Video-on-Demand Broadcasting Protocols: A Comprehensive Study

Ailan Hu
Intel Corporation
2200 Mission College Blvd
Santa Clara, CA 95052
ailan.hu@intel.com

Abstract—

Broadcasting protocols are proved to be efficient for transmitting most of the popular videos in video-on-demand systems. We propose a generalized analytical approach to evaluate the efficiency of the broadcasting protocols and derive the theoretical lower bandwidth requirement bound for *any* periodic broadcasting protocols. By means of the proposed analytical tool - temporal-bandwidth map, the approach can be used to direct the design of periodic broadcasting protocols to achieve different goals, *e.g.,* server bandwidth requirement, client waiting time, client I/O bandwidth requirement etc. As the most important performance index in VOD system is the required server bandwidth, we give the solution to achieve the optimal bandwidth efficiency given client waiting time requirement and the length of the video. To take into account the popular compressed video with variable bit rate, the optimal approach is applied readily to the VBR videos and can achieve zero loss and best bandwidth efficiency. We give proof why existing techniques such as smoothing and prefetching is not necessary and in some cases inefficient in broadcasting protocols. We also discuss how broadcasting schemes can be tailored to support true and interactive VOD service. An insightful comparison between broadcasting and multicasting schemes is also given in this paper.

I. Introduction

Video-on-Demand (VOD) proposes to provide subscribers with the possibility of watching the video of their choice at the time of their choice, as if they were watching a rented video cassette. So far VOD has not been a commercial success because the technology is still very expensive and its potential users are unwilling to pay much more for a VOD selection than they are used to paying for a video cassette rental.

Previous research has shown that performance of VOD system can be greatly improved through the use of multicast or broadcast schemes. Most of the multicast protocols [16], [17], [18], [19], [20] are *reactive* in the sense that they transmit data in response to the user requests. Multicast schemes try to let user share the same stream of data as much as possible. While some of the multicast approaches can provide immediate service and save server bandwidth by avoiding unnecessary transmission of data, they are subject to data loss and can not guarantee on time delivery of data if user requests are bursty or too high. Broadcasting schemes can address this problem by periodically transmitting video segment in a *proactive* way and guarantee service latency within certain amount of time.

The idea behind periodic broadcasting schemes is to divide the video into a series of *segments* and broadcast each segment periodically on dedicated server channels. While user is playing the current video segment, it is guaranteed that the next segment is downloaded on time and the whole video can be played out continuously. User will have to wait for the occurrence of the first segment before they can start playing the video. User waiting time is usually the length of the first segment. A *true* VOD service does not require user to wait for the video. Since user can not watch the video immediately, broadcasting protocols can only provide *near* VOD service.

In the literature, researchers were trying to find out the *golden factor* to divide the video to achieve the lowest server bandwidth while still guarantee on time delivery of each segment. In this paper, we propose an insightful analysis for VOD broadcasting protocols and give the solution to this golden factor. A convenient tool, the *temporal-bandwidth map*, is provided to evaluate broadcasting protocol performance efficiency. The theoretical lower server bandwidth requirement for any broadcasting protocols is derived and the optimal broadcasting schemes to achieve the lowest server bandwidth given certain user waiting time and number of segments are presented. It is also shown that our approach can be readily tailored to analyze variable bit rate (VBR) based videos. As a result, the VBR video is transformed to constant bit rate (CBR) streams to be broadcast. We show that smoothing techniques in broadcasting protocols could not achieve better performance for saving server bandwidth. The proposed approach can also be utilized to design broadcasting protocols to satisfy other performance requirements such as client I/O bandwidth and client storage constraint. We also discuss how broadcasting protocols can be modified to provide true VOD service and the possibility of interactive functions, which are rarely mentioned for broadcasting protocols in previous research. We also show the relationship between broadcasting and multicasting schemes and how multicasting schemes degenerate to broadcasting schemes.

Throughout the paper we use S to denote the total length (in time units) of the video, S_i the ith segment, w the client waiting time requirement, b the video consumption or display rate and n the number of segments for a given video.

II. Overview of the Periodic Broadcasting Protocols

Staggered broadcasting [15] is the simplest broadcasting protocol proposed in the early days. It allocates K server channels each with bandwidth b to transmit the whole video. The beginnings of each video replica are staggered evenly across the channels. Client access latency is S/K and it could not be improved without the expense of linear increase in the corresponding server bandwidth.

Some more efficient broadcasting protocols have been proposed. All these protocols share a similar organization. They divide each video into n *segments* that are simultaneously broadcast on different data streams (logical channels). One of these streams transmits nothing but the first segment of the video. The other streams transmit the remaining segments at their designated bandwidth. When users want to watch a video, they wait for the beginning of the first segment from the first stream. While they start watching that segment, their set-top box (STB) starts downloading enough data from the other stream(s) so that it will be able to play each segment of the video in turn.

These broadcasting protocols can be subdivided into three groups. Protocols in the first group partition the video into increasing size of segments and transmit them in logical channels of same bandwidth. They are based on Viswanathan and Imielinski's *Pyramid Broadcasting* (PB) protocol [1]. The segment size of the videos in this protocol follows a geometrical series and different videos are mingled together in each logical channel. To provide on time delivery of the videos, each segment channel has to transmit the segments in a very high rate and client I/O bandwidth and storage requirement are also high. Clients can download the next segment at its earliest occurrence and at any time they download at most two consecutive channels. To address the problem of high client side requirement in PB, *Permutation-based Pyramid Broadcasting* (PPB) was proposed. Instead of transmitting a segment in a very high bandwidth, PPB multiplexed the segment channel into P subchannels and transmitted them in P times lower rate. The P substreams are staggered with each other to meet the same timing requirement as in PB. Another important protocol in this family is the *Skyscraper Broadcasting* protocol (SB) [3]. SB transmits each segment in the video consumption rate b and its segment series progression is much lower but still meet the timing requirement. User needs to download from at most two streams at any time. The client disk storage requirement is constrained by the size of the last segment. A more efficient broadcast scheme, the *Fast Broadcasting* (FB) [7] is proposed to divide the video into geometrical series of [1, 2, 4, $\ldots 2^{K-1}$, 2^K]. The channel bandwidth is b and client needs to download from all K streams simultaneously. This protocol is the most efficient in terms of server bandwidth requirement.

It is the *Harmonic Broadcasting* (HB) [4] initiated another group of broadcasting protocols. They divide the video into equal size segments and transmit them in logical channels of decreasing bandwidth. The playout duration of a segment is defined as a *slot*. In HB, each segment is broadcast repeatedly on its dedicated channel with a bandwidth b/i. This requires much less server bandwidth than the PB family protocols. Client starts receiving data from *each* segment stream right after it can start downloading the first segment. When the client is ready to consume segment S_i, it will have received $i-1$ slots of data from that segment and the last slot of that segment stream can be received during the segment playout time. HB requires the client receiving bandwidth the same as the server transmission rate and the storage requirement about 37% of the entire video. The major flaw in HB is that it can not always deliver all the data on time. Paris et. al. proposed the *Cautious Harmonic* (CHB), the *Quasi-Harmonic* (QHB) [5] and the *Polyharmonic* (PHB) [6] protocol and solved this problem. These protocols provide almost the same performance results as HB while the timing requirement is still met. To further reduce the server bandwidth, HB-based schemes need to divide the video into more segments, this requires more logical channels.

A new family of the broadcasting protocol includes *Pagoda* broadcasting [8] and *New Pagoda* [9] broadcasting schemes. These protocols are hybrid of pyramid-based protocols and harmonic-based protocols. They partition each video into fixed size segments (as in HB) and map them into a small number of data streams of *equal bandwidth* (as in PB) and use time-division multiplexing to ensure that successive segments of a given video are broadcast at the proper decreasing frequencies (as in HB). The result is that they do not require significantly more bandwidth and at the same time do not use more logical streams or less segments than HB-based protocols.

All the above protocols are based on the assumption that the videos are Constant Bit Rate (CBR) encoded. In the real world there are many videos with Variable Bit Rate (VBR). Some more protocols are proposed to address this problem. Saparilla et. al. proposed a protocol in [12] (we will call it VBR-B) using the segmentation scheme in FB and applied the techniques of GoP smoothing, server buffering and client prefetching to improve its performance. Another protocol, the *Trace Adaptive Fragmentation* (TAF), proposed in [13] is an improvement of VBR-B to take into account the trace of each video and try to obtain lower aggregate bandwidth by comparing possible combinations of feasible video schedules. In [11] Paris proposed a VBR broadcasting protocol (VBHB) based on the cautious-harmonic protocol. This protocol does not use the smoothing technique as the other protocols for VBR videos. As a broadcasting protocol, it requires predetermined server bandwidth.

III. A General Analysis of Broadcasting Protocols

A. The Generalized Analytical Approach

To analyze the efficiency of the broadcasting protocols, we observe that there are three notions very important: segment size progression, bandwidth allocation for each logical channel (data stream to transmit a segment), and the satisfaction of continuous play condition. To describe the schemes with these notions, we introduce a *temporal-bandwidth map*, whose x-direction represents time, and y-direction represents bandwidth. On the lower part of the map is the *broadcasting area*. Each logical channel is

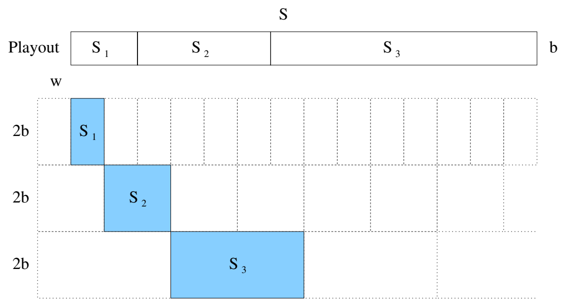


Fig. 1. Temporal-Bandwidth Map of Pyramid Broadcasting with $\alpha = 2$

presented in this part and its corresponding bandwidth is represented by its height. These logical channels are piled up together and the sum of their heights is equal to the server bandwidth requirement. The upper part of the map is the playout of the video segments corresponding to each of the broadcasting segment in the broadcasting area. We call this part the *playout area*.

Figure 1[1] is the analysis of pyramid broadcasting protocol using temporal-bandwidth map. The lower part (broadcasting area) corresponds to broadcasting of segments in pyramid broadcasting scheme. The shaded area of S_1, S_2, S_3 in the broadcasting area corresponds to the playout of segment S_1, S_2, S_3 in the playout area. From the time when user starts waiting to the end the whole video is played out, only the shaded areas are useful in the broadcasting area.

To measure the efficiency of the broadcasting protocols, we define broadcasting *bandwidth efficiency* as:

$$\text{server bandwidth efficiency} = \frac{\text{area of the playout area}}{\text{area of broadcasting area}}$$

Since the playout area is equal to the shaded segments in the broadcasting area, we can see that the efficiency is actually the fraction of the useful bits for playout in the broadcasting area to the total bits broadcast from the time user starts waiting to the end of the video playout.

One of the design goals of all broadcasting protocols is thus to maximize the bandwidth efficiency, in other words, reduce the sum of the bandwidth of each logical channels, *i.e.*, the height of the broadcasting area, and at the same time, meet the access time requirement and client I/O bandwidth and storage requirement.

B. Advantage of Separating Different Videos

Before applying the generalized analytical approach to analyze existing broadcasting protocols, we digress for a moment to show the advantage of *separating different videos*. In PB, all the videos are considered and they are broadcast sequentially in each logical channel. Each channel has bandwidth B (here we use B as bandwidth for each logical channel instead of total bandwidth for a video) and contains segments from M videos.

[1]The map provided here only considers one video. See the argument in the next section why we only consider one video here.

Clients have to download at speed B and only $1/M$ of the channel cycle is the video they want to watch. A slight change to the protocol could be made to achieve the bandwidth of B/M for each segment: separate the M video segments in one logical channel into M logical channels and let each segment transmit at a slower speed with bandwidth B/M. This slower transmission scheme is involuntarily adopted in all of the later protocols and contributes part to their better performance results. The point is that we can consider each video independently. Each of them is broadcast in its allocated bandwidth in its *own* channels. To mix them with other videos and transmit them serially can only force higher client I/O bandwidth and hence higher client storage requirement. We will also show in a later section that there is no performance gain to aggregate VBR videos together. Thus far, we will consider only one video.

C. Analysis of Existing Broadcasting Protocols

Now we compare the pyramid-based and harmonic-based protocols using the proposed approach. We will give the analysis of the other hybrid protocols in a later section. Without loss of generality, we set the access latency w as 1 and the total video duration S is a relative value to w. Note also that the bandwidth requirement for each logical channel is proportional to the consumption rate b, its value is not important and we can also assume it to be 1 for convenience. The comparison here is based on the same S/w ratio, the required bandwidth for each protocol is shown by the height of the broadcasting area.

Pyramid broadcasting protocol partitions each video into K segments of geometrically increasing sizes. The geometric series has factor α, where $\alpha > 1$. Each logical channel is now allocated with bandwidth B/K, where B is the total bandwidth allocated to this video. Figure 1 gives an example of pyramid broadcasting with $\alpha = 2$. Since the playout time of the first segment must be at least the broadcasting time of the second segment to guarantee on time delivery, and the first segment is $1/\alpha$ of the size of the second, the broadcast time of the first segment must be $1/\alpha$ of its playout time. So the bandwidth requirement for the first logical channel must be at least α times of the consumption rate b. We can see the bandwidth allocated for the whole video is quite large.

Skyscraper broadcasting scheme allocates fixed bandwidth - the consumption rate b for each logical channel. Its segment size is determined by a recursive function, whose materialized series is as follows:

$$[1, 2, 2, 5, 5, 12, 12, 25, 25, 52, 52, \ldots].$$

The intuition in this series is that beginning of any next segment must be encountered before the current segment is consumed. Since its bandwidth requirement is not as demanding as pyramid broadcasting, the broadcasting area is not as high as that in PB, and its segment is somewhat spread out during its playout time. Figure 2 shows one scenario of how skyscraper broadcasting works to achieve the same maximum latency time and play the same length of video as in PB. Note the *bandwidth efficiency* for

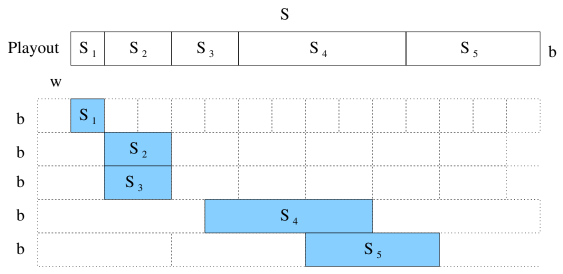
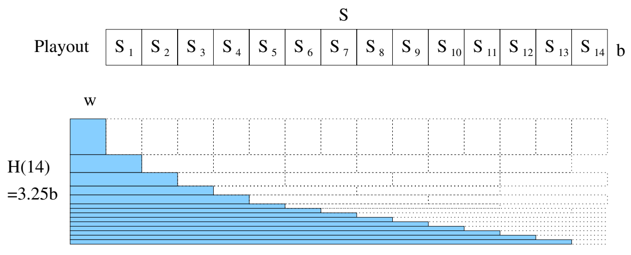


Fig. 2. Temporal-Bandwidth Map of Skyscraper Broadcasting



Not to scale

Fig. 3. Temporal-Bandwidth Map of Polyharmonic Broadcasting

the protocol is the same no matter what scenario is. Compared to pyramid broadcasting, SB has higher efficiency, as can be easily observed from its shaded area in the broadcasting area.

The major contribution for harmonic broadcasting is its low bandwidth requirement. The underlying mechanism for its better performance was not addressed in the original paper, but after applying temporal-bandwidth map to harmonic broadcasting protocol, it is clear that it actually managed to achieve a higher *bandwidth efficiency*: its shaded area has higher occupancy in the broadcasting area.

CHB and QHB solved the problem of the flaw in HB that could not always deliver the video segment on time. But observed in temporal-bandwidth map, they have less bandwidth efficiency. In CHB, if we transform the second and third segment into the equivalent 1/2 height and 2 slots length, starting from the third segment, each segment stream does not stretch as far as its counterpart in HB and thus has higher height and less efficiency. In QHB, every segment has some portion of redundancy and thus not as efficient. A better protocol similar to HB is the polyharmonic broadcasting protocol. The novel point in this protocol is to download the video right away when the user switches in. Although the segment is not downloaded from the beginning, it can be reassembled to form a whole segment and it is guaranteed that by the time the previous segment has finished playing, the next segment is downloaded. The same maximum access latency can be achieved with the same server available bandwidth as harmonic broadcasting, but PHB does not have the on-time delivery problem.

Figure 3 gives an example of how polyharmonic broadcasting

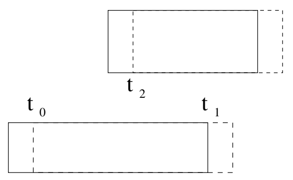


Fig. 4. right-edge restriction of the broadcast segment

scheme works [2]. Again, it is easy to observe that polyharmonic broadcasting tries to stretch its "useful" segment to get better efficiency.

IV. Optimization of Server Bandwidth Efficiency

The above analysis gives rise to the following observations as rules to design efficient broadcasting protocols:

Rule 1: Any efficient periodic broadcasting protocol should not repeat transmitting another cycle of the segment during the period when the user starts to wait the video until this part of information is actually *started* to be played out.

Proof: Observing the temporal-bandwidth map, appearance of a repeated cycle means that there must be some part of the cycle that is not in the shaded area (the useful part) before the time when the segment is played out. This implies there exists unneeded cycle(s) during the downloading time of the segment. Such protocols are not efficient since they waste part of the bandwidth to transmit the same data that have been already received. Note that we restrict the downloading of a whole segment to end before the *start* of its playout. This is necessary to meet the timing requirement. Figure 4 shows a broadcast segment on the lower part and its playout in the upper. Since user can switch in at any time t_0, the beginning of the next broadcast cycle t_1 should appear as early as the starting playout time of this segment, t_2. In other words, the right edge of the broadcast segment should not stretch further right to the starting playout time of this segment, otherwise the timing requirement will not be met, as is the case in harmonic protocol. We define the duration from the time when user switches in to the time when a segment starts to playout as the *good segment downloading time*. ■

Examples of inefficient protocols that violate this rule are pyramid broadcasting, permutation-based pyramid broadcasting, skyscraper broadcasting, and cautious harmonic broadcasting protocols. All of them have more than one cycle during their good segment downloading time. Harmonic broadcasting violates the timing requirement enforced by this rule.

Rule 2: Any efficient periodic broadcasting protocol should not include, in any of its cycle, unneeded portions of data for just in time playout of the video.

Proof: To include unneeded portion of information in any of its cycle means that the shaded area is somewhat interleaved with portions of blank area which include useless information. Such protocols will not be efficient. Quasi-harmonic, pagoda

[2]Here we use $m = 1$, a larger m could be used to achieve lower bandwidth

and new pagoda broadcasting are examples of such protocols that conflict with this rule. ■

Rule 3: Any periodic broadcasting protocol needs to repeat cycles for each segment within their good downloading time.

Proof: This is a requirement for any periodic broadcasting protocol. To meet the access latency requirement, repeated cycles are needed to broadcast in the network and to let users download and play the segments within certain waiting time. Each segment should repeat transmitting the necessary data at least during its good download time, otherwise this segment of data can not be delivered on time. We can cut all of the unneeded cycles or part of them because users actually do not want them for continuous playout, but we can not avoid broadcasting their next cycles in order to allow any user to switch in at any time. ■

The essence of these observations is to let the shaded area stretch as far to the left and right as possible so that the height of the broadcasting area, *i.e.*, the required bandwidth, is minimized. To the left means the starting downloading time is the time the client switches in. To the right means to let the downloading period as long as possible, to the point when the segment is needed to be played out - this is essentially to let the segment arrive *just in time*.

An interesting family of broadcasting protocols is the pagoda protocol and its variants. Their segments have equal size. Different segments can map to the same broadcast stream, but each segment only occupies part of the broadcast stream cycle. Each stream broadcasts at consumption rate b. The ith segment is broadcast at frequency close to but at least 1 every i slots. To analyze its efficiency, a *normalization* transformation can be performed by replacing the segment in the broadcasting area with a segment of length corresponding to reverse of the segment broadcast frequency and height calculated from the area of the segment in the playout area. The normalized map has one stream per segment. It can be shown that pagoda protocol has portion of data wasted during its good segment downloading time and is not as efficient as polyharmonic protocol in terms of bandwidth requirement.

The discussion above shows that polyharmonic protocol is the only protocol meets all the necessary conditions for efficient broadcasting protocols. We define the protocols that meet the three rules the *optimally-structured broadcasting protocols*. Optimally-structured protocols can have different video information encoding schemes which could be different from polyharmonic broadcasting protocol, but they can be normalized to a structure similar to polyharmonic protocol. Note that unlike polyharmonic broadcasting, optimally-structured schemes can have different segment size as well as different segment broadcasting bandwidth. The key restriction is the *area restriction*, *i.e.*, the segment area in the playout area should be equal to the one in the broadcasting area.

The structure of optimally-structured broadcasting schemes is illustrated in Figure 5. Every segment is downloaded during its good segment downloading time. Every broadcast segment

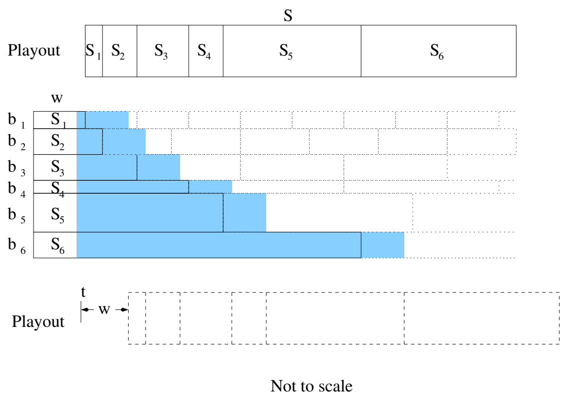


Fig. 5. Optimally-Structured Schemes

(shaded area) contains no portion of replicated information. Areas $S_1, S_2, S_3, \ldots$ in the playout area and broadcasting area are of the same size respectively.

Timing requirement can still be met as long as we keep the corresponding segment area restriction in mind. Figure 5 illustrates that starting from any time t, playout of segments (corresponding to the lower playout area, their corresponding segments in the broadcasting area are those shaded segments) have the same structure as those segments starting from the same time (corresponding to the solid line rectangles in the broadcasting area, their playout is in the upper playout area). So when analyzing these schemes, we can assume that all the segments in the broadcasting area are started from the same time.

A. Optimization of CBR Broadcasting Schemes

We observe that segment size and channel bandwidth in optimally-structured schemes can be different. The optimization problem is thus to choose the optimal segment progression among the optimally-structured schemes. Given segment number n, how can we partition the video to achieve the lowest bandwidth? Formally, let b_i be the channel bandwidth for the ith segment, S_i be the segment size, $w = 1$ and $b = 1$, the problem can be stated as:

$$\text{minimize} \sum_{i=1}^{n} b_i$$

subject to

$$b_i(1 + \sum_{j=1}^{i-1} S_j) = S_i \quad i = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n} S_i = S$$

$$S_i > 0, b_i > 0$$

Note that

$$(1 + b_i) = \frac{1 + \sum_{j=1}^{i} S_j}{1 + \sum_{j=1}^{i-1} S_j} \quad i = 1, 2, \ldots, n$$

Multiplying all n number of $(1+b_i)$ produces:

$$\prod_{i=1}^{n}(1+b_i) = 1 + \sum_{j=1}^{n} S_j = S+1$$

That is, the product of the n variables $(1+b_i)$ is a constant, $S+1$. To minimize $\sum_{i=1}^{n} b_i$ is the same as minimize $\sum_{i=1}^{n}(1+b_i)$. The latter is minimized when $(1+b_i), i = 1,2,\ldots,n$ are equal to each other, given their product is a constant. This means $\sum_{i=1}^{n} b_i$ is also minimized when b_i are equal to each other, say b^*. Consquently,

$$(1+b_i)^n = S+1$$

or

$$b_i = b^* = \sqrt[n]{S+1} - 1 \tag{1}$$

the bandwidth requirement is

$$\sum_{i=1}^{n} b_i = nb^* = n(\sqrt[n]{S+1} - 1) \tag{2}$$

and the optimized segment size progression is

$$\begin{aligned} S_i &= b^*(1+b^*)^{i-1} \\ &= (\sqrt[n]{S+1} - 1)(\sqrt[n]{S+1})^{i-1} \end{aligned}$$

The segment progression follows a geometrical sequence. The *golden factor* for segment progression is thus $\sqrt[n]{S+1}$, here S is a relative value to access latency w. The golden factor is solely determined by the duration of the video, the required access latency and the number of segments. The original idea is proposed by Hu et. al. in [10]. The protocol is called *greedy equal bandwith broadcasting* (GEBB). Like PB, the optimized scheme has equal bandwidth broadcasting stream and geometrically growing segment size; unlike PB, it downloads each segment *greedily, i.e.*, immediately after user switches in. Interestingly, for a video with S/w equal to 127, the golden factor is 2. The Fast Broadcasting protocol uses this factor but it is not optimal if S/w has other values.

B. Optimization of VBR Broadcasting Schemes

So far we limited our discussion in CBR broadcasting. A similar approach can be applied to the compressed video or a VBR video. The playout area of the video will have variable bit rate in the temporal-bandwidth map and its broadcasting segment is still of constant bit rate. Area restriction still applies. The optimal partition of the video is a problem that can be formalized in a similar manner as the previous section. This is a non-linear optimization problem with n linear inequality constraints, we can construct a Lagrangian function and solve it using Kuhn-Tucker necessary conditions. In theory, a series of n equations need to be solved to obtain the optimal partition.

A more practical dynamic programming approach is presented in Figure 6. The idea is to construct a table B_{min} starting from one segment and video length of one frame, then go on step by step for larger segment numbers and larger video length. The given arguments are: client waiting time requirement w, total video length N frames, the ith frame size $f[i]$, and frame consumpution rate F frames/sec. $B_{min}[j,i]$ indicates the minimum bandwidth requirement for a video with j frames and i segments. It is calculated from the minimum bandwidth requirement for $i-1$ segments and video length k frames ($B_{min}[k, i-1]$), where k is less than j frames and larger or equal to $i-1$ frames [3]. $\frac{A[j]-A[k]}{w+k/F}$ is the bandwidth requirement for a segment with frames from k to $j-1$. $P_{min}[j,i]$ saves the starting frame position of the ith segment with video length of j frames. The computational complexity of this algorithm is in $O(nN^2)$ and the space complexity is in $O(nN)$. In any case when the complexity is inhibiting, a ready way to reduce it is to relax the frame granularity, or combine several frames together to form a larger "frame".

```
// Initialize table $B_{min}$ to infinity
for ($i = 1; i \le N; i++$)
    for ($j = 1; j \le n; j++$)
        $B_{min}[i,j] = \infty$

// Initialize $A[i]$ to be the accumulated frame
// sizes from the first frame to the $i$th frame.
// Initialize the required bandwidth for one
// segment and $i$ frames of video length.
$A[0] = 0;$
for ($i = 1; i \le N; i++$)
    $P_{min}[i,1] = i;$
    $A[i] = A[i-1] + f[i];$
    $B_{min}[i,1] = \frac{A[i]-A[0]}{w};$

// Construct the rest of the table
for ($i = 2; i \le n; i++$)
    for ($j = i; j \le N; j++$)
        for ($k = i-1; k < j; k++$)
            $B' = B_{min}[k, i-1] + \frac{A[j]-A[k]}{w+k/F};$
            if ($B' < B_{min}[j,i]$)
              $B_{min}[j,i] = B';$
              $P_{min}[j,i] = k;$
```

Fig. 6. Pseudocode for the dynamic programming solution

Due to the design of the optimally-structured scheme, the VBR video is naturally transformed to CBR streams to broadcast in the network and it is inherently not subject to loss. It is shown that the optimal solution out-performs the existing VBR broadcasting schemes significantly. Experiments with different video traces using the optimal approach indicate the later segments in VBR video follow a similar geometric series as a CBR video. This is because the later segments tend to be large and include more frames. The average segment playout rates in later

[3] $B_{min}[k, j-1]$ with $k < j-1$ has no meaning since frame number is always no less than segment number.

segments are almost the same and this makes the optimal segment division similar to that of CBR. A simplified algorithm can be constructed based on the above observation to calculate a near optimal video division.

V. Theoretical Bandwidth Requirement Bound for Any Periodic Broadcasting Protocol

Given the required access latency w and video length S, what is the server bandwidth lower bound requirement for any periodic broadcasting protocol? From the discussion above, we see that only the optimally-structured broadcasting schemes are designed in the way that does not waste server bandwidth. Any other schemes can be transformed to a similar structure with each logical channel transmitting a portion of data from the video, whether this cycle of data is transmitted during its good downloading time will decide if it's optimally-structured or not.

What if we change the shape of the video itself while still let it be played out on time? This corresponds to change the shape of the playout area, with part of certain segment(s) moved to some previous[4] segment(s). This is essentially a kind of *smoothing* technique. Figure 7 shows why smoothing incurs more bandwidth demand. Suppose part of segment S_i is smoothed to its previous segment S_{i-1}. The bandwidth requirement for the $(i-1)$th broadcast channel will be increased. To compensate the lost area A_2, the combined two new channels' bandwidth will be greater than that before smoothing, *i.e.*, $b_{i-1} + b_i < b^{'}_{i-1} + b^{'}_{i}$. The same argument can be applied to segment S_{i-2}, S_{i-3} .... This implies that *smoothing by segment rearrangement does not help lowering the bandwidth requirement*, it can only make it worse. We also see why *just in time* is so desirable to reduce the bandwidth requirement. The earlier the part of the segment arrives ahead of the time when it is needed, the less efficient the protocol is.

We conclude that optimally-structured broadcasting scheme without changing the playout shape is the only candidate to achieve lower bandwidth requirement. The essence of the optimally-structured scheme is that any segment is downloaded and *only* downloaded during its good segment downloading time, and during this time period, there should not be any other portion of the segment included in this repeated cycle. The theoretical server bandwidth lower bound is reached when the optimally-structured schemes have an infinity number of segments as is shown by the following theorem.

Theorem 1: Given a video with required access latency of w and length of S, there exists a theoretic bandwidth requirement lower bound B_0. This bound is reached when the optimally-structured broadcast scheme has infinity number of segments. A CBR video with consumption rate b has the bound of $b \cdot ln(S/w+1)$. A VBR video with video consumption rate $b \cdot f(t)$ has the bound of $\int_0^S \frac{b \cdot f(t)dt}{w+t}$.

Proof: Suppose the video is partitioned into n segments (does not matter if it's optimally partitioned or not) and its re-

[4]Normally moving to latter segment is not feasible since timing requirement is violated.

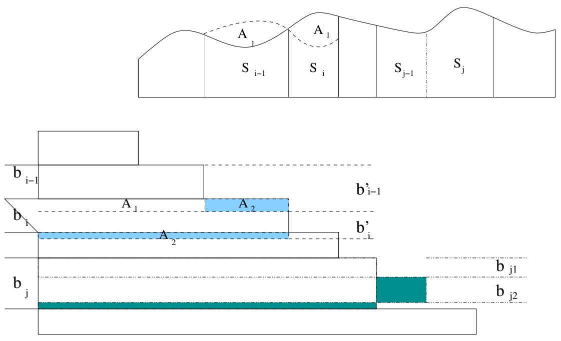


Fig. 7. The downside of smoothing and the upside of splitting

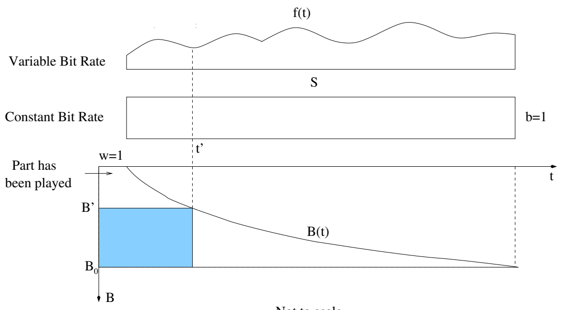


Fig. 8. Bandwidth Lower Bound Asymptote and Client Storage Requirement

quired bandwidth is $B(n)$. Figure 7 shows how further partition of any segment S_j into S_{j1} and S_{j2} results in lower bandwidth requirement. b_j must be larger than sum of b_{j1} and b_{j2}. Let every segment split and we get $2n$ segments, while $B(n) > B(2n)$. When $n \to \infty$, $\sum_n b_i$ monotonically decreases, but it has a lower bound. This concludes that series $B(n)$ converges, *i.e.*, the lower bound exists and is reached when segment number tends to inifinity. We denote its limitation as B_0. In the case of CBR video, B_0 can be calculated from the optimal segment bandwidth b^*:

$$\begin{aligned} B_0(CBR) &= lim_{n\to\infty} nb^* \\ &= lim_{n\to\infty} nb(\sqrt[n]{S/w+1} - 1) \\ &= b \cdot ln(S/w+1) \end{aligned} \quad (3)$$

For any video with consumption rate given by $b \cdot f(t)$, at any time t to $t + dt$, the broadcast scheme must at least transmit $b \cdot f(t)dt$ length of data. This amount of data should transmit fully during its good downloading time $w + t$. The bandwidth required to transmit this amount of data is $\frac{b \cdot f(t)dt}{w+t}$. The total bandwidth requirement for a video of length S is:

$$B_0(VBR) = \int_0^S \frac{b \cdot f(t)dt}{w+t} \quad (4)$$

■

Note that in the CBR case, $f(t) = 1$ and equation 4 reduces to the value in equation 3. By varying video length t, an asymptote

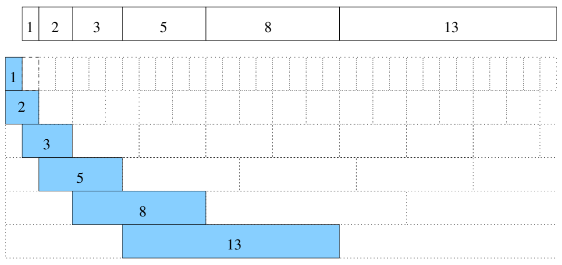


Fig. 9. An Illustration of Design Scheme to Restrict Client I/O Bandwidth

$B(t) = b \cdot ln(t/w + 1)$ can be obtained as shown in Figure 8. Any normalized broadcasting streams are confined by this asymptote.

VI. Client I/O Bandwidth and Storage Requirement Analysis

While server bandwidth requirement and client access latency are the key issues in current VOD systems, we should also take into account two other performance measures: client I/O bandwidth and buffer space requirements. Optimally-structured broadcasting protocols require client download at the same bandwidth as the server for the video. Although the optimal division of the video gives out the lowest server bandwidth, it could still cause too high demand at the client side.

Since both pyramid broadcasting and skyscraper broadcasting download no more than two logical channels, their I/O bandwidth requirement could be less than that of optimaly-structured protocols. A convenient change to the optimaly-structured protocol can be made to meet the maximum client I/O bandwidth requirement. As illustrated in Figure 9, assume client can download at most 2 streams at rate b, the first two segments are constructed as normal, starting from the third segment, segment i's left edge is aligned with segment $i-2$'s right edge. This guarantees that client needs to download at most at rate $2b$. The segment progression is a fibonacci series. Its server bandwidth requirement is better than skyscaper protocol and its variants. Note also that starting from the third broadcast stream, the unused portion of the cycle need not transmit any data, this is different from any other existing broadcasting protocols. As a result, the *average* server bandwidth for this scheme is only $4.78b$, less than $6b$ since the unused portion in each good segment downloading time can be used to transmit any other data.

As for the buffer space requirement, we first give an estimate of the optimally-structured scheme. The worst case buffer space requirement for the optimal case of CBR video when $n \to \infty$ can be calculated as follows. The buffer space requirement at any moment t' is given by the shaded rectangle area shown in Figure 8. The blank area below the asymptote and above the shaded area is the part that has been played. Client first downloads the video at bandwidth B_0 and starts accumulating video segments. After time w, it starts to consume the video at rate b. The downloading speed slows down and at the time when it reduces to the consumption rate b, the buffered data reaches its highest amount. After that the downloading speed will be less than b and the buffer space requirement is reduced. So the worst case buffer space requirement can be calculated when the shaded rectangle has the height of b, and its length corresponds to the t value given in $B(t) = b \cdot ln(t/w + 1) = B_0 - b = b \cdot (ln(S/w+1)-1)$. So the buffer space requirement is $b \cdot \frac{S+w}{e}$, almost $1/e$, or 37% of the video, if w is relatively much smaller than S. With fixed segment number n and optimal partition scheme, the buffered data are given by:

$$(w + \sum_{j=1}^{l} S_j) \cdot (n-l)b^*$$

where l is the last segment that satisfies $(n-l)b^* > b$, or $l = \lfloor n - \frac{b}{b^*} \rfloor$. Note that when $b^* > b$, $l = n-1$ and the worst case buffer space is the size of the last segment, *i.e.*, $b^*(1+b^*)^{n-1}$, if S is normalized with w, according to equation 1, it can be presented as:

$$((1 - \frac{1}{\sqrt[n]{S+1}}) \cdot \frac{S+1}{S}) \cdot S$$

Since $\frac{S+1}{S}$ is almost 1 when the video length is relatively larger than the access latency, the worst case buffer space requirement is $(1 - \frac{1}{\sqrt[n]{S+1}})$ of the total video. With 7 segments and video length of 127 and access latency of 1, it corresponds to 50% of the video. We can expect less buffer space requirement when we partition the video into more segments. With more segments, client will be able to download at lower pace. So the client storage requirement is also related to client I/O bandwidth. Restriction on client download speed will require less client storage. The scheme illustrated in Figure 9 requires only 31% of the video.

The tradeoff among the different design goals can be observed from the above discussion. A well-designed broadcasting protocol needs to consider different design goals and tailor its structure to meet all the requirements. The proposed analytical approach is a good guide to analyze broadcasting protocols and direct the design of these protocols to achieve different performance goals in different categories. To reduce the server bandwidth of optimally-structured schemes will at the same time reduce the client I/O bandwidth as well as the storage requirement. But reducing server bandwidth could end up with too many segments and it also has a lower bound. If client I/O bandwidth and storage requirement are the bottleneck, the optimally-structured scheme can be modified to satisfy these requirement while still remains the best in terms of server bandwidth requirement.

One last note here is the possibility of reducing the segment number and broadcasting streams to decrease encoding and decoding complexity. So far most of the protocols have one segment per stream. The exception is the Pagoda family schemes. The artful design of these schemes packs different segments together in one stream, thus reducing the number of broadcasting and downloading streams. Note that streams in broadcasting protocols are logical streams. There is no requirement for streams in optimally-structured schemes transmit in different physical channels. Different streams can be packed together and transmitted in one physical channel. It adds no more complication to the client to reassemble the segments with the structure in optimally-structured scheme than Pagoda family schemes. As for the segment number, as we proved before, to reduce the segment number will end up with larger server bandwidth requirement. Again here is the trade-off between implementation complexity and performance results.

VII. True VOD and Interactive Function Support

All the existing broadcasting protocols do not support true VOD, or zero client waiting time. One possible solution is proposed in [14]. It requires the client set-top box predownload the first segments of the most popular videos. Client can start watching right away if he tunes into the popular videos. This scheme does not always work since client may not want to watch the predetermined popular videos. Here we propose a slight change to the broadcasting protocols to provide true VOD. Rather than broadcasting the first segment in the network, the first segment can be transmitted on demand. Its length should be equal to the good downloading time of the second segment that is broadcast in the network. Since the first segment could be very small, it won't require demanding server bandwidth. The remaining segments are broadcast as normal.

To support interactive VOD in broadcasting schemes is a tough issue. None of the known protocols support interactive functions. While pause and rewind can be supported by introducing more buffer space, fast forwarding is the most difficult to implement in broadcasting protocols. Fast forward requires the video data transmitted in shorter time than normal, but the most efficient broadcasting protocol in terms of server bandwidth tries to transmit data *just in time*. Optimally-structured schemes transmit video segments throughout its good downloading time. A x time units forward action would result in a later segment i downloading less $x \cdot b_i$ of data by the time when they are needed to be played out, where b_i is the transmission rate for the ith segment.

We propose two possible solutions here to support interactive VOD. The first approach is based on the optimally-structured scheme. Due to the predetermined nature of broadcasting schemes, we can only support limited fast forward service, say up to x time units of fast forwarding. Rather than transmitting each data segment throughout its good segment downloading time, we can change the broadcasting period for each segment to be x time units less than its good downloading time. Note that the corresponding segment stream will require higher transmission bandwidth. Another approach is to transmit unreceived portion of the segment on demand. The client should still receive each broadcasting stream as normal and the server will transmit the missing $x \cdot b_i$ data for each of the later segment i.

VIII. Broadcast vs. Multicast

Broadcasting schemes have proved themselves as efficient for most of the popular videos. They require predetermined server bandwidth to achieve certain access latency and they are not subject to data loss. The downsides are they are not flexible in providing true or interactive VOD service. Many multicasting schemes are proposed to provide immediate service to users while still keep server bandwidth requirement low. But all these schemes can not perform well when client arrival rate is high. The lower bound on required server bandwidth for multicast schemes is given in [20] as: $ln(S\lambda+1)$. The broadcasting lower bound given in Equation 3 is exactly a special case where we consider the arrival rate is 1 every w time units. In other words, multicast schemes with regular client arrival rate of $1/w$ degenerate to broadcast schemes. In real world, client arrival could be bursty and all multicast schemes can not scale well when λ is too high. We envision that a good way to provide VOD service is to combine both the proactive and reactive approaches together. The essence of the existing multicast schemes is to share the later part of the video as much as possible. To transmit first segment of the video immediately and use later segments from broadcasting streams is a promising way to provide guaranteed service for popular videos. This is essentially to *merge users to the shared stream just after the first segment*. Note however that there is a lower bound server bandwidth requirement for broadcasting schemes. Broadcasting is not efficient when the video is not required frequently. VOD system needs to decide certain threshold from which we can divide videos according to their popularity and choose to transmit them using broadcasting or multicasting schemes.

IX. Concluding Remarks

Network bandwidth has been identified as a serious bottleneck in today's media servers. Many researchers have shown that broadcasting is a good remedy for this problem. From the pioneering work of pyramid broadcasting protocol and its variants to the harmonic family protocols, we see the trend of lower broadcasting speed for the video segments. This lower transmission speed at the video servers results in a lower client I/O bandwidth requirement and less buffer space requirement at the client end. Based on our generalized analysis approach by means of the proposed temporal-bandwidth map, we observe that a common thread uniting all these schemes is to transmit the video segments *just in time*. To achieve the lowest server bandwidth requirement, it is crucial that video segments be transmitted during their good downloading time.

We proved that there exists a lower server bandwidth requirement given client access latency requirement and video length. The lower bound is reached when there are infinity number of segments broadcasting in the network. Given limited segment number n, the optimal partition of the video is to let the segment progression with the golden factor of $\sqrt[n]{S/w+1}$ and each broadcasting stream have equal bandwidth. The optimal partition of VBR videos can be calculated using a dynamic programming algorithm. VBR videos are naturally transmitted with CBR streams and there is no need to apply smoothing techniques to reduce data loss.

The proposed generalized analysis approach provides an insight look at the broadcasting protocols. The four important performance measures: broadcasting bandwidth, access latency, client I/O bandwidth requirement and buffer space requirement can be observed through the temporal-bandwidth map. To design a broadcasting scheme to meet different performance requirement, we can start with the optimally-structured schemes and then tailor them to meet the other requirements. The resulting scheme will require the least server bandwidth while still meet the other requirements. We provided examples on how to restrict client I/O bandwidth and provide interactive function based on this approach.

The essence of all the protocols in VOD system, no matter they are broadcasting or multicasting schemes, is to let users share as much data as possible. The reactive multicast approach degenerates to broadcast structure for the arrival rate of $1/w$. We envision a better way to provide VOD service is to combine both reactive and proactive approaches together to provide true interactive and still guaranteed VOD service.

REFERENCES

[1] S. Viswanathan and T. Imielinski. Pyramid Broadcasting for video on demand service. In *IEEE Multimedia Computing and Networking Conference*, Volume 2417, pp 66-77, San Jose, California, 1995.

[2] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. In *Proc. of the IEEE Int'l conf. on Multimedia Computing and Systems '96*, Hiroshima, Japan, June 1996.

[3] K. A. Hua and S. Sheu. Skyscraper Broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. In *SIGCOMM 97*, pp 89-100, Cannes, France, Sept. 1997. ACM.

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

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

[6] J.-F. Paris, S. W. Carter, and D. D. E. Long. A low bandwidth broadcasting protocol for video on demand. In *Proc. of IEEE Int'l Conference on Computer Communications and Networks (IC3N'98)*.

[7] L. Juhn and L. Tseng. Fast data broadcasting and receiving scheme for popular video service. In *IEEE Transactions on Broadcasting*, 44(1):100-105, Mar 1998.

[8] J.-F. Paris, S. W. Carter, and D. D. E. Long. A hybrid broadcasting protocol for video on demand. In *Proc. 1999 Multimedia Computing and Networking Conference (MMCN'99)*, San Jose, CA, Jan 1999, pp 317-326.

[9] J.-F. Paris. A simple low-bandwidth broadcasting protocol for video-on-demand. In *Proc. 8th Int'l Conference on Computer Communications and Networks (IC3N'99)*, Boston-Natick, MA, Oct 1999, pp 118-123.

[10] A. Hu, I. Nikolaidis, P. van Beek. On the design of efficient video-on-demand broadcast schemes. In *Proc. of 7th Int'l Symp. On Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '99)*, pp 262-269.

[11] J.-F. Paris. A broadcasting protocol for compressed video. In *Proc. Euromedia '99 Conference*, Munich, Germany, Apr 1999, pp 78-84.

[12] D. Saparilla, K. Ross, M. Reisslein. Periodic broadcasting with VBR-encoded video. In *Proc. of IEEE Infocom '99*, pp 464-471.

[13] F. Li, I. Nikolaidis. Trace-adaptive fragmentation for periodic broadcast of VBR video. In *Proc. of 9th Int'l Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '99)*, June 1999.

[14] J.-F. Paris, S. W. Carter, and P. E. Mantey. Zero-delay broadcasting protocols for video-on-demand. In *proc. 1999 ACM Multimedia Conference*, Orlando, FL, Nov 1999, pp 189-197.

[15] K. c. Almeroth and M. H. Ammar. The use of multicast delivery to provide a scalable and interactive video-on-demand service. *IEEE Journal on Selected Areas in Communications*, 14(5):1110-22, Aug 1996.

[16] C. C. Aggarwal, J. L. Wolf and P. S. Yu. On optimal piggyback merging policies for video-on-demand systems. In *Proc. 1996 ACM SIGMETRICS Conf. On Measurement and Modeling of Computer Systems*, Philadelphia, PA, May 1996, pp. 200-209.

[17] A. Dan, D. Sitaram and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proc. 6th Int'l. Multimedia Conf.(ACM Multimedia '94)*, San Francisco, CA, Oct 1994, pp. 15-23.

[18] K. A. Hua, Y. Cai and S. Sheu. Patching: a multicast technique for true video-on-demand services. In *Proc. 6th ACM Int'l. Multimedia Conf. (ACM Multimedia '98)*, Bristol, U.K., Sept 1998, pp 191-200.

[19] S. W. Carter and D. D. E. Long. Improving video-on-demand server efficiency through stream tapping. In *Proc. 6th Int'l. Conf. On Computer Communications and Networks (ICCCN'97)*, Las Vegas, NV, Sept 1997, pp. 200-207.

[20] D. Eager, M. Vernon, J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. In *Proc. 5th Int'l. Workshop on Multimedia Information Systems (MIS '99)*, Indian Wells, CA, Oct 1999.