*Proc. 7th ACM International Multimedia Conference*, Orlando, FL, Oct.-Nov. 1999, *to appear*.

# Zero-Delay Broadcasting Protocols for Video-on-Demand

Jehan-François Pâris
Department of Computer Science
University of Houston
Houston, TX 77204-3475
paris@cs.uh.edu

Darrell D. E. Long  Patrick E. Mantey
Jack Baskin School of Engineering
University of California
Santa Cruz, CA 95064
{darrell, mantey}@cse.ucsc.edu

## ABSTRACT

Broadcasting protocols for video-on-demand continuously retransmit videos that are watched simultaneously by many viewers. Nearly all broadcasting protocols assume that the client set-top box has enough storage to store between 48 and 60 minutes of video. We propose to use this storage to anticipate the customer requests and to preload, say, the first 3 minutes of the top 16 to 20 videos. This would provide instantaneous access to these videos and also eliminate the extra bandwidth required to handle compressed video signal.

We present two broadcasting protocols using partial preloading to eliminate this extra bandwidth. The first of these protocols, Polyharmonic Broadcasting with Partial Preloading (PHB-PP), partitions each video into between 20 and 160 segments of equal duration and allocates a separate data stream to each individual segment. Our second protocol, the *Mayan Temple Broadcasting* protocol, uses fewer data streams but requires more overall bandwidth.

**Keywords:** video-on-demand, broadcasting protocols, pyramid broadcasting, compressed video.

## 1. INTRODUCTION

Despite all its attractiveness, video-on-demand (VOD) [14] has yet to succeed in the marketplace. None of the companies that invested in VOD have been able to deploy a single successful commercial system. One of the reasons for this lack of success is that it must compete with cheaper, well established rivals such as video rentals and pay-per-view.



Most efforts aimed at reducing the cost of VOD services have focused on distributing the top ten or twenty most popular videos more efficiently since these videos are likely to be responsible for over forty percent of the total demand [4, 5]. One of the most promising approaches is to schedule repeated broadcasts of these "hot" videos rather than waiting for individual requests. This technique is known as *broadcasting*. The approach has been characterized as being *proactive* as it attempts to predict customer demand rather than responding to it in a *reactive* fashion. One limitation of broadcasting is that customers who want to watch a video now must wait, say, between five and fifteen minutes for the next scheduled broadcast of the video.

The past two years have seen the development of many efficient broadcasting protocols. All these protocols divide each video into *segments* that are simultaneously broadcast on different data streams. One of these streams transmits nothing but the first segment of the video. The other streams transmit the remaining segments at lower bandwidths. When customers want to watch a video, they wait first for the beginning of the first segment on the first stream. While they start watching that segment, their set-top box (STB) also starts downloading data from the other streams. The only drawback of the approach is that the STB must have enough local storage to store up to 40 or 50 percent of each video being viewed. In the current state of the storage technology, this implies that the STB must have a local disk.

We propose to use this local storage to anticipate the customer demand and to store the first few minutes of the top ten to twenty most popular videos. This would allow us to provide instantaneous access to these videos and would also reduce the bandwidth required to broadcast them. In addition, this would provide us with enough buffering to eliminate the extra bandwidth required to guarantee jitter-free delivery of the compressed video signal.

All the broadcasting protocols that have been developed so far assume that videos will have a fixed bandwidth corresponding to

a fixed video consumption rate. This assumption is incorrect because the server will broadcast compressed videos whose bandwidth requirements will depend on the rate at which the images being displayed change [2, 7]. To ensure jitter-free delivery of video in a system allocating a fixed bandwidth to each video, the VOD server will have to set the broadcasting bandwidth to the maximum bit rate required by the most rapidly changing moments of the fastest paced scenes of the video.

The two broadcasting protocols we are presenting avoid this drawback by guaranteeing that each segment of a video will always be completely received by the client set top box by the time the customer has finished viewing the previous segment of the video. Hence the individual arrival times of the frames at the client STB become irrelevant as long as all the frames arrive within the required time interval. The VOD server can thus transmit the segment data at the maximum bandwidth *b* allowed by the channel it is using to broadcast the segment.

The remainder of the paper is organized as follows. Section 2 presents the relevant broadcasting protocols for video-on-demand. Section 3 introduces our approach and presents our two new protocols and Section 4 discusses their bandwidth requirements. Finally, Section 5 contains our conclusions.

## a) BROADCASTING PROTOCOLS FOR VIDEO-ON-DEMAND

The simplest broadcasting protocol is *staggered broadcasting* [3]. A video broadcast under that protocol is continuously retransmitted over *k* distinct video channels at equal time intervals. The approach does not necessitate any significant modification to the set-top box but requires a fairly large number of channels per video to achieve a reasonable waiting time. All other broadcasting protocols can be subdivided into two groups. Protocols in the first group are all based on Viswanathan and Imielinski's *Pyramid Broadcasting* protocol [13]. These include Aggarwal, Wolf and Yu's *Permutation-Based Pyramid Broadcasting* protocol [1] and Hua and Sheu's *Skyscraper Broadcasting* protocol [8].

These three protocols subdivide each video into *k* segments of increasing sizes and transmit them over *k* data streams of equal bandwidth. When customers request a video, they wait for the start of an instance of the first segment of the video. Their STB then starts downloading data from other segments of the video. These protocols require much less bandwidth than staggered broadcasting to achieve the same maximum waiting time because they use much less bandwidth to transmit the later portions of the videos they broadcast. Nevertheless, they cannot match the performance of the so-called *harmonic protocols*, which we will discuss in more detail.

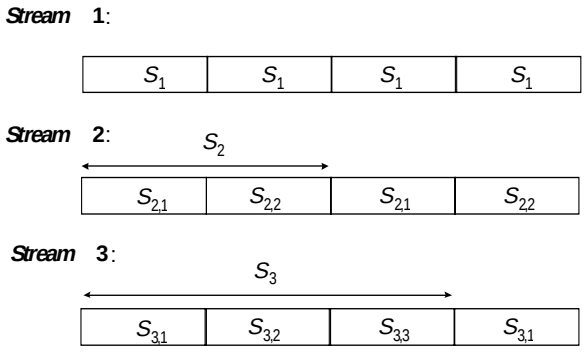


Figure 1: An illustration of the first three segments of a video under harmonic broadcasting

*Harmonic Broadcasting* (HB) [9] divides each video into *n* segments of equal duration $d = D/n$ where *D* is the total duration of the video. It repeatedly broadcasts each segment $S_i$, for $1 \le i \le n$, on a separate data stream with a bandwidth $b/i$, where *b* is the consumption rate of the video (see Figure 1).

The total bandwidth required by the HB protocol to broadcast a video is thus given by

$$B_{HB}(n) = \sum_{i=1}^{n} \frac{b}{i} = bH(n)$$

where $H(n)$ is the $n^{\text{th}}$ harmonic number.

Since the first segment is broadcast at a bandwidth equal to the video consumption rate *b*, the maximum amount of time customers will have to wait before viewing a video is given by the duration *d* of that first segment.

Unfortunately, HB cannot always deliver all data on time. To see this, let us define a *subsegment* as the fraction of a segment the client receives during *d* time units. The first segment only has one subsegment, the segment itself and every other segment $S_i$ has *i* equal subsegments, $S_{i,1}, \ldots, S_{i,i}$. Consider then the first two streams in Figure 1. If the client makes its request in time to receive the second instance of $S_1$ and starts receiving data at time $t_0$, it will need *all of the data* for $S_{2,1}$ by time $t_0 + 3/2d$. However, it will not receive all of that data until time $t_0 + 2d$. As it turns out, HB will not work unless the client always waits an extra *d* time units before consuming the data.

*Cautious Harmonic Broadcasting* (CHB) [10] does not impose this extra waiting time. It uses $n-1$ streams and broadcasts the first segment of the video on its first stream in a similar fashion as HB. Its second stream alternates between broadcasting $S_2$ and $S_3$ at bandwidth *b*. The remaining $n-3$ streams then broadcast segments $S_4$ to $S_n$ in a manner such that the $i^{\text{th}}$ stream transmits segment $S_{i+1}$ at a bandwidth $b/i$. As before, the client will start downloading data from all streams when it starts segment $S_1$. Hence the total bandwidth required by the CHB protocol will be given by

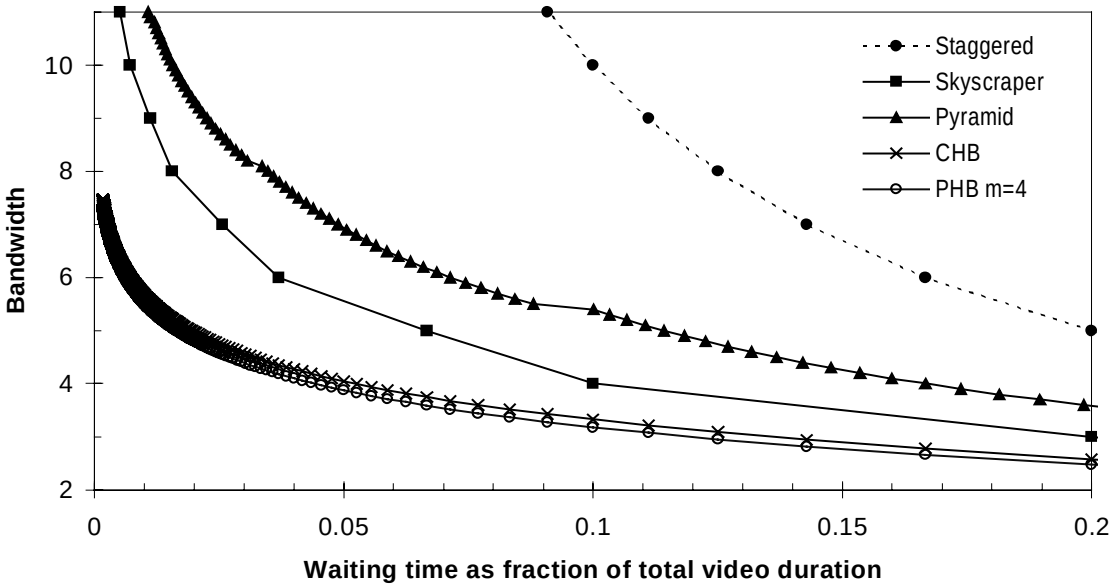


Figure 2: Bandwidth requirements of typical broadcasting protocols

$$B_{CHB}(n) = 2b + \sum_{i=4}^{n-1} \frac{b}{i-1} = \frac{b}{2} + bH(n-1)$$

that is, roughly half a standard video channel more than the original HB protocol.

Another harmonic protocol, *Polyharmonic Broadcasting* (PHB) [11], implements a deterministic wait policy: no client can start consuming the first segment of the video before having downloaded data from all *n* streams during a time interval of duration $w = md$ where *m* is some positive integer $m \geq 1$. As a result, PHB requires many more data streams but less total bandwidth than CHB to achieve the same maximum waiting time.

Figure 2 shows the bandwidth versus client waiting times for staggered broadcasting, Skyscraper Broadcasting with a maximum width of 52, Pyramid Broadcasting with $\alpha = 2$, Cautious Harmonic Broadcasting and Polyharmonic Broadcasting with *m* = 4. To eliminate the factor *D* representing the duration of the video, the maximum waiting times on the *x*-axis are expressed as percentages of the video lengths. All quantities on the *y*-axis are expressed in standard video channels. As one can see, staggered broadcasting requires much more bandwidth than the four other protocols to guarantee the same maximum waiting time. Moreover, the two harmonic protocols have the lowest bandwidth requirements of all five protocols.

## 3. OUR APPROACH

Skyscraper Broadcasting, Pyramid Broadcasting, Cautious Harmonic Broadcasting and Polyharmonic Broadcasting outperform staggered broadcasting because they assume that the user STB has enough free disk space to store between 40 and 50 percent of each video while it is being played. As a result, the later portions of each video can be broadcast less frequently and will require less bandwidth.

Let us now turn our attention to finding another use for this disk space. Even though it will be empty most of the time, it cannot be used for storing any permanent data since those would be erased any time the customer orders a video. A better solution would be to use this disk space to preload the first few minutes of the top ten to twenty most popular videos, that is, the videos that are likely to be responsible for over forty percent of all customer requests. Assuming that we have enough space to store 60 minutes of video, we could preload, say, the first 6 minutes of the top 10 videos or the first 3 minutes of the top 20 videos. We will call this technique *partial preloading*. It differs from other recent proposals in that the preloaded portions of each video will reside in the client STB rather than at a proxy server [6, 12].

Our proposal would offer three major advantages. First it will provide instant access to these videos. Second, it will reduce the bandwidth required to broadcast them as the first minutes of each video could be broadcast much less frequently. Finally, it could reduce the amount of extra bandwidth required to guarantee jitter-free delivery of the video signal.

As we mentioned earlier, the video signal received by the STB is very likely to be a *compressed video signal* whose bandwidth requirements will depend on the rate at which the images being displayed change [2, 7]. For instance, daytime action scenes and cartoons will require more bandwidth than slower moving scenes

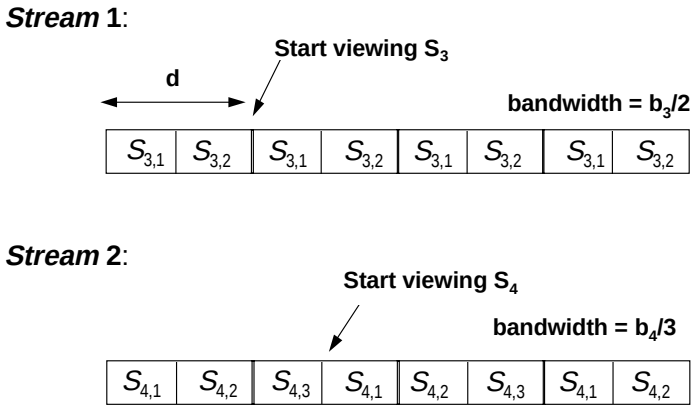


Figure 3: An illustration of the first two streams for Polyharmonic Broadcasting with Partial Preloading (PHB-PP) and m=2.

and night scenes. In the absence of any buffering, the VOD maximum bit rate required by the fastest moments of the fastest paced scenes of the video. As a result, a significant fraction of the bandwidth will remain unused most of the time.

Consider now a broadcasting protocol where each segment is always completely downloaded by the STB *before* the customer finishes watching the previous segment of the video. Under this hypothesis, the actual frame arrival times at the STB become totally irrelevant as long as all the frames arrive within the required time interval. The VOD server can thus transmit the segment data at the maximum bandwidth *b* allowed by the channel it is using to broadcast the segment.

We present two new broadcasting protocols with partial preloading that satisfy this requirement and guarantee that each segment of a video will always be completely received by the client STB by the time the customer is done with the previous segment of the video. The first of these protocols is a variant of the Polyharmonic Broadcasting protocol, which we will call Polyharmonic Broadcasting with Partial Preloading (PHB-PP). The PHB-PP protocol will partition each video to be broadcast into *n* segments of equal duration and preload *m* of these segments. It will then dedicate a separate data stream to each the remaining $n-m$ segments. This approach results in a very low aggregate bandwidth since each segment can be broadcast at the exact bandwidth it requires. Unfortunately, it complicates significantly the task of the VOD server, which could have to broadcast a fairly large number of streams per video. Our second protocol avoids this problem by using segments of increasing size and allocating one standard video channel to each of these segments. We will call this protocol the *Mayan Temple Broadcasting* protocol after the shape of its stacked segments to emphasize that it is a pyramid-based protocol.

## 3.1. Notations

Let *V* be a compressed video of duration *D*. Its bandwidth requirements will vary over time and can be expressed by a function of time $b(t)$, which will be defined over the time interval $[0, t]$. For our convenience, we will also define a *cumulative bandwidth function* $F(t)$ that will be defined as

$$F(t) = \int_0^t b(u)du$$

Consider now a video segment *S* of duration $t_2 - t_1$ starting at time $t_1$ and ending at time $t_2$. The *average bandwidth* $b_S$ of that segment will be given by

$$b_S = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2} b(u)du = \frac{1}{t_2 - t_1}[F(t_2) - F(t_1)]$$

Like all efficient broadcasting protocols, our two protocols assume that the client STB has enough free space to store at least fifty to sixty percent of each video being broadcast. This should cease soon to be a problem as disk capacities have been doubling every year over the last three years and this trend is expected to continue within the foreseeable future. In addition, our protocols assume that this free space can be used to preload the first *d* minutes of the *N* most popular videos. Given the varying bandwidth requirements of each video, this means that the sizes of the preloaded data will vary.

## 3.2. The Polyharmonic Broadcasting Protocol with Partial Preloading

The Polyharmonic Broadcasting protocol with Partial Preloading (PHB-PP) partitions each video into $n = mD/d$ equal segments $S_1, S_2, ..., S_n$, where *D* is the duration of the video, *d* the duration of its preloaded portion and *m* some positive integer ≥ 1. The duration $d_S$ of an individual segment is given by

$$d_S = \frac{D}{n} = \frac{d}{m},$$

The first $m$ of these segments will be preloaded in the customer STB. Each of the remaining $n-m$ segments, that is segments $S_{m+1}$ to $S_n$, will have its own dedicated data stream that will continuously rebroadcast the segment. Hence stream $i$ will continuously rebroadcast segment $S_{m+i}$.

Let us now compute the individual bandwidths of these $n-m$ streams. The bandwidth $b_{m+i}$ at which segment $S_{m+i}$ will be transmitted must always be sufficient to guarantee that $S_{m+i}$ will always be completely downloaded by the client STB by the time that the customer has finished watching the previous segment.

Consider, for instance, the example of Figure 3 where $m$ = 2. Since the preloaded part of the video contains the first two segments, the first stream will repeatedly broadcast segment $S_3$ every $2d_S$ time units and the second stream will repeatedly broadcast segment $S_4$ every $3d_S$ time units.

More generally, segment $S_{m+1}$ will be the first segment not to be preloaded and its average bandwidth will be given by

$$b_{m+1} = \frac{1}{d_s}[F((m+1)d_S) - F(md_S)].$$

To guarantee that the segment will be completely downloaded by the time the customer has finished viewing the preloaded part of the video, its entire contents need to be repeated every $d$ time units. Hence the bandwidth of the first stream will be equal to

$$\frac{1}{md_s}[F((m+1)d_S) - F(md_S)].$$

Similarly, the average bandwidth of segment $S_{m+i}$ will be given by

$$b_{m+i} = \frac{1}{d_s}[F((m+i)d_S) - F((m+i-1)d_S)].$$

Since the segment will have to be repeated every $(m+i-1)d_S$ time units, the bandwidth of stream $i$ will be given by

$$\frac{1}{(m+i-1)d_S}[F((m+i)d_S) - F((m+i-1)d_S)].$$

The total bandwidth required to broadcast the $n-m$ segments that were not preloaded in the client STB is then given by

$$B_{PHB-PP}(n,m) = \sum_{i=1}^{n-m} \frac{1}{(m+i-1)d_S}[F((m+i)d_S) - F((m+i-1)d_S)].$$

If all segments would have the same average bandwidth $b_S$, this expression would simplify to

$$\begin{aligned} B_{PHB-PP}(n,m) &= \sum_{i=1}^{n-m} \frac{1}{(m+i-1)} b_S \\ &= b_s[H(n-1) - H(m-1)]. \end{aligned}$$

where $H(n)$ is the $n^{\text{th}}$ harmonic number. To see how the parameters $n$ and $m$ affect the total bandwidth, let us define $k = n/m$ and rewrite the above expression as

$$\begin{aligned} B_{PHB-PP}(km,m) &= \sum_{i=1}^{km-m} \frac{1}{(m+i-1)} b_S \\ &= b_s[H(km) - H(m-1)]. \end{aligned}$$

We can then show that

$$B_{PHB-PP}(k,1) = \sum_{i=1}^{k-1} \frac{1}{i} b_S = b_s H(k-1) = B_{HB}(k-1)$$

and

$$\begin{aligned} \lim_{m\to\infty} B_{PHB-PP}(km,m) &= \lim_{m\to\infty} \sum_{i=1}^{km-m} \frac{d/m}{d + \frac{i-1}{m} d} b_S \\ &= \int_d^D \frac{b_s}{u}\, du = b_s \log\frac{D}{d} \\ &= b_s \log k. \end{aligned}$$

These two expressions respectively provide upper and lower bounds for the bandwidth required to broadcast a video of duration $D$ minutes when its first $D/k$ minutes are already loaded in the client STB buffer. The upper bound corresponds to the case when $m = 1$ and the protocol uses exactly $k-1$ data streams. The bandwidth requirements of the protocol are then equal to those of a Harmonic Broadcasting protocol with $k-1$ segments. The lower bound corresponds to the limit case when $m$ goes to infinity. This lower bound is purely theoretical as it would require an infinite number of very low-bandwidth data streams. It can however be approached very closely with values of $m$ as small as 4. Consider for instance the case of a two-hour video whose first three minutes are preloaded. With $m = 1$, we would use 39 data streams to achieve a total bandwidth of 4.75 standard video channels. Selecting $m = 4$ would require 156 data streams but their total bandwidth would only be 3.75 video channels, that is, only 0.06 video channels more than the theoretical minimum.

## 3.3. The Mayan Temple Broadcasting Protocol

The only limitation of Polyharmonic Broadcasting is the rather large number of independent data streams it requires. The *Mayan Temple Broadcasting* protocol avoids this problem by using segments of increasing size and allocating a standard video channel to each of these segments. We will assume these video channels to be identical and have the same bandwidth $b$.

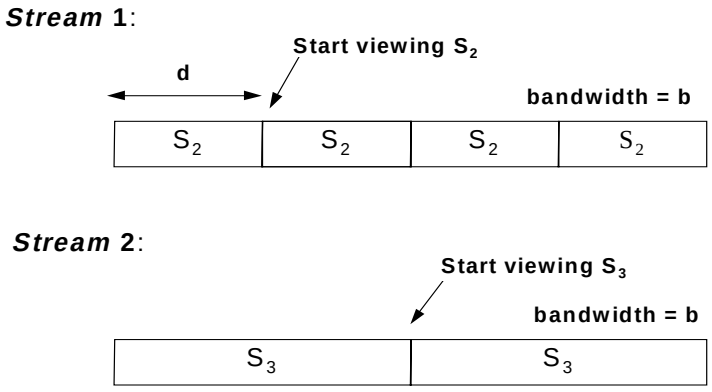


Figure 4: A representation of the first two streams for the Mayan Temple Protocol.

Let us now compute the durations of these segments. Since the first segment of the video $S_1$ will be preloaded, its duration will be equal to $d$. As seen on Figure 4, the second segment $S$ will have to be downloaded during this time interval and its duration will be an inverse function of its average bandwidth $b_2$. More precisely its duration $d_2$ will be given by the equation

$$F(d+d_2)-F(d)=bd$$

or

$$F(d+d_2)=F(d)+bd$$

where $F(t)$ is the cumulative bandwidth function of the video.

Since $F(t)$ is strictly increasing, it has an inverse function $F^{-1}(x)$ and we can write the solution of the previous equation as

$$d_2=F^{-1}(F(d)+bd)-d$$

Similarly, segment $S_3$ will have to be downloaded while segments $S_1$ and $S_2$ are being watched, that is over a time interval of duration $d+d_2$.

Consider now an arbitrary segment $S_i$ with $i \geq 3$. It will have to be downloaded while the previous segments are being played, that is, over a time interval equal to

$$d+\sum_{j=2}^{i-1} d_j$$

where $d_j$ represents the duration of segment $S_j$. Its own duration $d_i$ will be the solution of the equation

$$F(d+\sum_{j=2}^{i-1} d_j+d_i)-F(d+\sum_{j=2}^{i-1} d_j)=b(d+\sum_{j=2}^{i-1} d_j),$$

that is,

$$d_i=F^{-1}(F(d+\sum_{j=2}^{i-1} d_j)+b(d+\sum_{j=2}^{i-1} d_j))-d+\sum_{j=2}^{i-1} d_j.$$

The total bandwidth required to broadcast a given video will depend on the duration $D$ of the video, the length $d$ of its preloaded fragment and the bandwidth requirements of the video expressed either through its instantaneous bandwidth $b(t)$ or its cumulative bandwidth function $F(t)$.

There is a simple general method for evaluating the storage requirements of the protocol. Let $n$ be the number of segments in a video. When the video starts being played, the client will receive data from all $n$ channels and accumulate data on its disk drive. Consider what will happen when the customer will finish watching segment $S_{n-1}$ and start watching segment $S_n$. It will then have in storage the whole segment $S_n$ and will stop downloading any data from the video server. Assuming that the average consumption rate during the viewing of segment $S_{n-1}$ remained lower than the arrival rate of segment $S_n$ data, the maximum storage required by the protocol would be equal to the size of the last segment of the video, that is,

$$F(D)-F(D-d_n)$$

All these expressions simplify greatly if we assume that all segments have the same average bandwidth $b_S$. The duration $d_2$ of the second segment $S_2$ would then be given by the condition

$$b_S d_2 = bd\ ,$$

and we would have

$$d_2=\frac{bd}{b_S}.$$

More generally, the duration of the $i^{\text{th}}$ segment $S_i$ would be given by

$$b_S d_i = b(d+\sum_{j=2}^{i-1} d_j)\ ,$$

and we would have the recurrence

$$d_i = \frac{b(d + \sum_{j=2}^{i-1} d_j)}{b_S},$$

which simplifies into

$$d_i = d \sum_{j=1}^{i-1} (\frac{b}{b_S})^j.$$

As we can see from the preceding formula, the growth of the sizes of the successive segments is strongly affected by the ratio of the effective bandwidth $b$ of channels used to transmit the video to the average bandwidth $b_s$ of the segments. A higher ratio means that we will be able to pack more minutes of video into each segment and that their effective durations are increasing faster. Hence videos that mostly consist of slow-paced scenes will require less channels than faster-paced videos.

### 3.4. Implementing Partial Preloading

As we mentioned earlier, our two protocols assume that that the client STB will be able to preload on its disk drive the first $d$ minutes of the $N$ most popular videos. The task of distributing these data will be assigned to a single channel continuously broadcasting these first $d$ minutes according to a well defined schedule. We need now to describe how the protocol adjusts to changes in the set of videos being broadcast and how it allows the consecutive watching of two and more videos.

Any change in the set of videos being broadcast will require each STB to download the first $d$ minutes of the new videos being offered and to store them of its hard drive. Our protocol will thus need a mechanism allowing the VOD server to notify the STB's that they have new data to download but this mechanism could be as simple as agreeing upon some predefined time. There might also be a period of transition during which some of the videos that were previously programmed become unavailable while some of the new videos are not yet ready to be watched. This is not very different of what would happen with any other broadcasting protocol.

Handling customers who want to order a new video immediately after having watched another one is not much more difficult. Recall that the channel broadcasting the first $d$ minutes of the programmed videos will broadcast these according to a well defined schedule. If we make this schedule known ahead of time to each STB, any STB receiving the data for the various segments of a video being currently watched can store each of these segment data at the precise locations containing the video fragments that will be rebroadcast exactly after the data have been consumed. Hence, the STB will start replenish its cache of initial video fragments while the current video is still watched and will have its cache of videos full again when this video will end.

Handling customers who want to stop watching a video before its end and immediately order a new one is a more difficult problem because the STB will not have any time to replenish its cache with the data that were overwritten while it was downloading the various segments of the previous video. One possible solution would be to transmit the missing information *on demand*. This means that any STB that cannot find in its cache the first $d$ minutes of a video it is supposed to play can request the VOD server to sent immediately the missing data. The solution is not likely to require too much additional bandwidth as long as:

a) this customer behavior remains exceptional and

b) the sizes of the preloaded data remain small.

A second solution consists of broadcasting more frequently the initial fragments of the $N$ videos. One could, for instance, replace the single channel transmitting the first $d$ minutes of all $N$ videos by $N$ dedicated channels each transmitting the first $d$ minutes of a specific video. This solution offers the major advantage of making the preloading process *optional*: customers whose STB have a copy of the first segment of a video in their cache would continue to be able to watch that video without any delay while other customers would have to wait up to $d$ minutes. The price for this additional flexibility would one extra channel per video.

A third solution is based on the observation that disk capacities increase now much faster than either network or disk bandwidths. We could thus eliminate the problem by deciding that the preloaded segments should never be overwritten. This solution would double the disk space requirements of our two protocols but would not require any additional bandwidth.

## 4. DISCUSSION

Evaluating the bandwidth requirements of our two protocols is a difficult task because these requirements depend on the method used to deliver the preloaded parts of the programmed videos and the instantaneous bandwidth requirements of each video.

Figure 5 displays the bandwidth requirements of our two protocols with preloading. To eliminate the effect of the video duration $D$, the durations of the preloaded fragments on the $x$-axis are expressed as percentages of the video lengths. As in Figure 2, all quantities on the $y$-axis are expressed in standard video channels. To obtain these values we had to make two significant hypotheses:

a) we neglected to include the bandwidth required to broadcast the initial fragments of the programmed videos as this bandwidth depends on the number $N$ of videos being programmed;

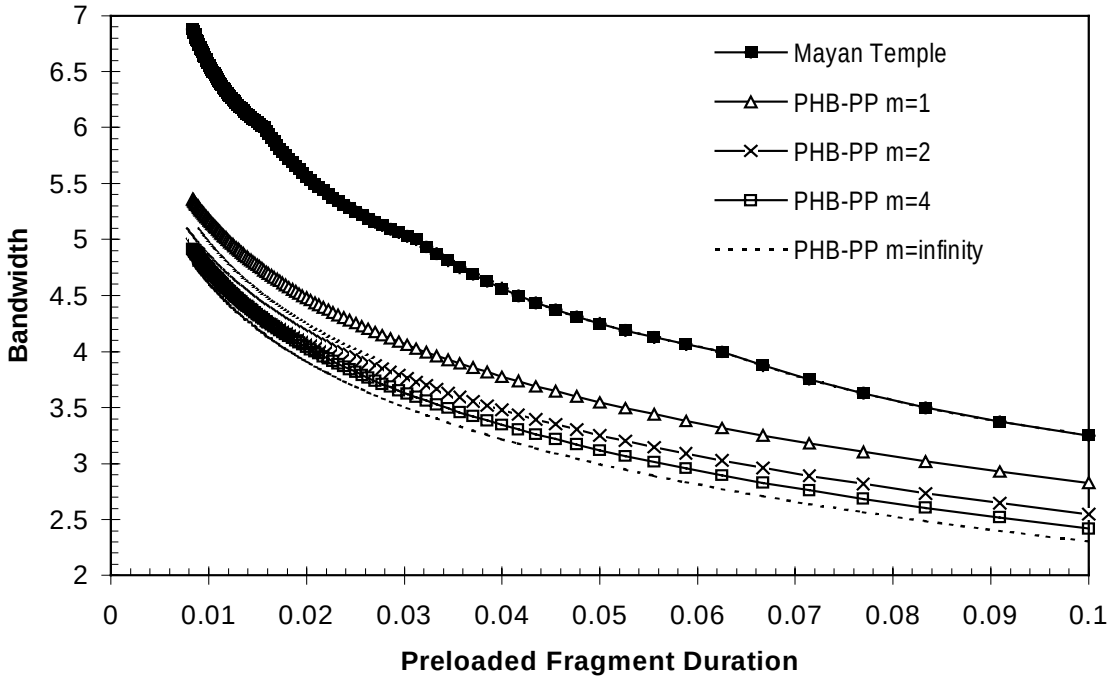


Figure 5: Bandwidth requirements of the two broadcasting protocols with partial preloading

b) we assumed that the average bandwidth requirement $b_S$ of the video being broadcast was equal to a standard video channel (that is $b_S = b$); this is a very conservative hypothesis as a standard video channel transmitting compressed video will require enough bandwidth to accommodate the highest bandwidth scenes of all videos being broadcast and we should have $b_S < b$.

We did not include any bandwidth data for other broadcasting protocols because these protocols cannot offer true zero-delay access to the videos they broadcast. While these protocols can provide arbitrarily small delays, smaller delays are always achieved at the cost of higher aggregate bandwidths. For instance, no harmonic protocol can achieve a maximum response time of less than one minute for a two hour video for less than the equivalent of 5.12 standard video channels. Similarly, the Pyramid Broadcasting protocol with $\alpha = 2$ would require 8 video channels to bring the maximum waiting time of the same video below one minute.

Several lessons can be learned from these data. First the PHB-PP protocol performs much better than the Mayan Temple protocol. This should be expected as the PHB-PP protocol partitions each video into many more segments than the Mayan Temple protocol and broadcasts each segment at the minimum bandwidth required to have it delivered on time. In contrast, the Mayan Temple protocol uses much larger segments and broadcasts each of them at the minimum bandwidth required by the initial part of each segment on time.

Second the bandwidth requirements of the PHB-PP protocol decrease when the number of segments increases. This was also expected as having more segments allows the protocol to tune more finely the bandwidths allocated to each segment. This effect becomes quickly negligible as the ratio $m$ of the duration $d$ of the preloaded fragment of each video to the duration of an individual segment $d_S$ becomes greater than four.

Finally, the bandwidth requirements of both protocols strongly decrease when the size of the preloaded fragment of the video increases. Consider for instance a two hour video whose first three minutes are preloaded. Broadcasting that video under the PHB-PP protocol would require the equivalent of 3.75 standard video channels while broadcasting it under the Mayan Temple Broadcasting protocol would require 5.25 channels. Doubling the size of the preloaded fragment would bring the bandwidth requirements of the PHB-PP protocol down to the equivalent of 3.12 standard video channels and those of the Mayan Temple Protocol down to 4.25 video channels. This is because the first few minutes of a video have to be retransmitted much more frequently than the other parts of the video. Hence preloading more of these early minutes will have a very significant impact on the total bandwidth consumption of the video.

This last observation suggests a potential improvement to our two protocols. We had assumed earlier that the only disk space that

was available to store the initial fragments of all programmed videos was the space that each video would occupy while being watched by the customer. One very simple way to improve the performance of our two protocols would be to allocate more space to these fragments as this would allow to store larger fragments of more videos. Current trends in disk drive technology are making this proposition very feasible as the disk capacities of the least expensive disk drives are expected to double every year.

Another approach to reduce the bandwidth requirements of our protocols would be to include a few minutes of video previews at the beginning of each program. This would give to the protocols more time to download the various segments of the video being watched at a minimum cost as we could have one single set of trailers updated every few hours.

## 5. CONCLUSIONS

One of the most promising approaches to reduce the cost of video-on-demand services is to broadcast continuously the most frequently requested videos. With the sole exception of staggered broadcasting, all extant broadcasting protocols assume that the client set-top box has enough storage space to store between 40 and 50 percent of the duration of each video. We have shown how this space could be used to anticipate the customer requests by preloading the first few minutes of the videos being broadcast. This would provide instantaneous access to these videos. We have also shown how the same approach could be used to reduce the extra bandwidth required to handle compressed videos.

We have presented two new broadcasting protocols that use partial preloading to eliminate this extra bandwidth while providing instantaneous access to the videos being broadcast. Our first protocol, Polyharmonic Broadcasting with Partial Preloading partitions each video into between 20 and 160 segments of equal duration and allocates a separate data stream to each individual segment that was not preloaded. As we saw, this approach results in a very low aggregate bandwidth. Our second protocol, the *Mayan Temple Broadcasting* protocol, uses segments of increasing sizes and dedicates a full video channel to each of these segments. It never uses more than 8 to 10 data streams but requires between 40 and 45 percent of additional bandwidth.

## ACKNOWLEDGEMENTS

This research was partially supported by the Office of Naval Research under Grant N00014-92-J-1807 and by the National Science Foundation under grant PO-10152754.

We wish to thank the anonymous referees for their numerous suggestions.

## REFERENCES

[1] Aggarwal, C. C., J. L. Wolf, and P. S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. Proc. International Conference on Multimedia Computing and Systems, pages 118–26, June 1996.

[2] Beran, J., R. Sherman, M. Taqqu, and W. Willinger. Long-range dependence in variable bit-rate video traffic. IEEE Transactions on Communications, 43:1566–1579, 1995.

[3] Dan, A., P. Shahabuddin, D. Sitaram and D. Towsley. Channel allocation under batching and VCR control in video-on-demand systems, Journal of Parallel and Distributed Computing, 30(2):168–179, Nov. 1994.

[4] Dan, A., D. Sitaram and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In Proc. ACM Multimedia Conference, pp. 15–23, Oct. 1994.

[5] Dan, A., D. Sitaram, and P. Shahabuddin. Dynamic batching policies for an on-demand video server. Multimedia Systems, 4(3):112–121, June 1996.

[6] Eager, D., M. Ferris and M. Vernon.. Optimized regional caching for on-demand data delivery. In Proc. 1999 Multimedia Computing and Networking Conference (MMCN'99), San Jose, CA, Jan. 1999.

[7] Garrett, M. and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In Proc. ACM SIGCOMM '94 Conference, pages 269–280, Aug. 1994.

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

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

[10] Pâris, J.-F., S. W. Carter and D. D. E. Long. Efficient broadcasting protocols for video on demand. In Proc. 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98), pages 127–132, July 1998.

[11] Pâris, J.-F., S. W. Carter and D. D. E. Long. A Low bandwidth broadcasting protocol for video on demand. In Proc. 7th International Conference on Computer Communications and Networks (IC3N'98), pages 690–697, Oct. 1998.

[12] Sen, S., J. Rexford, and D. Towsley, Proxy prefix caching for multimedia streams. Proc. IEEE INFOCOM '99, March 1999.

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

[14] Wong, J. W. Broadcast delivery. Proceedings of the IEEE, 76(12), 1566–1577, Dec. 1988.