Lossless VBR Video Broadcasting Considering User Bandwidth Limit

Shufang Wu and Tiko Kameda
School of Computing Science, Simon Fraser University
{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 using 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. FSEB takes a video as a frame sequence, divides the video into segments, and broadcasts each segment periodically on a separate logical channel. Experimental results with real videos reveal that FSEB requires less server bandwidth and user bandwidth than the only other lossless scheme for VBR videos currently known, which also can limit the number of channels that a user can access simultaneously.

1. Introduction

Broadcasting [11] has been shown to be a bandwidth-efficient way to transport frequently requested videos in Video-On-Demand (VOD) systems. The idea of using segmentation together with broadcasting was first conceived in the little-known work in 1991 by Hollmann and Holzscherer [1]. Since then, many segmentation-based schemes for constant bit rate (CBR) encoded videos with varying performance have been reported. For a brief review on these schemes, the reader is referred to an insightful survey [3]. Several other schemes, such as Client-Centric Approach (CCA) [2] and Generalized Fibonacci Broadcasting (GFB) [13], take the bandwidth capability of the user into consideration.

To consider the variable bit rate (VBR) encoded videos in practice, some broadcast schemes have been proposed [5,8,10]. Further, schemes that respect video frame boundaries have also been proposed.

Given the wait time and number of segments, Loss-Less and Bandwidth-Efficient scheme (LLBE) [7] addresses the issue of minimizing the server bandwidth for real VBR videos. StairCase Broadcast (SCB) [6] addresses the same issue with the additional constraint that the number of user channels is also pre-specified. 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 is not controllable.

To be able to control the channel bandwidth, we propose a new broadcast scheme, named Forward Segmentation using Equal Bandwidth (FSEB). To approach the problem from a different angle, FSEB uses uniform channels off the butt by giving the upper bound on the bandwidth \( c \) (bits/sec) of each channel as an input. The other inputs are the wait time \( \ell \) (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 \( \ell \) or \( c \) can be decreased slightly to make full use of the last channel. FSEB gives implementers fine-grained control in terms of the input parameters \( \ell \), \( c \) and \( K \).

The remainder of the paper is organized as follows. In Section 2, we introduce FSEB. Section 3 illustrates through experimental results the properties of FSEB. In Section 4, we evaluate the performance of FSEB. Finally, Section 5 concludes the paper.

2. Forward Segmentation using Equal Bandwidth (FSEB)

It is shown in [4] that for CBR videos the total bandwidth is minimized when all channels have equal bandwidth. Even for VBR videos, it is observed in [6] that when the total bandwidth is minimized, the channel bandwidths are roughly equal. FSEB thus uses channels of equal bandwidth, which is an obvious advantage in implementation. A round of FSEB algorithm computes in linear time the sizes of 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 then the second channel, and so on, down to the last channel. Therefore, it is likely that the last or the nth channel will be only partially filled. If that is the case, we decrease w or c in small increments, until there is little 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.

Given \( w, c \) and \( K \) as inputs, \( n \) and \( X(j) \) \( (j = 1, ..., n) \) need to be computed, where \( X(j) \) \( (j = 1, ..., n) \) stands for the index of the last frame in the \( j \)-th segment \( S_j \) \( (j = 1, ..., n) \). The detailed procedure is given below.

First, to ensure an uninterrupted display of the video once it starts, the download time of the \( j \)-th segment must not be larger than \( w \) plus the time to display the previous \( j-1 \) segments. Since the user can download from up to \( K \) channels simultaneously, we can compute \( X(j) \) for \( 1 \leq j \leq K \) from

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

where \( A(i) \) is the size in bits of the first \( i \) frames, \( F \) is the display rate of the video in frames per second.

As for \( S_j \) when \( K < j \leq n \), once the user finishes downloading \( S_{j-1} \), 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 \( S_{j-1} \) completes. So, the download time for \( S_j \) is the display time of segments \( S_{j-2}, ..., S_{j-1} \), which is the number of frames in those \( K \) previous consecutive segments divided by the display rate. We thus have

\[
\frac{A(X(j)) - A(X(j-1))}{X(j-1) - X(j-K-1)} \leq c, \quad K < j \leq n.
\]

When we construct \( S_j \) using (1) or (2), it could happen that even the first frame after \( X(j-1) \) may not satisfy the relevant inequality. That is, \( X(j)=X(j-1)+1 \) fails to satisfy it. Segmentation fails in this situation. What we can do is to increase \( c \) or \( K \) and try again. Increasing \( c \) or \( K \) implies increasing the user bandwidth \( B_c \). This means that there must be a minimum user bandwidth to avoid unsuccessful segmentation of a video for a given wait time.

The pseudocode for FSEB is shown below:

```plaintext
// Compute A(j):
// f(j) is the size in bits of the i-th frame (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 (1) is true) {
            i = i + 1;
        }
        X(j) = i - 1;
    } else {
        while (inequality (2) is true) {
            i = i + 1;
        }
        X(j) = i - 1;
    }
    if (X(j) == X(j-1)) {
        j = j - 1;
    }
    break; // Segmentation fails
}
// 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 \( O(N) \), while SCB has time complexity \( O(N^2) \).

### 3. Experimental Results

We applied FSEB to some real VBR video traces from [9]. Since the performance results are similar regardless of the VBR video content, we only use one representative video here.

**Figure 2. Server bandwidth vs. wait time for FSEB**

Figure 2 plots the server bandwidth \( B_c \) against the wait time \( w \) when \( K \) and \( c \) are fixed at 19 and 64Kbps,
respectively, for a VBR video trace (BOND) from [9]. The figure shows that the server bandwidth in general decreases when we increase the wait time with the 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 $w$.

FSEB gives implementers fine-grained control via the input parameters $w$, $c$, and $K$. Given the maximum user bandwidth $B_u$ and a fixed $w$, we can adjust $K$ and $c$ within the constraint of $K \times c \leq B_u$. Decreasing (increasing) $c$ would increase (decrease) both $K$ and $n$. But decreasing $c$ would delay the downloading of later frames, thereby decreasing the total server bandwidth.

Figure 3 shows the effect of varying $c$ and $K$ within the constraint of $K \times c = 2.001b$ for video BOND, where $b$ (bits/sec) is its average display rate. It can be seen that for a fixed wait time and user bandwidth, a smaller $c$ leads to smaller server bandwidth.

Figure 4 shows the trend more clearly. In Figure 4, 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 $w = 16$ (thus, the normalized wait time is 0.01).

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 the server bandwidth so that we can trade the user side bandwidth with the server bandwidth to a certain extent (similarly to the idea in CCA [2]). Figure 5 shows the server bandwidth as a function of $K$ for video BOND when $w = 16$ and $c = 16000$. It demonstrates that the server bandwidth can be saved by increasing the user bandwidth.

Figure 5. Server and user bandwidth vs. number of user channels

Figure 6. Server and user bandwidth vs. channel bandwidth

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

4. Performance Evaluation

We evaluate the performance of FSEB by comparing it with SCB. Here we only give one example, since all other VBR-encoded videos we have tested yielded similar results.

In our implementation of SCB, we chose to merge two adjacent segments if merging them resulted in the minimum difference between the maximum and minimum channel bandwidths among all adjacent pairs.

For comparison purposes, we used the same set of values for the two common input parameters, \( w \) and \( K \), by setting \( w = 2 \) (corresponding to 1% of the video length) and \( K = 3 \) as in Figure 2 of [6] (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 7 shows the results for VBR video FUSS from [9]. For FSEB, the bandwidth values plotted are \( n \times c \) and \( K \times c \), which include the small amounts of unused bandwidth. If only the actually used amounts were plotted, the curves for FSEB would be lower than those shown in the figure. In spite of this unfairness, it can be observed that FSEB outperforms SCB, in some cases rather significantly.

5. Conclusion

We have presented a new lossless broadcasting scheme for VBR videos to take user bandwidth constraint into consideration. Using equal bandwidth for each channel, our FSEB scheme segments a given VBR-encoded video forward from the first frame in linear time. Each segment is periodically broadcast in a different channel. With a given user bandwidth limit, the two input parameters, i.e., the number of user channels \( K \) and the bandwidth \( c \) of each channel, can be adjusted to achieve the required server bandwidth. If the channel bandwidth is small enough, our scheme would achieve near minimum server bandwidth for any given user bandwidth and wait time. Besides adjustable parameters, being able to use channels of “standard” bandwidths would facilitate implementation. Experimental results show that FSEB in general outperforms SCB, which is the only other similar scheme.

A modification that is necessary when deploying FSEB in practice is to segment videos taking the frame types (I, P and B) into consideration. An approach similar to that in [12] can be used for this purpose.

6. References