

# Lossless VBR Video Broadcasting with User Bandwidth Limit using Uniform Channels

Shufang Wu and Tiko Kameda  
School of Computing Science, CMPT-TR 2003-08  
Simon Fraser University  
Burnaby, B.C., Canada V5A 1S6  
{vswu, tiko}@cs.sfu.ca

*Abstract*—This paper proposes a new Video-On-Demand (VOD) broadcasting scheme for variable bit rate (VBR) encoded videos, called Forward Segmentation with Equal Bandwidth (FSEB). For all practical purposes, FSEB solves the elusive problem of minimizing the server bandwidth for delivering VBR videos with no loss, given an upper bound on the wait time, a uniform upper bound on the channel bandwidths, and the number of channels that a user can access simultaneously. Like many other broadcasting schemes, it divides a video into segments, and broadcasts each segment periodically on a separate logical channel. The main difficulty in solving this problem is the fact that VBR frames have irregular sizes and it is difficult to fit whole frames into channels of fixed bandwidth  $c$ . The only other lossless scheme for VBR videos currently known that can limit the number of channels that a user can access simultaneously is StairCase Broadcasting (SCB), which is a heuristic. Experimental results with real videos reveal that FSEB in general requires less server bandwidth and user bandwidth than SCB.

*Keywords*—Video-on-demand; VBR encoded videos; lossless broadcasting; user bandwidth limit; optimization

## I. INTRODUCTION

Broadcasting [25] has been shown to be a bandwidth-efficient way to transport frequently requested videos in Video-On-Demand (VOD) systems. **Staggered Broadcasting** [2] is the earliest and perhaps the most natural broadcasting scheme, but its performance has been exceeded by more recent schemes based on segmentation. Performance is commonly measured by the **start-up latency** (i.e., initial user **wait time**) for given server bandwidth or, equivalently, the required server bandwidth for given start-up latency. The segmentation-based broadcasting schemes partition a video into segments and repeatedly broadcast each segment through a dedicated logical channel.

Since the pioneering work by Viswanathan and Imielinski [23] in 1995, many segmentation-based schemes for **constant bit rate (CBR)**-encoded videos with varying performance have been proposed. All the existing schemes for CBR videos can be classified into three groups. The first group consists of schemes that use uniform channels and segments of different sizes. **Pyramid Broadcasting (PB)** [23,24], **Permutation-**

**based Pyramid Broadcasting (PPB)** [1], **Skyscraper Broadcasting (SB)** [3] and **Greedy Equal Bandwidth Broadcasting (GEBB)** [6] belong to this group. The second group uses equal segment size and non-uniform channels. It includes **Harmonic Broadcasting (HB)** [7], **Cautious Harmonic Broadcasting (CHB)**, **Quasi-Harmonic Broadcasting (QHB)** [15] and **Poly-Harmonic Broadcasting (PHB)** [16]. There are other schemes whose group membership may not be clear at first glance. **Fast Broadcasting (FB)** [8], for example, refers to equal-sized blocks as “segments”, but several consecutive blocks form a **segment** in our terminology, and segments are broadcast on uniform channels. Therefore, FB belongs to the first class.

**Pagoda Broadcasting (PaB)** [17], **New Pagoda Broadcasting (NPB)** [13] and **Fixed-Delay Pagoda Broadcasting (FDPB)** [14] all ostensibly use “channels” of equal bandwidth and “segments” of equal size. At closer examination, however, “segments” refer to equal-sized blocks, and each “subchannel” repeatedly broadcasts a sequence of consecutive blocks. Such a sequence is really a segment in our terminology. A “channel” is divided into a set of “subchannels”. Therefore, in our terminology, a “subchannel” is really a (logical) channel, and a “channel” transmits a sequence of interleaved segments. In summary, these schemes all use channels of unequal bandwidths and segments of unequal sizes, and they belong to the third group.

For a brief review on the above schemes, the reader is referred to an insightful survey [5].

Several other schemes take the bandwidth capability of the user into consideration. **Client-Centric Approach (CCA)** [4] leverages the bandwidth of the user to reduce the server bandwidth. **Generalized Fibonacci Broadcasting (GFB)** [27] achieves the best performance among the known schemes, given the user bandwidth limit and the number of segments a video is divided into.

Some important concepts have been developed in CBR schemes. One of them is the **greedy** (or **fixed-delay**) approach. In greedy schemes [6,14,16], the user starts downloading immediately after tune-in. Another concept is **zero-delay** broadcasting (using “partial preloading” or **prefix-caching**) proposed in [18,19].

These schemes require the user to download the initial segments of the most popular movies, so that display can be started without any delay at all.

All the above schemes are designed for CBR videos. But in practice, videos are commonly compressed and **variable bit rate (VBR)**-encoded. In order to broadcast VBR-encoded videos, two approaches have been investigated in the past. One is to allow occasional data losses, provided they are infrequent enough not to disturb viewing very much. The other approach, which we pursue in this paper is to design a scheme which guarantees no data loss.

In 1999, Saporilla et al. proposed a scheme (**VBR-B**) [21] based on the segmentation of FB [8], using **Group of Pictures (GoP) smoothing**, server buffering and client **prefetching**. **Trace-Adaptive Fragmentation (TAF)** [9] combines this idea with CCA [4] and uses the video trace to achieve less data loss.

Unlike VBR-B and TAF, P aris proposed a lossless scheme called **Variable Bandwidth Harmonic Broadcasting (VBHB)** [12] based on CHB [15], which they had proposed earlier for CBR videos. In VBHB, the first segment is broadcast concurrently with its display, while the other segments are completely downloaded before their display starts. The first channel uses high enough bandwidth (higher than the average for the first segment) to guarantee no loss. The extra data it transmits (because it has higher-than-average bandwidth) is deducted from the second segment. So the second segment is smaller and the bandwidth of the second channel is adjusted accordingly.

A real video consists of a sequence of **frames**. A segment cannot be displayed entirely if it contains a fraction of a frame. Several schemes that respect frame boundaries have been proposed.

**Loss-Less and Bandwidth-Efficient (LLBE)** [11] addresses the issue of minimizing the server bandwidth for real VBR videos for given wait time and number of segments. LLBE formulates this problem as the shortest path problem in a directed acyclic graph. The disadvantage of LLBE is that its solution requires channels with irregular bandwidths, which would make its implementation difficult. **StairCase Broadcast (SCB)** [10] addresses the same issue with the additional constraint that the number of user channels is also prespecified. SCB is a heuristic for minimizing both server and user bandwidth. In particular, the bandwidth of each channel is an output from SCB, and it cannot be used to minimize the server bandwidth for a given user bandwidth limit.

To solve this problem, in this paper we propose a new broadcast scheme, named **Forward Segmentation with Equal Bandwidth (FSEB)**. To attack the problem from a different angle, FSEB uses uniform channels off the butt by giving the upper bound on channel bandwidth  $c$  (bits/sec) as an input. The other inputs are the wait time  $w$  (sec) and the number of channels  $K$  that

the user can access simultaneously, as well as the trace (containing the frame sizes) of a video. FSEB systematically segments the frames into  $n$  segments, where  $n$  is one of the outputs. If the last (i.e., the  $n$ -th) channel is not fully utilized, then  $w$  or  $c$  can be decreased slightly until it is.

FSEB gives implementers fine-grained control via the input parameters  $w$  (sec) and  $c$  (bits/sec) and  $K$ . As with GEBB [6] and GFB [27], for fixed wait time  $w$  and user bandwidth  $B_u$ , a smaller channel bandwidth  $c$  in general leads to smaller server bandwidth  $B_s$ . However, there are practical lower limit on  $c$  and practical upper limit on  $K$  and  $n$ .

The remainder of the paper is organized as follows. In Section 2, we review SCB in detail and point out the problems with it. In Section 3, our FSEB scheme is introduced and explored. In Section 4, we compare the performance of FSEB with that of SCB and GFB. Finally, Section 5 contains the conclusion and possible future work.

## II. RELATED WORK

To the best of our knowledge, only SCB deals with the server bandwidth optimization with user bandwidth limit for VBR videos respecting frame boundaries. Here, we first review some ideas used in SCB. For details, please see [10].

Given a wait time  $w$  (sec), the number of channels  $K$  through which the user can download simultaneously and the number of segments  $n$  into which the video is divided, the minimum required bandwidth  $b_i$  of channel  $i$  can be expressed as follows:

$$b_i = \begin{cases} \frac{|S_i|}{w + \frac{X(i-1)}{F}}, & 1 \leq i \leq K \\ \frac{|S_i|}{\frac{X(i-1) - X(i-K-1)}{F}}, & K \leq i \leq n \end{cases} \quad (1)$$

where  $|S_i|$  denotes the size in bits of the  $i$ -th segment, and  $X(i)$  denotes the index of the last frame in the  $i$ -th segment.

So the server bandwidth (i.e., the total bandwidth required to transmit the  $n$  segments)  $B_s$  is given by:

$$B_s = \sum_{i=1}^n b_i \quad (2)$$

The user bandwidth  $B_u$  (maximum bandwidth to download  $K$  consecutive segments simultaneously) is:

$$B_u = \max_{1 \leq i \leq (n-K+1)} \sum_{j=i}^{i+K-1} b_j \quad (3)$$

The key issue is how to divide the whole video into  $n$  segments. There are two optimization problems. One is to find the segmentation to minimize the server bandwidth given the user bandwidth limit, which can be stated as:

(A) Solve (1) such that (2) is minimized and (3) is no larger than the given user bandwidth.

The other is to find the segmentation so that the user bandwidth is minimized given the server bandwidth, which can be stated as:

(B) Solve (1) such that (3) is minimized and (2) is no larger than the given value of the server bandwidth.

The authors of [10] observe that “most likely the closer the necessary bandwidth for each segment, the lower the sum for any  $K$  consecutive segments and the lower for the total necessary bandwidth for all the segments”. Based on this observation a heuristic algorithm is proposed in [10] to merge adjacent segments. Initially, there are  $N$  segments (each segment consists of just one frame). In each round, among all the adjacent pairs of segments, the pair that would result in the smallest difference among the bandwidths required to download them is chosen and merged. Since  $N-n$  merges need to be carried out, so that  $n$  segments remain, the time complexity of the heuristic is  $O(N^2)$ .

FSEB also makes use of the above observation, but in a different way. As stated earlier, its inputs are the channel bandwidth  $c$  (bits/sec), the wait time  $w$  (sec) and the number  $K$  of channels that the user can access simultaneously, as well as the trace (containing the frame sizes) of a video. A **round** of FSEB algorithm computes in linear time the  $n$  segments for the video such that each segment can be downloaded in time for display through a channel with bandwidth  $c$ . It tries to “fill” the first channel by packing as many initial frames as possible without exceeding its bandwidth, and so on, down to the last channel. Therefore, it is likely that the last or the  $n^{\text{th}}$  channel will be only partially filled. If that is the case, we decrease  $w$  or  $c$  in increments, until there is no space left in the last channel. Each of these increments incurs a new round of execution. Fortunately, it turns out that the last channel gets filled rather quickly within a small number of rounds if we choose one second as the incremental step for  $w$ . If we need to adjust  $w$  or  $c$  to make best use of the last channel, the initial input  $w$  or  $c$  can be considered as an upper bound on the actual wait time or channel bandwidth.

### III. FORWARD SEGMENTATION WITH EQUAL BANDWIDTH WITH CLIENT BANDWIDTH LIMIT

#### A. Notation

To facilitate our discussion, we use the following notation:

- $N$ — Total number of frames in a video;
- $F$ — Display rate of the video in frames per second;
- $F_i$ —  $i$ -th frame of the video ( $i = 1, \dots, N$ );
- $f(i)$ — Size in bits of  $F_i$  ( $i = 1, \dots, N$ );
- $A(i)$ — Size in bits of the first  $i$  frames ( $i = 1, \dots, N$ );
- $n$ — Number of segments = number of channels used by the server ( $n \leq N$ );
- $K$ — Maximum number of channels from which user can download the video simultaneously ( $K \leq n$ );
- $S_j$ — The  $j$ -th segment of the video ( $j = 1, \dots, n$ );
- $X(j)$ — Index of the last frame in segment  $S_j$  ( $j = 1, \dots, n$ );
- $c$ — Bandwidth in bits per second of each channel;
- $w$ — User wait time in seconds;
- $B_s$ — Total server bandwidth in bits per second;
- $B_u$ — Maximum user bandwidth in bits per second;
- Initial values:  $A(0) = 0$  and  $X(0) = 0$ .

#### B. Segmentation

Given  $w$ ,  $c$  and  $K$  as inputs,  $n$  and  $X(j)$  ( $j = 1, \dots, n$ ) need to be computed. Then we can get  $B_s = n \times c$ . Note that  $B_u$  can be obtained directly from the inputs by  $B_u = K \times c$ . The detailed procedure is given below.

First of all, let us see how to determine the first segment  $S_1$  by computing  $X(1)$ . To ensure that  $S_1$  can be displayed after a delay of  $w$  (sec), the  $X(1)$  frames in  $S_1$  have to be downloaded in  $w$  (sec). Thus we have

$$\frac{\text{Size in bits of } S_1}{\text{Download time in seconds of } S_1} \leq c,$$

in other words,

$$\frac{A(X(1))}{w} \leq c,$$

For uniformity we rewrite this as:

$$\frac{A(X(1)) - A(X(0))}{w + \frac{X(0)}{F}} \leq c.$$

From the above inequality, we can determine  $X(1)$  as the maximum argument of the first  $A(\cdot)$  on the left-hand side.

In order to display the video continuously, the download time of  $S_2$  must not be larger than  $w$  plus the display time of  $S_1$ , which is  $X(1)/F$ . Since the size in bits of  $S_2$  is  $(A(X(2)) - A(X(1)))$ , from

$$3 \frac{\text{Size in bits of } S_2}{\text{Download time in seconds of } S_2} \leq c,$$

we get

$$\frac{A(X(2)) - A(X(1))}{w + \frac{X(1)}{F}} \leq c.$$

$X(2)$  can be calculated as the maximum argument of the first  $A()$  on the left-hand side.

Similarly, to ensure uninterrupted display of the video once it starts, the download time of  $S_j$  must not be larger than  $w$  plus the time to display the previous  $j-1$  segments. So from

$$\frac{\text{Size in bits of } S_j}{\text{Download time in seconds of } S_j} \leq c,$$

we get

$$\frac{A(X(j)) - A(X(j-1))}{w + \frac{X(j-1)}{F}} \leq c, \quad 1 \leq j \leq K. \quad (4)$$

$X(j)$  for  $1 \leq j \leq K$  can be computed as above, since the user can download from up to  $K$  channels simultaneously.

As for  $S_j$  for  $K < j \leq n$ , once the user finishes downloading the  $(j-K)$ -th segment, it can start downloading  $S_j$  immediately. Also, in order to play the video continuously, the user has to finish downloading  $S_j$  by the time the display of the  $(j-1)$ -st segment completes. So, the download time for  $S_j$  is the display time of segments  $S_{j-K}, \dots, S_{j-1}$ , which is the number of frames in those  $K$  previous consecutive segments divided by the display rate. We thus have

$$\frac{A(X(j)) - A(X(j-1))}{\frac{X(j-1) - X(j-K-1)}{F}} \leq c, \quad K < j \leq n. \quad (5)$$

We now present the computation of  $X(j)$  for  $1 \leq j \leq n$  in the form of pseudocode as follows:

```
//Compute A(i):
//Size in bits of the first i
//frames (i=1,...,N)
A(0) = 0;
for ( i = 1; i ≤ N; i ++ )
    A(i) = A(i-1) + f(i);

// Segmentation
i = 1;
j = 0;
X(0) = 0;
```

```
while ( i ≤ N ) {
    j = j + 1;
    if ( j ≤ K ) {
        while (inequality (4) is true) {
            i = i + 1;
            if ( i > N ) break;
        }
        X(j) = i - 1;
    } else {
        while (inequality (5) is true) {
            i = i + 1;
            if ( i > N ) break;
        }
        X(j) = i - 1;
    }
}

// Get n: Number of segments
n = j;
```

Figure 1. Pseudocode of FSEB

It is easy to see that the time complexity of the above algorithm is  $O(N)$ .

### C. Minimum User Bandwidth for VBR Videos

When we construct  $S_j$  using (4) or (5), it could happen that even the first frame after  $X(j-1)$  may not satisfy the relevant inequality. In other words,  $X(j)=X(j-1)+1$  fails to satisfy it. Segmentation fails in this situation. What we could do is to increase  $c$  or  $K$  and try again. Increasing  $c$  or  $K$  implies increasing  $B_u$ . From (4), we can see that there must be a minimum channel bandwidth to avoid unsuccessful segmentation of a video for a given wait time. From (5), we can also see that there must be a minimum number of user channels to avoid unsuccessful segmentation of a video for a given channel bandwidth  $c$ . These minimum values depend on the frame sequence of the video as well as on  $w$  and/or  $c$ .

With this observation, we modify the segmentation part in the previous algorithm as in Figure 2:

```
// Segmentation
i = 1;
j = 0;
X(0) = 0;

while ( i ≤ N ) {
    // Same as the outside while-loop
    // in Figure 1
    if ( X(j) == X(j-1) ) {
        j = j - 1;
        break;
        // Segmentation fails
    }
}
```

Figure 2. Enhanced segmentation part of FSEB pseudocode

#### D. Wasted Bandwidth

If inequality (4) or (5) is a strict inequality for  $X(j)$ , then there is some residual bandwidth in the  $j$ -th channel that is not fully utilized. This is the cost that we pay because the frame boundaries must be respected. For almost all the real videos that we have tested, however, this waste is less than 1% of the bandwidth  $c$ . The last channel is an exception, since there may be only a few frames left to be put in the last segment, and only a small fraction of  $c$  may be needed. If that is the case, we can decrease  $w$  or  $K$  or even  $c$  to fully utilize it.

Figure 3 plots the server bandwidth  $B_s$  against the wait time  $w$  when  $K$  and  $c$  are fixed at 19 and 64Kbps, respectively, for a real MPEG-1 video trace (BOND) from [20], whose properties are summarized in Table I. The figure shows that the server bandwidth in general decreases when we increase the wait time with user bandwidth fixed. Note that if we fix  $c$  and  $K$ , we should get the same  $n$  for a range of  $w$ . This explains why the graph has the form of a staircase. The height of each step equals  $c$ . The left end of each step corresponds to the case where the last channel is fully used. Therefore, in a sense, such a point represents an “optimal” selection of wait time  $w$ .

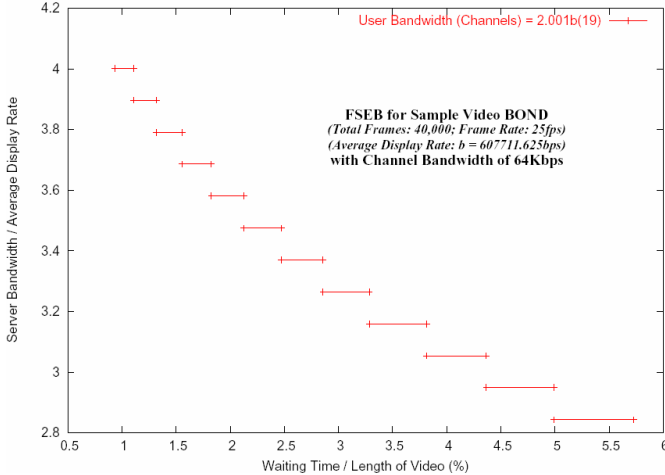


Figure 3. Normalized server bandwidth vs. normalized wait time for FSEB

TABLE I. PROPERTIES OF SAMPLE VIDEO BOND

Size(MB)	Average Display Rate(bps)	$N$	$F$ (fps)
115.912	607,711.625	40,000	25

In order to use standard channels within the user channel and bandwidth constraints, we adjust the wait time to make best use of the last channel. We decrease the wait time by one second at a time, since it is both practical (in terms of computation time) and accurate enough (one second is  $1/7200$  of a two hour video and  $1/1600$  of our example video BOND used later). The run time would be no larger than 10 times that of FSEB if

normalized wait time is no larger than 5%. So we will be able to find a value of  $w$  that is within one second of an “optimal” value. Since each round of FSEB takes only milliseconds to execute on a computer with a Pentium 4 processor, such an “optimal” can be found within a small fraction of one second. If we decrease the wait time by one millisecond at a time to increase the accuracy, we should use binary search instead of linear search as suggested above. In this paper, only Figure 3 was plotted using one-millisecond decrements for the wait time, while others were plotted using one-second decrements.

Table II lists some user bandwidth values for different combinations of  $c$  and  $K$  when the normalized wait time is fixed at 0.01 (or 16 seconds). In the last row,  $c$  is 4687 (bps), which is the minimum channel bandwidth for which FSEB can be applied to BOND without segmentation failure. Actually it is the minimum value so that the first frame can be put in the first segment. In other words, for  $f(1) = 74984$  and  $w = 16$  to satisfy (4) with  $j = 1$ , channel bandwidth  $c$  has to be at least 4687. The second column gives the minimum number of user channels for the given channel bandwidth in column 1. The product of the values in column 1 and column 2 divided by the average display rate is the value of normalized user bandwidth in column 3. From Table II, we can see that the minimum user bandwidth is about the average display rate of the sample video BOND. This verifies the correctness and necessity to consider the minimum user bandwidth in the subsection C.

TABLE II. USER BANDWIDTHS OF SAMPLE VIDEO BOND

$C$ (bps)	Minimum $K$	Normalized User Bandwidth
64000	11	1.158
32000	20	1.053
16000	40	1.053
12000	53	1.047
8000	79	1.040
4687	136	1.049

#### E. Adjusting Input Parameters $K$ and $c$

Sometimes, we only want to impose the maximum user bandwidth  $B_u$  and we can adjust  $K$  and  $c$  within the constraint of  $K \times c \leq B_u$ . Decreasing (increasing)  $c$  with fixed  $w$  would increase (decrease) both  $K$  and  $n$ . But decreasing  $c$  would have the beneficial effect of delaying the downloading of later frames, thereby decreasing the total server bandwidth for a fixed wait time.

Figure 4 shows the results of varying  $c$  and  $K$  within the constraint of  $K \times c = 2.001b$  for video BOND, where  $b$  is its average display rate. It can be seen that for a

fixed wait time, a smaller  $c$  leads to smaller server bandwidth.

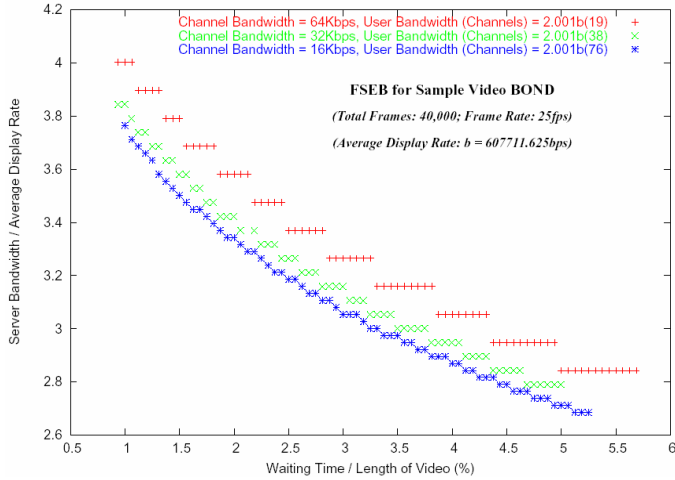


Figure 4. Normalized server bandwidth vs. normalized wait time for FSEB with the same user bandwidth constraint and different channel bandwidth

If we fix  $w$  and  $c$  for a video, there is the minimum  $K$  that prevents segmentation failure, as we commented earlier. Increasing  $K$  can decrease server bandwidth so that we can trade the user side bandwidth to save server bandwidth to a certain extent (similar to the idea in CCA [4]). Figure 5 shows the server bandwidth as a result of changing  $K$  for video BOND when  $w = 16$  (sec) and  $c = 16000$  (bps). It demonstrates that the server bandwidth can be saved by increasing the user bandwidth.

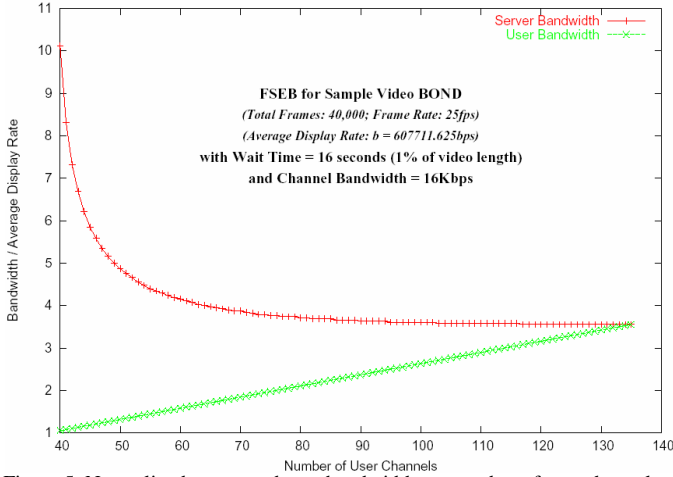


Figure 5. Normalized server and user bandwidth vs. number of user channels

Another way to save server bandwidth by exploiting the user bandwidth is to increase  $c$  for fixed  $w$  and  $K$ . Figure 6 shows the server bandwidth as a result of changing  $c$  from 8Kbps to 18Kbps for video BOND, when  $w = 16$  (sec) and  $K = 80$ . It demonstrates that the server bandwidth can also be reduced by increasing channel bandwidth with a given number of user channels.

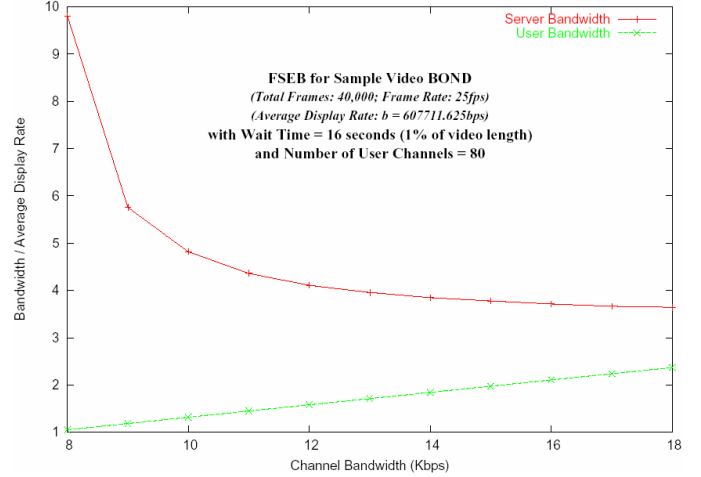


Figure 6. Normalized server and user bandwidth vs. channel bandwidth

Because of the small amount of bandwidth wasted in each channel, especially the last channel, if we don't adjust  $w$  to achieve optimal value, the monotonically decreasing trend of server bandwidth shown in Figure 6 will only be true for relatively small  $c$  (no larger than 3% of average display rate) and large  $K$ , which reduces the impact of wasted bandwidth in FSEB. If we used relatively large  $c$  and small  $K$ , which may make the wasted bandwidth significant, the server bandwidth would not decrease monotonically. We give an example in Table III where we change  $c$  from 60Kbps to 70Kbps with  $K = 25$  (keeping  $w$  the same).

TABLE III. SMALL  $K$  AND LARGE  $c$  FOR CHANGING CHANNEL BANDWIDTH

$c$ (Kbps)	Normalized Channel Bandwidth	Normalized Server Bandwidth
60	9.87%	3.851
61	10.04%	3.814
62	10.20%	3.775
63	10.37%	3.836
64	10.53%	3.791
65	10.70%	3.851
66	10.86%	3.801
67	11.02%	3.748
68	11.19%	3.804
69	11.35%	3.747
70	11.52%	3.801

#### F. Lower Bound on Server Bandwidth

As we observed above, for a given  $B_u$ , we can choose different  $c$  or  $K$  to get different server bandwidth for a VBR video with a fixed wait time. The smaller  $c$  we choose, the smaller the server bandwidth gets.

Figure 7 shows the trend. In Figure 7, we keep  $B_u$  the same (1.896 times the average display rate  $b$ ) for video BOND, while changing  $c$  from 8Kbps to 128Kbps with  $w = 16$  (sec) (thus, the normalized wait time is 0.01).

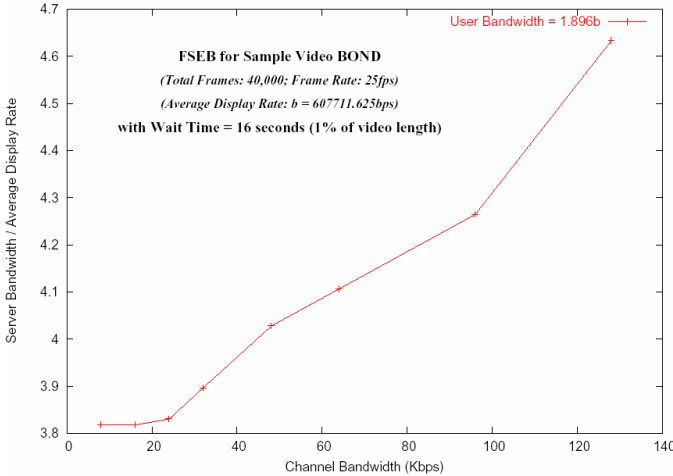


Figure 7. Normalized server bandwidth vs. channel bandwidth for FSEB with the same user bandwidth

Since the bandwidth usage of the last channel is different for different  $c$  with a fixed wait time, we cannot draw the conclusion that choosing minimum  $c$  always leads to the minimum server bandwidth. Table IV shows the bandwidth usage of the last channels for some small channel bandwidths with the same user bandwidth limit and wait time as those in Figure 7 for video BOND. From the values in Table IV, we can see that those normalized server bandwidth values differ from each other by no more than 0.8%.

TABLE IV. BANDWIDTH USAGE OF LAST CHANNEL

$c$ (bps)	$K$	Bandwidth Usage of Last Channel	Normalized Server Bandwidth
16000	72	61.984%	3.818
8000	144	42.693%	3.818
4687	245	13.999%	3.849
4687	246	8.410%	3.841

Although choosing the minimum  $c$  cannot ensure the minimum server bandwidth, it is still safe to say that choosing smaller  $c$  would make the server bandwidth closer to the minimum value.

#### IV. COMPARISONS

We compare the performance of FSEB and SCB [10]. In our implementation of SCB, we choose the adjacent segments to merge if merging them results in the minimum difference between the maximum and minimum channel bandwidths among all adjacent pairs.

##### A. Input parameters

FSEB and SCB have different sets of input parameters.

SCB has three design parameters:

- $n$  — Number of segments (=number of channels);
- $K$  — Max number of user channels ( $K \leq n$ );
- $w$  — User wait time;

FSEB also has three design parameters:

- $w$  — User wait time;
- $c$  — Channel bandwidth;
- $K$  — Maximum number of user channels.

##### B. Outputs

Given a set of chosen parameters, these two schemes generate a set of results.

Outputs of SCB:

- $b_i$  — Channel bandwidth in bps of each segment given by (1) ( $i = 1, \dots, n$ );
- $B_s$  — Server bandwidth in bps given by (2);
- $B_u$  — User bandwidth in bps given by (3).

Outputs of FSEB:

- $n$  — Number of segments;
- $B_s$  — Server bandwidth in bps, which is  $n \times c$ ;
- $B_u$  — User bandwidth in bps, which is  $K \times c$ .

##### C. Common features

These two schemes have the following common features:

- Both ensure that each segment has a whole number of frames;
- Wait time  $w$  can be directly controlled;
- Number of user channels  $K$  can be directly controlled.

##### D. Advantages of FSEB

- Channels have equal bandwidth;
- Channel bandwidth  $c$  can be directly controlled;
- FSEB considers user bandwidth limit ( $K \times c$ ), while SCB only considers the limit  $K$  on the number of user channels.

##### E. Disadvantages of FSEB

- There is the minimum value for  $K$  to ensure no segmentation failure due to (5), while SCB has no limitation for  $K$ ;
- $n$  cannot be directly controlled, while in SCB it is an input parameter.

To compare FSEB and SCB, we use the same set of values for the two common input parameters,  $w$  and  $K$ , by setting  $w = 2$  (corresponding to the normalized wait time = 1%) and  $K = 3$  as in Figure 2 of [10] (which discusses SCB). We first ran SCB for  $n$  from 3 to 7. We then ran FSEB with different channel bandwidths to produce  $n$  from 3 to 7. Figure 8 shows the results for video FUSS from [20]. For FSEB, the bandwidth values plotted are  $n \times c$  and  $K \times c$ , which include the unused portions. If only the actually used amounts were plotted, the curves for FSEB would be lower than those shown in the figure. In spite of the unfairness, it can be observed that FSEB outperforms SCB, in some cases rather significantly.

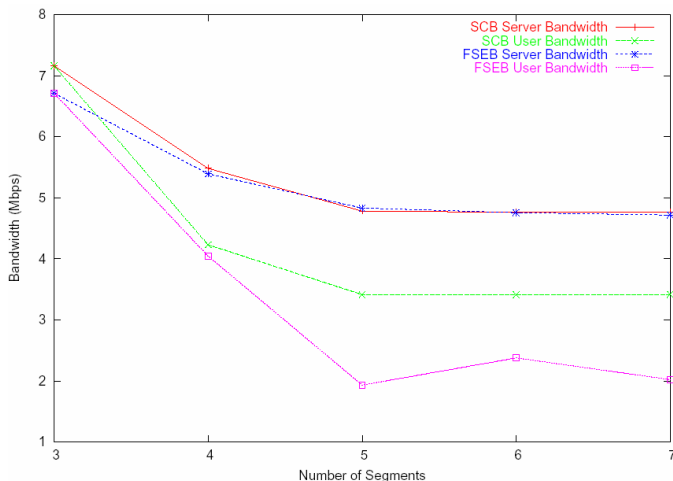


Figure 8. Server and user bandwidth vs. number of segments of FUSS for FSEB and SCB with the same wait time and user channels

Here we only give one example although all other VBR-encoded videos we have tested yielded similar results.

Table V lists the properties of the sample video FUSS used above.

TABLE V. PROPERTIES OF SAMPLE VIDEOS FUSS

Video	Size(MB)	Average Display Rate(bps)	$N$	$F$ (fps)
FUSS	16.71	700,889.080	5,000	25

## V. CONCLUSION AND FUTURE WORK

We have presented a new lossless scheme to consider client bandwidth constraint. Using equal bandwidth for each channel, our FSEB scheme segments a given VBR-encoded video forward from the first frame by linear time. Each segment is periodically broadcast in a different channel. With a given user bandwidth limit, the two input parameters of user channels  $K$  and channel bandwidth  $c$  can be adjusted to achieve required server bandwidth. If the channel bandwidth is small enough, our scheme would be close to the minimum server bandwidth for any given client bandwidth and wait time.

Besides adjustable parameters, standard equal channels also make it very easy to be implemented.

A modification that is necessary when deploying FSEB in practice is to segment videos taking the frame types into consideration. An approach similar to what we use in [26] can be used for this purpose.

The two optimization problems (A) and (B) mentioned in Section II haven't been solved. We leave them as future work.

## REFERENCES

- [1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, A permutation-based pyramid broadcasting scheme for video-on-demand systems, *Proc. IEEE Int'l conf. on Multimedia Systems '96*, pp. 118-126, Hiroshima, Japan, June 1996.
- [2] A. Dan, D. Sitaram and P. Shahabuddin, Scheduling policy for an on-demand video server with batching. *Proc. ACM Multimedia '94*, San Francisco, CA, Oct. 1994, pp.15-23.
- [3] K. A. Hua and S. Sheu, Skyscraper Broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems," *Proc. SIGCOMM 97*, pp. 89-100, Cannes, France, September 1997.
- [4] K. A. Hua, Y. Cai and S. Sheu. Exploiting client bandwidth for more efficient video broadcast. *Proc. IEEE ICCN '98*, pp. 848-856, October 1998.
- [5] A. Hu, Video-on-demand broadcasting protocols: A comprehensive study, *Proc. INFOCOM '01*, pp. 508-517, Anchorage, AK, April 2001.
- [6] A. Hu, I. Nikolaidis and P. van Beek, On the design of efficient video-on-demand broadcast schedules, *Proc. 7th Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '99)*, pp. 262-269, 1999.
- [7] L. Juhn and L. Tseng, Harmonic broadcasting for video-on-demand service, *IEEE Trans. on Broadcasting 43* (3), pp. 268-271, September 1997.
- [8] L. Juhn and L. Tseng, Fast data broadcasting and receiving scheme for popular video service, *IEEE Trans. on Broadcasting 44* (1), pp.100-105, March 1998.
- [9] F. Li and I. Nikolaidis, Trace-adaptive fragmentation for periodic broadcast of vbr video. *Proc. 9th Int'l Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '99)* June 1999.
- [10] F. Li and I. Nikolaidis, SCB: Staircase broadcast for media-on-demand systems. *Proc. IEEE MoMuC'2000*, 3B-1-1-3B-1-5.
- [11] I. Nikolaidis, F. Li and A. Hu, An inherently lossless and bandwidth efficient scheme for periodic broadcast of vbr video. *Proc. ACM SIGMETRICS '00*, pp. 116-117, June 2000.
- [12] J.-F. Pâris, A broadcasting protocol for compressed video, *Proc. EUROMEDIA '99, Conf.*, pp. 78-84, Munich, April 1999.
- [13] J.-F. Pâris, A simple low-bandwidth broadcasting protocol for video-on-demand, *Proc. 8th Int'l Conf. on Computer Communications and Networks (IC3N '99)*, pp. 118-123, Boston-Natick, MA, October 1999.
- [14] J.-F. Pâris, A fixed-delay broadcasting protocol for video-on-demand, *Proc. 10th Int'l Conf. on Computer Communications and Networks 2001*, pp. 418-423, October 2001.
- [15] J.-F. Pâris, S.W. Carter and D.D.E. Long, Efficient broadcasting protocols for video on demand, *Proc. 6th Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98)*, pp. 127-132, July 1998.
- [16] J.-F. Pâris, S.W. Carter and D.D.E. Long, A low bandwidth broadcasting protocol for video on demand, *Proc. 7th Int'l Conf. on Conference on Computer Communications and Networks (IC3N'98)*, pp. 690-697, October 1998.
- [17] J.-F. Pâris, S.W. Carter and D.D.E. Long, A hybrid broadcasting protocol for video on demand, *Proc. 1999 Multimedia Computing and Networking Conf. (MMCN'99)*, pp. 317-326, San Jose, CA, January 1999.

- [18] J.-F. Pâris, S.W. Carter and P.E. Mantey, Zero-delay broadcasting protocols for video-on-demand, *Proc. 1999 ACM Multimedia Conf., Orlando, FL*, pp. 189-197, November 1999.
- [19] J.-F. Pâris and D.D.E. Long, The case for aggressive partial preloading in broadcasting protocols for video-on-demand, *Proc. 2001 IEEE Int'l Conf. on Multimedia and Expo*, pp.113-116, Tokyo, August 2001.
- [20] O. Rose, Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems, *Tech. Rept. 101*, Universitaet Wuerzburg, February 1995.
- [21] D. Saporilla, K. Ross and M. Reisslein, Periodic broadcasting with vbr-encoded video, *Proc. IEEE INFOCOM '99*, pp. 464-471, March 1999.
- [22] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*, Prentice Hall, Inc., 1995.
- [23] S. Viswanathan and T. Imielinski, Pyramid Broadcasting for video on demand service, *Proc. IEEE Conf. on Multimedia Computing and Networking*, Vol. 2417, pp. 66-77, San Jose, CA, 1995.
- [24] S. Viswanathan and T. Imielinski, Metropolitan area video-on-demand service using pyramid broadcasting, *Multimedia Systems*, 4(4): 197-208, August 1996.
- [25] J.W. Wong, Broadcast delivery, *Proc. IEEE*, 76 (12): 1566-1577, December 1988.
- [26] S. Wu, General frame level segmentation for periodic broadcast of vbr videos, *Proc. 12th Int'l Conf. on Conference on Computer Communications and Networks (IC3N'03)*, October 2003, To appear.
- [27] E.M. Yan and T. Kameda, An efficient VOD broadcasting scheme with user bandwidth limit, *Proc. SPIE/ACM Conf. on Multimedia Computing and Networking*, Vol. 5019, pp. 200-208, Santa Clara, CA, January 2003.