sizes) and two alternative fragmentation options for the same sequence. In (b) the broadcast series is $\{1, 1\}$ and the aggregate peak is 10 units. In (c) the broadcast series is $\{1, 2\}$ and the aggregate peak is 17 units.**

width are roughly 5 times the video consumption bandwidth.

Until recently, all periodic broadcast techniques assumed CBR encoded videos. Recently, [14] proposed a series of multiplexing schemes for the periodic broadcast of VBR encoded video (VBR-B). Due to the significant rate variability of VBR-encoded video, different fragmentation schemes could lead to different aggregate traffic shape when multiplexing the periodically broadcast segments together, resulting in different data loss rates. Figure 1 illustrates how two alternative 2-segment broadcast schedules of the same VBR video, result in substantially different aggregate traffic patterns, depending on the particular lengths of the segments. Based on this observation, we will focus on a way to generate multiple candidate fragmentations for the same playout latency requirement. Subsequently, we will select the fragmentation that minimizes the data loss that occurs due to the limited broadcast link capacity.

## 3. Scope and Assumptions

In this paper we consider only simple VoD service. The key requirement of VoD service is the ability for uninterrupted playout once playout of a video starts. We will not investigate additional (VCR-like) operations such as fast-forward/backward and pause. We note that one can always trivially support VCR-like operations once the entire video is stored on secondary storage.

We will consider the combined broadcast of $M$ videos, with $M$ in the range of 10 to 20 in order to correspond to observed distributions of "hot set" videos. All $M$ videos are multiplexed on the same broadcast physical link of bandwidth $B$ (in Mb/sec). All videos have the same frame rate $F$ (in frames/sec). The frame sequence of each video is fully known a-priori. Let $f_m^n$, $n = 1, \ldots, N_m$, $m = 1, \ldots, M$ stand for the number of bits in the $n^{th}$ frame of the $m^{th}$ video, where $N_m$ denotes the total number of frames of the $m^{th}$ video.

In its general form, the problem of the periodic broadcast of VBR videos can be stated as follows: Given $M$ different videos, characterized by their individual number of frames, $N_m$, $m = 1, \ldots, M$ and the actual per-video frame sizes $f_m^n$, $n = 1, \ldots, N_m$ we wish to fragment them for periodic broadcast and transmit them over a link of capacity equal to $B$ Mb/sec.

Several constraints need to be satisfied for the broadcast schedule construction: (a) the playout latency constraint for video $m$ should be less or equal to $w_m$ seconds, (b) the number of segments video $m$ is split into must be less or equal to $K_m$, and (c) the number of concurrently downloaded channels by the receiver must be less or equal to $C_m$. Note that $K_m$ limits the bandwidth allocated by a VoD server for the $m$–th movie. Similarly, $C_m$ limits the bandwidth necessary for the receiver.

We claim that the objective of minimizing the client secondary storage requirements is of minor importance and it will not be taken into consideration in this paper. Improvements in storage capacities for magnetic hard disk drives as well as the availability of re-writable optical disks allow now the storage of an entire feature-length video. Instead, we believe that the important restriction is that of the I/O throughput of the secondary storage system. $C_m$ can be used to capture such an I/O throughput constraint.

We wish to derive a broadcast series for each of the $M$ videos. The $m$–th video broadcast series is represented by $\{s_m^1, s_m^2, \ldots, s_m^{K_m}\}$. Note that $s_m^1 = 1$ and all other $s_m^i$ are normalized relative to $s_m^1$. A common theme in the current literature is to only consider integer values for $s_m^i$. One benefit of integer $s_m^i$ appears to be the ability to express all time durations as multiples of a basic unit (typically $s_m^1$) and hence reduce the complexity of keeping all the segments synchronized with each other. We will thus maintain in this paper the assumption of integer $s_m^i$.

Each segment is broadcast on a separate *logical channel*. The term *logical channel* is used in VoD literature to capture the use of a single physical broadcast link to convey multiple flows of data. Hence, the link capacity is split into several logical channels. The mechanisms through which the sharing of the physical link is achieved will not be detailed. However, note that in the case of CBR video and logical channels of equal capacity, the mechanism required can be straightforward Time Division Multiplexing (TDM). If the logical channels are of different but constant bandwidths, then a Fair Queueing scheduler could be used. In the case of variable bandwidth per logical channel (as in the case of VBR video), the sharing is achieved through statistical multiplexing. Depending on the scheme used, a corresponding jitter absorption scheme is necessary at the receiver. Given that the multiplexing we consider in this paper is *bufferless*, no significant jitter absorption is necessary.

Finally, we note that in the construction of a periodic schedule, there exist several choices for optimization criteria, such as minimization of the server bandwidth, minimization of the client buffer size etc. The objective we will use is the minimization of overflow traffic beyond the link capacity, when VBR traffic segments are aggregated. However, by committing to the minimization of data loss, the schedule construction must still be able to capture all the other relevant constraints, i.e., the ones stemming from the synchronization between segments, and the hardware capabilities of the server and clients. How this is accomplished is explained in the next two sections.

## 4. Periodic Broadcast of VBR Video

We follow the definitions of [14] regarding the periodic broadcasting scheme for VBR traffic. Each video $m$ is divided into $K_m$ segments prior to broadcasting. The server broadcasts $\sum_{m=1}^{M} K_m$ video streams simultaneously, one per segment per video. Frames from $\sum_{m=1}^{M} K_m$ logical channels are multiplexed into the single broadcast link without buffering. Bits are lost whenever the aggregate broadcast traffic rate exceeds the link capacity. The server broadcasts each video segment at a rate $F$ frames per second, the consumption rate of the videos.

A user waits until the next starting point of the first segment. At the next broadcast of the first segment the user 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 [8, 7]. The choice of the download strategy depends crucially on the client bandwidth, on the number of channels, $C_m$, that can be simultaneously received by the client.

Part of the download strategy is that, if a segment size is larger than the segment following it, the next segment can always be downloaded to the disk in advance of being needed for playout, resulting in overhead in terms of disk space and disk bandwidth. In this sense, there exists an inherent conflict between start–up delay constraints and disk space or disk bandwidth constraints. If these constraints are tight, there may exist no schedule that simultaneously satisfies all of them.

### 4.1. Playout Latency

Next, we indicate how the playout latency depends on the selected broadcast series. Let $\{\ s_m^1, s_m^2, ..., s_m^{K_m}\ \}$ be a broadcast series for the $m^{th}$ video. Without any loss of generality, set $s_m^1 = 1$. Let $N_m^i$ indicate the number of frames in the $i^{th}$ segment of the $m^{th}$ video. The broadcast series implies that the segment sizes are:

$$N_m^i = s_m^i \times N_m^1, \quad i = 2, \ldots, K_m, m = 1, \ldots, M. \qquad (1)$$

The size of the first video segment is therefore determined by:

$$N_m^1 = \frac{N_m}{\sum_{i=1}^{K_m} s_m^i}. \qquad (2)$$

Given a particular fragmentation of the video, the playout latency, is bounded by the time it takes to start receiving the first segment of the video, which in turn is equal to the broadcast duration of the first segment. Let $D_m$ denote the service latency for the $m^{th}$ video. We have:

$$D_m = \frac{N_m^1}{F} \qquad (3)$$

In general, for a broadcast series $\{s_m^1,\ s_m^2, \ldots,\ s_m^{K_m}\}$ where $s_m^1 = 1$ the service latency is given by:

$$D_m = \frac{N_m}{(F \sum_{i=1}^{K_m} s_m^i)} \qquad (4)$$

According to our assumption, the fragmentation strategy must guarantee a playout latency of $w_m$ seconds maximum for the users before they can watch a video of their choice. From equation (4), we have:

$$D_m = \frac{N_m^1}{F} = \frac{N_m}{F \sum_{i=1}^{K_m} s_m^i} \leq w_m \qquad (5)$$

thus:

$$\sum_{i=1}^{K_m} s_m^i \geq \frac{N_m}{F w_m} \qquad (6)$$

Clearly, equation (6) provides a simple benchmark as to whether a broadcast series satisfies the playout latency constraint.

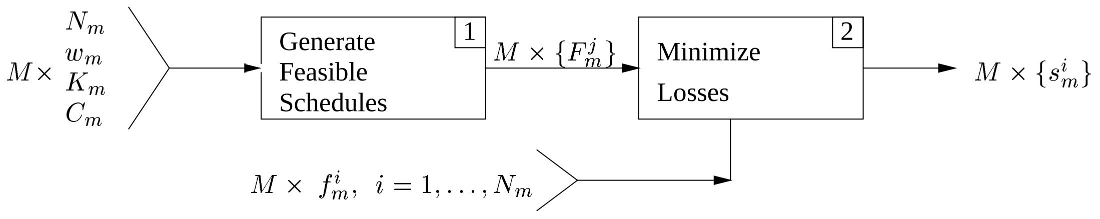

**Figure 2. Block diagram of TAF.**

### 4.2. Bufferless Multiplexing Data Loss

In the bufferless multiplexer 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. Let $A_{i,m}^t$ indicate the actual arrival bits sent by the stream of the $i^{th}$ segment of the $m^{th}$ video during frame time $t$. Let $A_t$ denote the total arrival bits sent by all of the $\sum_{m=1}^{M} K_m$ streams. Then, we have the following:

$$A_t = \sum_{m=1}^{M} \sum_{i=1}^{K_m} A_{i,m}^t \tag{7}$$

$$A_{i,m}^t = f_m^j \tag{8}$$

where j is given by:

$$j = \sum_{l=1}^{i-1} N_m^l + (t \bmod N_m^i) \tag{9}$$

Notably, $j$ stands for the index for the frame of the $m^{th}$ video that is sent during frame time $t$. Thus, loss occurs in frame time $t$ if:

$$A_t > B/F \tag{10}$$

Let $P_{loss}$ denote the long-run fraction of traffic loss, we have:

$$P_{loss} = \lim_{T\to\infty} \frac{\sum_{t=1}^{T} (A_t - B/F)^+}{\sum_{t=1}^{T} A_t} \tag{11}$$

Equation (11) provides the definition of the data loss which is the quantity we wish to minimize. Note that $P_{loss}$ is defined in terms of data loss in units of bits. However, we consider the bit loss ratio as a sufficiently accurate approximation for the packet loss ratio when the packets are small, e.g. if ATM cells were to be used as the underlying means of conveying the segments.

## 5. The Trace–Adaptive Fragmentation (TAF)

The Trace–Adaptive Fragmentation is summarized in the block diagram of Figure 2. In the first stage, a number of alternate fragmentations are found that satisfy the per-video delay constraints. In the second stage, one fragmentation is selected for each video in such a way that combined traffic of all the videos under the selected fragmentations minimizes the lost data. Note that in Figure 2 the input consists of the $C_m$, $K_m$, $w_m$ and $N_m$ parameters for each one of the $M$ videos. $N_m$ characterizes the corresponding video frame sequence but the rest ($C_m, K_m, w_m$) are all constraints of the VoD system. That is, at the first stage, the actual frame sequence plays no role, except through its total number of frames.

### 5.1. Continuity Constraint and Segment Sizes

The segments must be transmitted in such a relation to each other that the *continuity condition* holds. The continuity condition ensures that no starvation will occur at the client once it starts to consume the video stream and until the end of the video.

We know that the client can download the video data from $C_m (2 \le C_m \le K_m)$ streams simultaneously. If $C_m$ is equal to 1, then all of the segments have to be equally-sized in order to guarantee the continuity condition. Immediately after the client process begins to download the video segments, the user can start playing back the video at its normal consumption rate of $F$ frames per second in the order [1] $S_m^1 \bullet S_m^2 \bullet S_m^3 \ldots S_m^{K_m}$.

We follow a similar approach taken by CCA [7], that is, to receive and playback the data fragments, a client uses $C_m + 1$ service routines: $C_m$ data loaders, $L_1, L_2, \ldots L_{C_m}$, and one video player. A multi-threaded client multiplexes itself among these routines. Each data loader can download data at the consumption rate. The data segments are downloaded in $G_m$ rounds. During each round, each of the $C_m$ loaders is responsible for downloading its respective data segment in a certain transmission group, say the $r^{th}$ group, at its *earliest* occurrence. When the download of the current group has been completed, the loaders proceed to download the next transmission group, i.e., $(r + 1)^{th}$ group, in the same manner.

[1] Note that $S_m^i$ is notation representing a set: the frames of the $i$–th segment of the $m$–th video, while $s_m^i$ is an integer: the length of the $i$–th segment of the $m$–th video relative to segment $s_m^1 = 1$.

The solution we adopt operates by fragmenting each video file into $K_m$ segments and partitioning the $K_m$ segments further into $G_m$ transmission groups, where $G_m$ is given by $\lceil K_m/C_m \rceil$. Each group contains $C_m$ segments except for the last group that contains $K_m - (\lceil K_m/C_m \rceil - 1) \times C_m$ elements. Specifically, the $i$–th transmission group of the $m^{th}$ video contains the segments from $S_m^{(i-1)C_m+1}$ up to $S_m^{iC_m}$ (inclusive).

Given the above definitions, the segment sizes in each group possess the following properties:

1. a segment is greater or equal to the previous segment;
2. the size of the last segment of a group must be equal to the size of the first segment of the next group;
3. the size of the segment in each group is an integer multiple of the size of the first segment of the group.

We now introduce $X_m^i$ to indicate the maximum segment size for the $i^{th}$ segment of the $m^{th}$ video. $X_m^i$ represents the maximum allowable length for a segment in order to guarantee, that regardless when this segment starts relative to all the other segments of the same group, it can provide uninterrupted playout. The definition of $X_m^i$ relies on the knowledge of how much time is necessary for consuming all the segments the precede it in the same group:

$$X_m^i = \begin{cases} 1 & i = 1, \\ s_m^{i-1} & i \bmod C_m = 1, \\ s_m^{C_m \lfloor i/C_m \rfloor + 1} + & i \bmod C_m \neq 1. \\ \sum_{j=C_m \lfloor i/C_m \rfloor + 1}^{i-1} s_m^j & \end{cases} \qquad (12)$$

for $2 \leq i \leq K_m$. (Note that, $s_m^{C_m \lfloor i/C_m \rfloor + 1}$, represents the first segment of the group in which segment $i$ belongs.) In fact, the definition of $X_m^i$ restates the fragmentation principle of CCA [7] and its correctness can be proven using exactly the same arguments as for CCA. The difference is that the $X_m^i$ is only the upper bound on the length of a segment. The correctness (continuity) holds even if the segments are chosen to have length less than their corresponding $X_m^i$. In such case, starting to store the earliest occurrence of a segment may not be the most buffer–efficient approach, but we are not concerned with the size of secondary storage at the client. Overall, if $s_m^i \leq X_m^i$, each segment is guaranteed to have become available (started transmission) once or more times while the preceding segments were being consumed.

For $s_m^i = X_m^i$, the scheme specializes to CCA. If in addition $K_m = C_m$, then the scheme specializes to the the geometric broadcast series $\{1, 2, 4, \ldots\}$ which has been used in the VBR-B scheme [14]. Furthermore, as in SB, we can use a constant $W$ to restrict the size of the largest segment from becoming too large. $W$ solves the so-called *big-segment problem*. If a segment is larger than $W$ times the size of the first segment, we force it to be exactly $W$ units.

```
for $i = 1, \ldots, K_m$ do
    $s_m^i = 1$;
    calculate $X_m^i$ per equation (12);
endfor
while $s_m^2 \leq X_m^2$ do
        if equation (6) satisfied then
            output $s_m^i \forall i$ // Feasible fragmentation.
        endif
        $s_m^{K_m} = s_m^{K_m} + s_m^{C_m \lfloor K_m/C_m \rfloor + 1}$;
        for $i = K_m, \ldots, 2$ do
            if $s_m^i \geq X_m^i$ then
                if $(i-1) \bmod C_m = 1$ then
                    $s_m^{i-2} = s_m^{i-2} + s_m^{C_m \lfloor (i-2)/C_m \rfloor + 1}$;
                    $s_m^{i-1} = s_m^{i-2}$;
                else
                    $s_m^{i-1} = s_m^{i-1} + s_m^{C_m \lfloor (i-1)/C_m \rfloor + 1}$;
                    for $j = i, \ldots, K_m$ do
                        calculate $X_m^i$ per equation (12);
                        $s_m^j = s_m^{i-1}$;
                    endfor
                endif
            else
                break
            endif
        endfor
endwhile
```

**Figure 3. Pseudocode for the generation of all possible feasible broadcast series that guarantee the delay and continuity condition for video $m$.**

## 5.2. An Enumeration Algorithm

We present an enumeration algorithm which forms the first component of the TAF scheme (Figure 2). The input of the algorithm is parameters $K_m, C_m, N_m$ and $w_m$ and its output are all feasible broadcast schedules that conform to the playout latency and the continuity constraints. The enumeration starts with the base case where all segments are of equal length. Very likely, such a configuration will fail with respect to the delay constraints. The length of the segments is increased gradually starting from the last segment. If the last segment, even after increased to its maximum, cannot satisfy the delay constraint, an increase of the second segment from the end can start. The process continues until it

| $s_m^1$ | $s_m^2$ | $s_m^3$ | $s_m^4$ | $s_m^5$ | $s_m^6$ | Feasible? |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | No |
| 1 | 1 | 1 | 1 | 1 | 2 | No |
| 1 | 1 | 1 | 1 | 1 | 3 | No |
| 1 | 1 | 1 | 1 | 2 | 2 | No |
| 1 | 1 | 1 | 1 | 2 | 3 | No |
| 1 | 1 | 1 | 1 | 2 | 4 | No |
| 1 | 1 | 2 | 2 | 2 | 2 | No |
| 1 | 1 | 2 | 2 | 2 | 4 | No |
| 1 | 1 | 2 | 2 | 2 | 6 | No |
| 1 | 1 | 2 | 2 | 4 | 4 | No |
| 1 | 1 | 2 | 2 | 4 | 6 | No |
| 1 | 1 | 2 | 2 | 4 | 8 | No |
| 1 | 1 | 3 | 3 | 3 | 3 | No |
| 1 | 1 | 3 | 3 | 3 | 6 | No |
| 1 | 1 | 3 | 3 | 3 | 9 | No |
| 1 | 1 | 3 | 3 | 6 | 6 | No |
| 1 | 1 | 3 | 3 | 6 | 9 | No |
| 1 | 1 | 3 | 3 | 6 | 12 | No |
| 1 | 2 | 2 | 2 | 2 | 2 | No |
| 1 | 2 | 2 | 2 | 2 | 4 | No |
| 1 | 2 | 2 | 2 | 2 | 6 | No |
| 1 | 2 | 2 | 2 | 4 | 4 | No |
| 1 | 2 | 2 | 2 | 4 | 6 | No |
| 1 | 2 | 2 | 2 | 4 | 8 | No |
| 1 | 2 | 3 | 3 | 3 | 3 | No |
| 1 | 2 | 3 | 3 | 3 | 6 | No |
| 1 | 2 | 3 | 3 | 3 | 9 | No |
| 1 | 2 | 3 | 3 | 6 | 6 | No |
| 1 | 2 | 3 | 3 | 6 | 9 | No |
| 1 | 2 | 3 | 3 | 6 | 12 | Yes ($F_m^1$) |
| 1 | 2 | 4 | 4 | 4 | 4 | No |
| 1 | 2 | 4 | 4 | 4 | 8 | No |
| 1 | 2 | 4 | 4 | 4 | 12 | Yes ($F_m^2$) |
| 1 | 2 | 4 | 4 | 8 | 8 | Yes ($F_m^3$) |
| 1 | 2 | 4 | 4 | 8 | 12 | Yes ($F_m^4$) |
| 1 | 2 | 4 | 4 | 8 | 16 | Yes ($F_m^5$) |

**Figure 4. Example run of the enumeration algorithm, for $K_m = 6, C_m = 3, F = 25$ frames/s$, N_m = 40,000$ frames. $F_m^1$ to $F_m^5$ are the five feasible fragmentations that satisfy $w_m \leq 60$ sec.**

becomes necessary to increase $s_m^2$ to more than 2 (since, for $s_m^2 = 3 \geq X_m^2$ the continuity cannot hold).

The algorithm is presented in Figure 3. An example run of the algorithm is shown in Figure 4 where only five feasible solutions are found including the one that CCA constructs ($F_m^5$). We denote the set of feasible fragmentations of the $m$–th video as $\{F_m^j\}$. The generation of the feasible schedules (Figure 2) is performed independently for each one of the $M$ videos. Note that the set of feasible schedules is potentially large. From Figure 4 we also extract the instance (which fails the playout latency constraint) with broadcast series $\{1, 2, 3, 3, 6, 6\}$ and present it in Figure 5 to illustrate the temporal relation between the segments as constructed by the first stage of the TAF scheme.

### 5.3. Loss Minimization

The next step is the selection of the optimal broadcast schedule for each video, given the feasible schedules for $M$ videos (stage 2 of Figure 2). At this point, we can follow a computationally expensive process of determining the aggregate traffic of all possible combinations of feasible video schedules to determine the one that minimizes the loss rate. For $M$ videos, the combinations that need to be examined are:

$$\prod_{m=1}^{M} |\{F_m^i\}| \tag{13}$$

This approach requires extensive computation (depending on the set of feasible solutions), especially if the computation of the data loss for each combination is computationally expensive as well. Since all computation is performed off–line, one can claim that processing power is not going to be an issue. However, we can approximate the optimal selection by picking for each video the feasible schedule which results in the minimum peak rate. Thus, the aggregation of all the individual schedules is likely to produce traffic which has a small peak rate as well, and hence small loss ratio when multiplexed. In the presented experiments we use this particular approximation instead of the optimal solution.

At this point we must note that the computation of the data loss ratio of equation (11) can be simplified. Let:

$$\begin{aligned} \tilde{s_m} &= LCM\{s_m^1, s_m^2, \ldots, s_m^{K_m}\} & (14)\\ \tilde{s} &= LCM\{\tilde{s_1}, \tilde{s_2}, \ldots, \tilde{s_M}\} & (15)\\ \tilde{N} &= LCM\{N_1^1, N_2^1, \ldots, N_M^1\} & (16)\\ \tilde{T} &= \tilde{s}\tilde{N}. & (17) \end{aligned}$$

Due to the periodic nature of the broadcast strategy, the aggregate traffic of the $\sum_{m=1}^{M} K_m$ segments is also periodic with a period of $\tilde{T}$ frame times. To determine the data loss

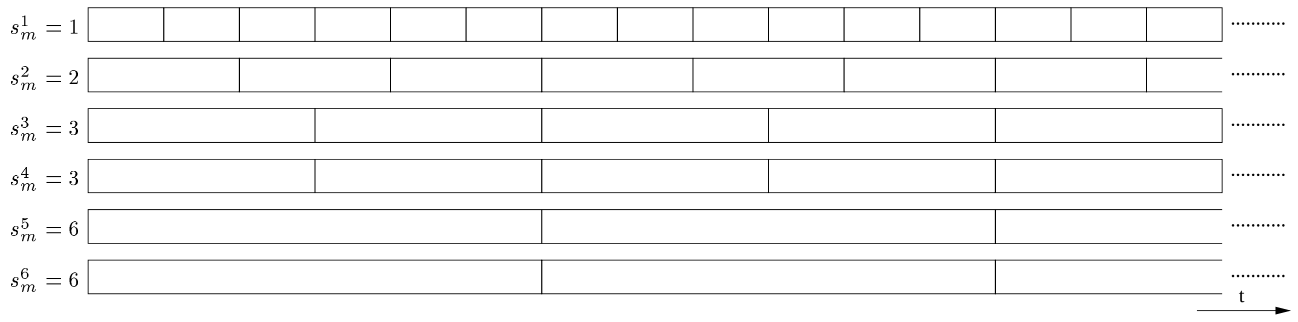

**Figure 5. Example timing of the segments of the broadcast series $\{1, 2, 3, 3, 6, 6\}$ generated for $C_m = 3, K_m = 6$.**

rate, we only need to observe the traffic during the first $\tilde{T}$ frame times. According to the relationship among successive segment sizes expressed in equation (1), we observe that:

$$\begin{aligned} LCM \quad & \{N_1^1, N_1^2, \ldots, N_1^{K_1}, \\ & N_2^1, N_2^2, \ldots, N_2^{K_2}, \ldots, \\ & N_M^1, N_M^2, \ldots, N_M^{K_M}\} = \\ LCM \quad & \{s_1^1 \times N_1^1, s_1^2 \times N_1^1, \ldots, s_1^{K_1} \times N_1^1, \\ & s_2^1 \times N_2^1, s_2^2 \times N_2^1, \ldots, s_2^{K_2} \times N_2^1, \ldots, \\ & s_M^1 \times N_M^1, s_M^2 \times N_M^1, \ldots, s_M^{K_M} \times N_M^1\} = \\ LCM \quad & \{\tilde{s_1} \times N_1^1, \tilde{s_2} \times N_2^1, \ldots, \tilde{s_M} \times N_M^1\} = \\ LCM \quad & \{\tilde{s_1}, \tilde{s_2}, \ldots, \tilde{s_M}\} \times \\ LCM \quad & \{N_1^1, N_2^1, \ldots, N_M^1\} = \\ \tilde{s}\tilde{N} \quad & = \tilde{T} \end{aligned} \tag{18}$$

Thus, to compare the packet loss rates of different fragmentations of the video files, we only need to calculate the packet loss rate in the first $\tilde{T}$ frame times. Hence, given a particular selection of broadcast series for each one of the $M$ videos we can produce an estimate of the data loss probability in time $O(\tilde{T})$.

In the calculation of the loss probability, the frame sizes of the video traces are necessary, thus the need to include the information about $f_m^i$ in the second stage of TAF (Figure 2). Therefore, the total space requirements for keeping track of the frame sizes if $O(\sum_{m=1}^{M} N_m)$. Note that CBR coded videos can be included trivially ($f_m^i = ct.$) allowing TAF to produce schedules for a mixture of VBR and CBR coded videos.

## 6. Experimental Results

The approximation introduced in the previous section selects for each video the feasible schedule that minimizes the peak bitrate of the aggregated bandwidth (over all the segments) for the specific video trace. The computational complexity of identifying the per–video peak bitrate for a given broadcast series is $O(LCM(N_m^1, N_m^2, \ldots, N_m^{K_m}))$, which using the notation of the previous section, becomes $O(N_m^1 \tilde{s_m})$.

Figure 6 provides the peak bitrate found for each one of 10 example video traces of 40,000 frames each (see [13] for the origin of the traces) depicting diverse material, including feature movies, TV news, sporting events etc. The set of feasible schedules explored were the ones produced in Figure 4. Observe that the peak bitrate depends drastically on the selection of the feasible broadcast schedule. For example, trace `talk_2` exhibits peak bitrates from a maximum of 12.26 Mb/s (for $F_m^4$) down to almost half the maximum at 6.19 Mb/s (for $F_m^3$). Notably, the broadcast series used by CCA, $F_m^5$, provides the minimum peak bitrate for only two of the ten examined traces.

The numbers in parentheses in Figure 6 are the client peak bandwidth requirements. That is, the peak of the aggregate traffic received by downloading simultaneously $C_m = 3$ segments of the same group. It should be pointed out that for VBR traffic, the peak experienced by a set-top box depends on the random instant at which it started downloading. In other words, a set-top box does not necessarily remain active downloading $C_m$ channels for $N_m^1 \tilde{s_m}$ frame times. In fact, the entire video can be completely downloaded in much lesser time. Thus, the reported peak bitrate for a client is a worst–case unrealistic scenario. Nevertheless, Figure 6 clearly illustrates that reduced client bandwidth is achievable by limiting the number of channels that can be simultaneously downloaded.

It must be emphasized that in the presentation of the experimental results we have taken a server–centric view. Hence the selection criterion we apply for a broadcast schedule is to exhibit the lowest peak bitrate. One can use a client-oriented view which encourages the selection of schedules that minimize the client download bandwidth. We can see from Figure 6 that doing so (thus selecting the schedules with the boldface parenthesized values) does not always result in low bandwidth demands for the server.

| Video | $F_m^1$ | | $F_m^2$ | | $F_m^3$ | | $F_m^4$ | | $F_m^5$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| mtv_1 | 14.84 | (11.89) | 10.14 | (8.05) | 10.15 | (8.05) | 16.48 | (10.09) | **10.02** | **(7.89)** |
| mtv_2 | 11.32 | (9.52) | **9.85** | (8.54) | **9.85** | (8.54) | 14.31 | (9.76) | 10.27 | **(7.08)** |
| race | 12.54 | (8.43) | 11.11 | **(7.07)** | **10.48** | **(7.07)** | 16.71 | (9.09) | 10.99 | (7.76) |
| talk_1 | 8.51 | (6.01) | 7.91 | (5.38) | 7.44 | **(4.91)** | 13.66 | (7.14) | **7.35** | (5.43) |
| talk_2 | 7.79 | (5.03) | 7.17 | (4.45) | **6.19** | **(4.40)** | 12.26 | (6.35) | 6.86 | (4.52) |
| simpsons | 10.13 | **(6.90)** | **9.59** | (6.92) | **9.59** | (6.92) | 14.61 | (8.14) | 10.10 | (7.25) |
| terminator | 5.41 | (3.50) | 4.49 | **(3.02)** | **4.14** | **(3.02)** | 7.89 | (4.57) | 5.01 | (3.21) |
| soccer_1 | 12.02 | (8.88) | 10.95 | (7.07) | **10.12** | (7.07) | 17.80 | (9.81) | 10.55 | **(6.67)** |
| soccer_2 | 9.99 | (8.35) | 10.27 | **(7.35)** | **9.97** | (7.36) | 15.19 | (9.23) | 13.73 | (8.76) |
| news_2 | 8.60 | (6.96) | 7.91 | **(5.71)** | **7.69** | **(5.71)** | 14.22 | (8.37) | 9.17 | (6.78) |

**Figure 6. Peak aggregate transmission (reception) bitrate in Mb/sec of the multiplexed periodic broadcast traffic for each feasible schedule of Figure 4 for 10 video traces taken from [13]. The numbers in boldface correspond to the minimum peak aggregate bitrate for each video over all the five feasible fragmentations. (**$w_m \leq 60$ **sec**, $C_m = 3, K_m = 6, N_m = 40,000$ **frames**, $F = 25$ **frames/sec).**

What is a good for the client set-top box is frequently not good for the server.

Another angle to view the same results is to consider the fact that given several feasible schedules, external factors can be incorporated in the selection of the best schedule. For example, in systems with heterogeneous set-top boxes, it is possible to determine which fragmentation is better used for which set-top boxes, possibly transmitting the same video in two or more different fragmentation formats, one for each set-top box technology.

## 6.1. Comparison of VBR-B and TAF

We conduct a comparison between TAF and VBR-B using the same set of 10 selected MPEG traces. However, to provide a basis for the comparison, no limit on the number of downloading channels is imposed ($C_m = K_m$). We are already aware that the search for feasible schedules of TAF subsumes the schedules produced for VBR-B (geometric series). At the same time, TAF uses an approximate search (instead of the exhaustive search implied by equation (13). Therefore, it is not at all clear whether the approximate TAF performs consistently better than VBR-B. For this reason, we perform a set of additional experiments comparing directly TAF and VBR-B.

The lower peak bitrate for TAF compared to that of VBR-B is readily confirmed in Figure 7. The differences are not always spectacular and they are sensitive to the particular video trace. However, once multiplexed on the same link, the modest differences in peak bitrate on a per-video basis compound resulting in distinctly different loss behavior as Figure 8 reveals. Clearly TAF provides a performance advantage in this environment.

| Video | VBR-B | TAF |
|---|---|---|
| mtv_1 | 11.70 | 9.86 |
| mtv_2 | 11.75 | 9.17 |
| race | 12.88 | 11.22 |
| talk_1 | 9.51 | 7.33 |
| talk_2 | 9.12 | 6.76 |
| simpsons | 11.41 | 9.06 |
| terminator | 5.90 | 4.55 |
| soccer_1 | 13.48 | 10.32 |
| soccer_2 | 12.76 | 10.44 |
| news_2 | 10.45 | 7.86 |

**Figure 7. Peak aggregate bitrate (in Mb/sec) of the multiplexed periodic broadcast traffic for VBR-B and TAF for 10 video traces taken from [13] (**$w_m \leq 16.5$ **sec**, $C_m = K_m = 7, N_m = 40,000$ **frames**, $F = 25$ **frames/sec).**

To determine the effect of server bandwidth on performance (Figure 8), we varied the link capacity $B$ between 50 Mb/sec and 85 Mb/sec while the maximum playout latency, $w_m$, was set to a borderline small value of 16.5 seconds. The playout latency pushed the scheduling to its limits forcing particularly small first segments and much lengthier subsequent ones. Because the number of segments was small, $K_m = 7$, the limit value $W = 32$ was used to force VBR-B to not create too large segments and to match exactly the playout latency objective. A similar intervention on TAF did not prove necessary since it identified several feasible schedules satisfying (or even exceeding) the requirements of the playout latency constraint.

Similar to bufferless multiplexing, we can compare VBR-B to TAF assuming buffered multiplexing at the server. That is, all streams sent to the clients are fed to a common buffer before transmission by the server. The results for this case are shown in 9. It is clear that for large enough buffer size no appreciable loss is present, but at the same time the arrival jitter at the clients can be increased. Thus, we provide this example only to point out the flexibility of TAF which is in all aspects similar to the flexibility of VBR-B while providing better loss rate results. Note that Figures 8 to 11 are plotted using log–scale for the y–axis to capture the wide range of values over which the loss probability spans for even small changes in system parameters (e.g. link capacity $B$). Hence, seemingly small differences in Figure 9 between TAF and VBR-B are quite substantial in terms of absolute numbers.

The Join-the-Shortest Queue prefetching scheme by Reisslein and Ross in [12] can also be used by the server to force the clients from any of the ongoing video streams to prefetch video frames and to send the prefetched frames to the buffers in the appropriate clients to fully utilize the shared links's bandwidth (when the link is idling due to the VBR nature of the multiplexed video segments). It is assumed that each stream has a virtual buffer and the one with the shortest queue has the highest priority to prefetch one more frame if the aggregated bandwidth is not currently over the shared links' capacity. We refer the readers to [12] for the details of the JSQ prefetching policy. Figure 10 presents the results produced by TAF and VBR-B with the addition of the Join the Shortest Queue (JSQ) policy. Even in this case TAF has an advantage although its potential for improvement is reduced w/ increasing virtual buffer size. This is partly because of the more "compact" aggregate traffic in the case of TAF compared to VBR-B. That is, by avoiding extreme values in the required aggregate bandwidth, TAF presents less opportunities to prefetch using JSQ.

A final extension of the basic scheme is the inclusion of Group of Picture (GOP) smoothing to the broadcast schedule computation. Smoothing drastically reduces the variance of the aggregate bandwidth, and this in turn reduces the differences of the peak bitrate of the different candidate fragmentations for TAF. Thus the selection of the minimum peak bitrate becomes less consequential to the loss rate performance (Figure 11). Remarkably, even under settings that smooth and "reshape" the traffic (such as buffered multiplexing, JSQ–prefetch and GOP–smoothing) TAF still provides a consistently better performance than VBR-B.

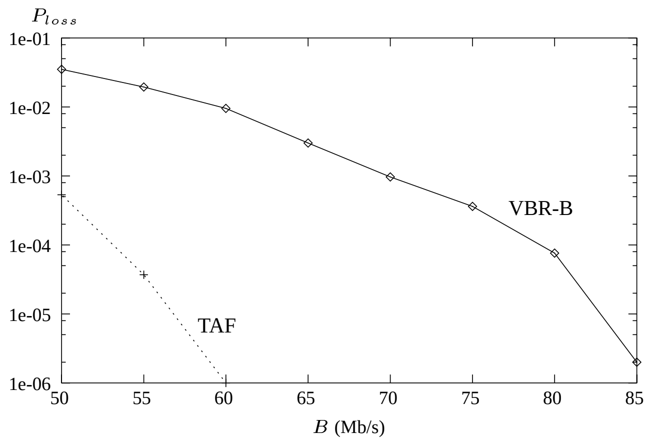

**Figure 8. Comparison of the loss rate for variable $B$ between VBR-B and TAF. The 10 videos of Figure 7 were used. ($w_m \leq 16.5$ sec$, C_m = K_m = 7, N_m = 40,000$ frames$, F = 25$ frames/sec).**

## 7. Conclusions and Future Directions

In this paper we consider the problem of periodic broadcast of VBR video for VoD systems. Given the a–priori knowledge of the entire traffic and given the system objectives we have approached the problem as essentially a *deterministic* problem, for which choice of an "optimal" broadcast schedule may be possible. In order to cope with the many and conflicting performance objectives we have focused on the construction of broadcast schedules that satisfy playout latency constraints and enforce a maximum number of server and client channels. An enumerative process identifies broadcast schedules that satisfy the constraints. Subsequently, the selection of a particular schedule is the result of an optimization process which accounts for the VBR nature of the transmitted video segments. The optimization process appears currently to be computationally expensive, and, as a result, we propose a fast approximation which has been shown to outperform existing schemes in a variety of settings.

The lesson learned is that the rigid fragmentation schemes used in the past may not be the best choice given

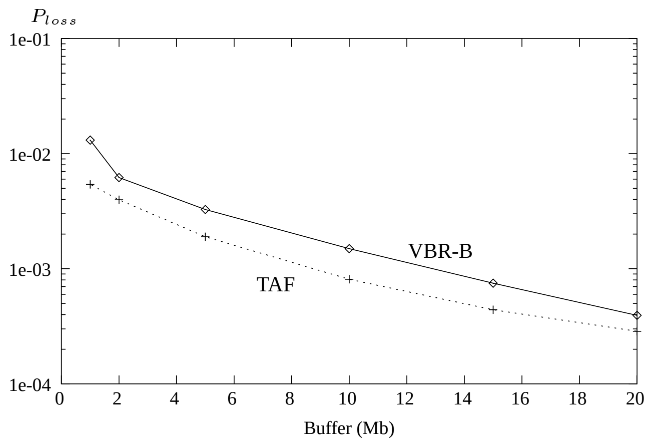

**Figure 9. Comparison of loss rate between VBR-B and TAF using buffered multiplexing for variable shared buffer size at the server. The 10 videos of Figure 7 were used. ($w_m \leq 16.5$ sec$, C_m = K_m = 7, N_m = 40,000$ frames$, F = 25$ frames/sec$, B = 38$ Mb/sec).**

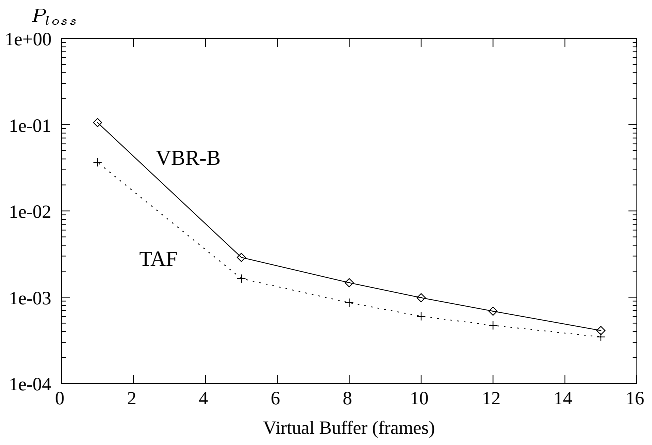

**Figure 10. Comparison of loss rate for variable virtual buffer size between VBR-B and TAF using JSQ prefetching for the allocation of the spare bandwidth after multiplexing. The 10 videos of Figure 7 were used. ($w_m \leq 16.5$ sec$, C_m = K_m = 7, N_m = 40,000$ frames$, F = 25$ frames/sec$, B = 38$ Mb/sec).**

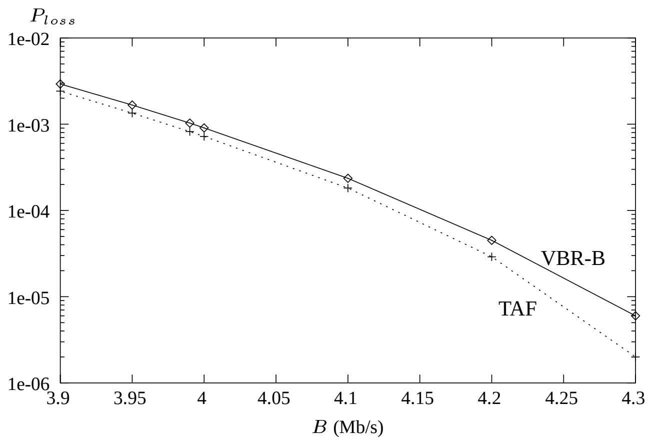

**Figure 11. Comparison of loss rate for variable $B$ between VBR-B and TAF using GOP–smoothing over periods equal to a single GOP (12 frames). The 10 videos of Figure 7 were used. ($w_m \leq 16.5$ sec$, C_m = K_m = 7, N_m = 40,000$ frames$, F = 25$ frames/sec).**

the high variability of VBR traffic. Future research in this direction will focus on the reformulation of the optimization problem in a manner that will allow better insight in terms of a fast solution technique. In the meantime, several heuristics will be tried for the selection of the best fragmentation such as heuristics for minimizing the variability of the aggregated segments. We are also exploring techniques to support interactive VCR–like operations and to support potentially heterogeneous clients/set–top boxes.

## 8. Acknowledgements

This work was supported in part by NSERC under grant OGP 0194424.

## References

[1] C. Aggarwal, J. Wolf, and P. Yu. On optimal batching policies for video-on-demand storage servers. In *Proc. of IEEE Intl. Conf. on Multimedia Systems '96*, 1996.

[2] C. Aggarwal, J. Wolf, and P. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. In *Proc. of IEEE Intl. Conf. on Multimedia Systems '96*, 1996.

[3] K. Almeroth and M. Ammar. The use of multicast delivery to provide a scalable and interactive video-on-demand service. *IEEE JSAC*, August 1996.

[4] M. Ammar and J. W. Wong. The design of teletext broadcast cycles. *Performance Evaluation*, 5(4):235–242, 1985.

[5] I. Dalgic and F. Tobagi. 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.

[6] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proc. of ACM Multimedia '94*, pages 15–23, October 1994.

[7] K. Hua, Y. Cai, and S. Sheu. Exploiting client bandwidth for more efficient video broadcast. In *Proc. of IEEE ICCCN '98*, 1998.

[8] K. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In *Proc. of ACM SIGCOMM '97*, pages 89–100, 1997.

[9] L. Juhn and L. Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Trans. Broadcasting*, 43(3):268–271, September 1997.

[10] J. Paris, S. Carter, and D. Long. Efficient broadcasting protocols for video on demand. In *Proc. of MASCOTS '98*, pages 127–132, July 1998.

[11] J. Paris, S. Carter, and D. Long. A low bandwidth broadcasting protocol for video on demand. In *Proc. IEEE ICCCN '98*, 1998.

[12] M. Reisslein and K. Ross. A join-the-shortest-queue prefetching protocol for vbr video on demand. In *Proc. IEEE ICNP '97*, October 1997.

[13] O. Rose. Statistical properties of mpeg video traffic and their impact on traffic modelling in atm systems. Technical Report Technical Report 101, University of Wuerzburg, Germany, February 1995.

[14] D. Saparilla, K. Ross, and M. Reisslein. Periodic broadcasing with vbr-encoded video. In *Proc. IEEE INFOCOM '99*, 1999.

[15] S. Viswanathan and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *Multimedia Systems*, 4(4):197–208, August 1996.