

# Survey on VOD Broadcasting Schemes

Tiko Kameda and Richard Sun  
 School of Computing Science  
 Simon Fraser University

**Warning: This survey article is still undergoing occasional revisions. Please make sure you read the latest version.** Original version = July 20, 2002; last update = April 2, 2005.

**Abstract**—This paper discusses the principles underlying efficient video broadcasting and surveys a large number of recent papers written on the subject of video-on-demand (VOD), using consistent terminology and notation.

**Index Terms**—Video-on-demand, VOD, CBR, VBR, video broadcasting.

## Table of Contents

Section	Title
I	Introduction
II	Preliminaries
II-A	Basic principles and strategies
II-B	Lower bound on server bandwidth for FD schemes
II-C	Lower bound on server bandwidth for FSP schemes
II-D	Simple broadcasting schemes
II-D.1	Staggered broadcasting
II-D.2	Fast Broadcasting (FB)
III	Most Efficient FD Schemes
III-A	Greedy Equal Bandwidth Broadcasting (GEBB)
III-B	GEBB with prefix caching
III-C	Generalized Fibonacci Broadcasting (GFB)
IV	Classification
IV-A	Slotted schemes
IV-B	Harmonic-based broadcasting
IV-C	Pagoda Broadcasting Schemes (PaB)
V	Schemes with user bandwidth limit
V-A	Pyramid Broadcasting (PyB)
V-B	Skyscraper Broadcasting (SkB)
V-C	Fixed-delay PaB with user bandwidth limit (FDPB)
	References

## I. INTRODUCTION

CLIENT-server paradigm has been used successfully to provide various services over networks. However, it has the scalability problem, if the number of clients increases beyond a certain limit. To overcome this difficulty in video-on-demand, researchers have investigated the idea of broadcasting popular videos for the last several years. It was reported in [11] that the frequency of requests for videos follow Zipf's distribution with the skew factor of 0.271. This implies that people watch first 20 most popular videos most of the time.

The authors welcome any feedback/suggestions on this draft.

T. Kameda and Y. Sun are with the School of Computing Science, Simon Fraser University, Burnaby, B.C. V5A 1S6 Canada, Email: {tiko,sunyi}@cs.sfu.ca;

This observation is often used to justify the broadcasting of popular videos.

Since a novel idea of broadcasting videos was published by Viswanathan and Imielinski in 1995,<sup>1</sup> there has been tremendous interest in the subject of broadcast-based VOD. Researchers have proposed many different broadcasting schemes to date, most of which deal with **constant-bit-rate (CBR)** videos. Since commercial videos are actually encoded using a more efficient **variable-bit-rate (VBR)** encoding, more recent schemes are specifically tailored for VBR videos. Undoubtedly, more new schemes will be invented in the future, but it is probably safe to say that we now understand the basic principles of broadcasting, especially of CBR videos.

The purpose of Section II is to introduce the reader into the basic principles that underlie the various schemes. We also introduce the uniform notation that we will be using throughout this paper. We describe a few simple schemes by which it is easy to demonstrate how these principles are actually used, although they are not the most efficient.

In Section III, we present the most efficient broadcasting schemes for CVR videos. In the first one, Greedy Equal Bandwidth Broadcasting, the server constantly broadcasts segments of each video on  $n$  channels, and a user must be capable of receiving from all of the  $n$  channels, at least initially. It is the most efficient under the constraint that each segment must be preloaded entirely before display. The second one, Generalized Fibonacci Broadcasting, assumes that a user's STB has a limit in its receiving bandwidth.

Section IV classifies all the currently known schemes based on several criteria. We also discuss slotted schemes such as Harmonic-based schemes and Pagoda schemes.

In Section V, we then describe several schemes with user bandwidth limit.

## II. PRELIMINARIES

A video is displayed (played out) on a TV screen or a computer monitor. It consists of a large number of frames, each of which carries encoded data that is needed to display a screenful of image. The **frame rate**  $F$  (frames/second) is a constant for a given system. To avoid flickering,  $F$  typically ranges from 24 to 30 (frames/second). If  $F$  is lower than 20, the human eye will perceive annoying flickers. Let  $N$  denote the number of frames in a video. If  $F = 25$ , for example, then for a two-hour video we have  $N = 25 \times 60 \times 60 \times 2 = 180,000$ . For  $i = 1, 2, \dots, N$ , let  $f_i$  stand for the number of bits used to

<sup>1</sup>There is little-known prior work by Hollmann and Holzschner [21], which predates this work by four years. We are thankful to Jan Korst of Philips Research Laboratory for bringing this information to our attention.

encode the  $i^{\text{th}}$  frame. Clearly,  $f_i$  is a constant independent of  $i$  for a CBR video. The display rate  $r$  (bits/second) is the rate of display. Let  $D$  (seconds) denote the **duration** of a video, namely the time needed to display the entire video. For any video (CBR or VBR), we have the following relationship,:

$$D = N/F. \quad (1)$$

All broadcasting schemes (except the staggered scheme) divide a video into  $n$  **segments**, where  $n \geq 1$ . In this paper, a segment consists of consecutive frames. In some schemes, a “segment” is divided into subsegments and they are not broadcast consecutively. In such a case, in our definition each subsegment is called a segment. For  $j = 1, 2, \dots, n$ , we denote the  $j^{\text{th}}$  segment by  $S_j$ , and consider it as a sequence of frames.  $S_j$  is broadcast on a separate (logical) **channel**,  $C_j$ , with bandwidth  $B_j$ . The number of frames in  $S_j$  is denoted by  $|S_j|$ , while its duration is denoted by  $D_j$  (seconds). We thus have  $D_j = |S_j|/F$ .

We mentioned a channel in the above paragraph. A logical channel can be realized in many different ways. Fig. 1 illustrates 32 channels realized by **time division multiplexing (TDM)**. In this example, a 2.048Mbps link is subdivided into 32 logical channels. (This link may itself be a logical link, which is a subchannel of a super channel which multiplexes multiple 2.048Mbps channels.) The bit stream on this link is structured into repetitive 256-bit TDM frames. The first group of 8 bits of the TDM frame belongs to channel 0, the second group belongs to channel 1, etc., up to channel 31. Each channel thus has a rate of 2.048Mbps/32=64Kbps. Some or all of these 32 channels may be used to transmit data for a single video. It is also possible that a video is broadcast using more than 32 such channels.

In general, a TDM slot may consist of more than 8 bits or even just one bit. There are also other ways of realizing logical channels, such as **frequency division multiplexing (FDM)**. The interested reader is referred to a standard text book on digital communications [49].

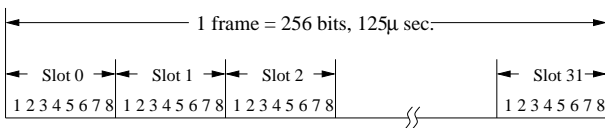


Fig. 1. ISDN primary access frame format.

In general, when the user wishes to view a video, she gives her command to her **set-top box (STB)** for short, upon which the STB initiates some actions. These actions involve the downloading of the segments of the video. After a delay of  $w$  (seconds), the STB is ready to start displaying the video.<sup>2</sup> See Fig. 2. As stated earlier, this initial delay may be fixed (i.e., independent of the particular moment the user pushes a button on the STB) or may depend on the “tune-in” time.

<sup>2</sup>Some schemes assume that a **prefix** (initial segment denoted by  $S_0$ ) is already cached in the STB ahead of time. In such a scheme there is no initial waiting time, or **start-up latency**, i.e.,  $w = 0$ .

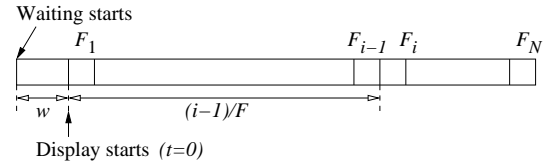


Fig. 2. Display starts after initial waiting time of  $w$  (seconds). Frame  $F_i$  must be downloaded within  $w + (i - 1)/F$  (seconds) to ensure continuity.

### A. Basic principles and strategies

Unless otherwise specified, we assume CBR-encoded videos, and let  $r$  (bits/sec) denote the **display** (or **consumption**) rate. Suppose a video consist of  $n$  ( $\geq 1$ ) segments,  $S_1, S_2, \dots, S_n$ .

Some schemes rely on the fact that time is slotted into **slots** of a fixed size  $d$ , and the time slots among all channels are aligned, i.e., start and end at the same time. We call such schemes (synchronized) **slotted** schemes. A slotted scheme has the property that a segment is broadcast using an integral number of slots.

### Downloading/Display Policies

- (1) **Fixed Delay (FD)**: Start downloading immediately, but wait a fixed amount of time in the beginning until the entire  $S_1$  is buffered [39]. (Since  $S_1$  is repeatedly broadcast on channel  $C_1$ , and the time the user “tunes in” is not synchronized with this period, buffering typically starts somewhere in the middle of  $S_1$ , continuing to the end of  $S_1$  and then back to the first part which was missed.) This policy is sometimes referred to as **greedy** [23].
- (2) **Fixed Start Points (FSP)**: This policy used with slotted channels. Wait until time  $t_1$ , when the next slot (or the next group of slots) starts in a channel, taking no action until  $t_1$ . Start downloading at  $t_1$ ,<sup>3</sup> and start viewing at time  $t_2$ , which is the beginning of a slot, where  $t_1 \leq t_2$ .

From now on, a scheme adopting the FD (resp. FSP) policy is called an **FD** (resp. **FSP**) **scheme**. Fig. 3 shows the **download-display diagram (DDD)** of an FD scheme. The horizontal axis represents time and the vertical axis represents the amount of data in the segments of a video. In this example, the video is divided into three segments,  $S_1, S_2$ , and  $S_3$ . The diagonal line labeled “Display line” indicates which bit of which segment is displayed as time progresses from  $t = t_1$ . The user “tunes in” at  $t = t_0$  and immediately starts downloading from all the three channels, but waits for a fixed amount of time  $w$  (called the (initial) **waiting time** hereafter) until  $t_1$  before s/he starts to display the video. For  $i = 1, 2, 3$ ,  $S_i$  is completely downloaded, before its display can start, which takes time  $D_i$ . In order for every bit of  $S_i$  to be downloaded before its display, if a horizontal line is drawn across the diagram at any vertical position, then the line must intersect a “download” line within the shaded area. In this example, the user tunes in at exactly the time when the

<sup>3</sup>Since this policy requires that it must work even when the tune-in time coincides with the beginning of  $S_1$ , there is no point downloading other segments during the wait time.

first bit of  $S_2$  is about to appear on channel 2, while about a half (quarter) of  $S_1$  ( $S_3$ ) has already appeared in channel 1 (3). In general, however, the user may tune in at an arbitrary time, and therefore, the display line can be at an arbitrary position. Consequently, the bit of  $S_i$  that channel  $i$  happens to be transmitting at that time can belong to any frame of  $S_i$ .

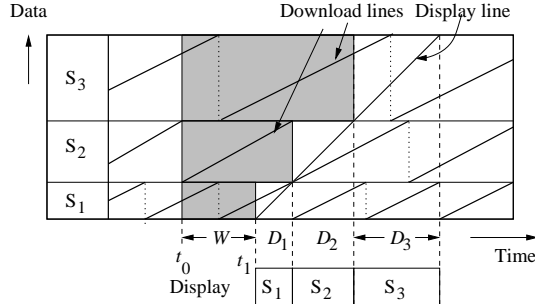


Fig. 3. Download-display diagram for a segment preloading scheme with fixed-delay.

The FSP policy implies that the bandwidth of the first channel  $C_1$  must be at least as large as the display rate  $r$ . Fig. 4 shows the download-display diagram for an FSP scheme. The user initially waits until the next slot starts, doing nothing until that time. As soon as the next slot starts, downloading starts from all the channels, while  $S_1$  is displayed as its successive frames come in. In the example, we have  $t_1 = t_2$  (see the definition of the FSP scheme given above), but display could start later than downloading. In any case the display line can start only at discrete time points that delineate slots. There are few schemes that sets  $t_1 < t_2$ , which we will discuss later. Note that the shaded area extends all the way to the display line, which implies that the user STB has a longer time (hence less bandwidth) to download segments. However, the FSP schemes initially waste time (hence bandwidth) waiting for time  $t_1$ . Which policy, FD or FSP, is better depends on the performance metric adopted. As we shall see, if the maximum waiting time for given bandwidth is used as the performance metric, then both policies have the same lower bound. If the average waiting time for given bandwidth is used, however, there are FSP schemes that outperform all the FD schemes.

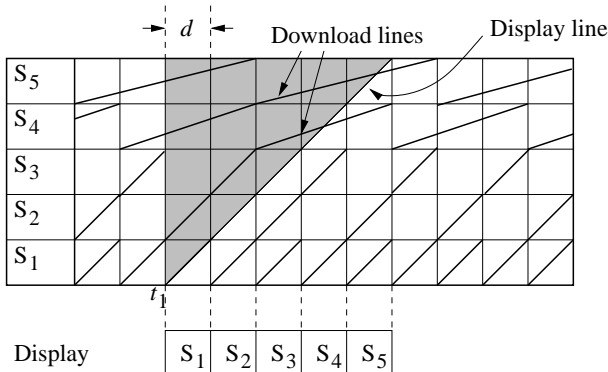


Fig. 4. Download-display diagram for an FSP scheme.

Let us define the *normalized* waiting time by  $w^* = w/D$ ,

where  $D$  is the video duration, and the *normalized* bandwidth  $B^* = B/r$ . Table I summarizes the notation that we have introduced so far.

TABLE I  
NOTATION SUMMARY

$D$	Total video duration (seconds)
$r$	Display or consumption rate (bits/sec)
$F$	Frame rate (frames/sec)
$N$	Number of frames
$F_i$	$i^{\text{th}}$ frame for $i = 1, 2, \dots, N$
$n$	Number of segments (channels)
$S_j$	$j^{\text{th}}$ segment for $j = 1, 2, \dots, n$
$ S_j $	Number of frames in $S_j$
$D_j$	Duration of $j^{\text{th}}$ segment (sec)
$d$	equal to $D_1$ ; size of time slot
$C_j$	$j^{\text{th}}$ channel for $j = 1, 2, \dots, n$
$B_j$	Bandwidth of $C_j$ (bits/sec)
$B$	Server bandwidth (bits/sec)
$B^*$	Normalized server bandwidth ( $= B/r$ )
$w$	Initial wait time (seconds)
$w^*$	Normalized initial wait time ( $= w/D$ )

#### B. Lower bound on server bandwidth for FD schemes

Let us assume for now that a frame is a unit of data transfer as well as display. Then it can be seen from Fig. 2 that frame  $F_i$  must be downloaded by time  $w + (i-1)/F$  (seconds) at the latest in order to avoid discontinuity in display. In other words, **frame preloading** is a necessity. The minimum bandwidth required to download just frame  $F_i$  is thus  $f_i/(w + (i-1)/F)$ , and the minimum total bandwidth can be computed as

$$B^* = \sum_{i=1}^N f_i/(w + (i-1)/F). \quad (2)$$

This minimum bandwidth is achievable in theory, if we employ  $N$  tiny channels. However, the overhead of managing 180,000 tiny channels would be impractical and thus we need to reduce the set of channels to a more manageable number. This naturally leads to the idea of segmentation. In other words, we divide the entire video into  $n$  segments, each consisting of a certain number of frames. As we saw above, a later frame requires less bandwidth because the denominator  $w + (i-1)/F$  gets larger for larger  $i$ . A similar reasoning implies that a later segment would require less bandwidth if each segment had the same size. Another way of looking at it is that, if we use a channel with equal bandwidth for downloading each segment, then a later segment can be larger, because it has a longer time to utilize its (dedicated) channel. There are two classes of broadcasting based on the above two observations. The schemes in the first class use equal-bandwidth channels to download segments of different sizes (i.e., containing different number of frames), while those in the second class use channels of different bandwidths to download equal-sized segments.

The summation in Eq. (2) can be approximated by integration as follows. Suppose the amount of data consumed (i.e., displayed) during a small time interval  $[t, t + dt]$  is  $f(t)dt$ , where  $t$  is measured from the display start time (downloading starts at time  $t = -w$ ). We need to download this amount of

data over the interval  $[-w, t]$ . Stated another way, this amount ( $f(t)dt$ ) of data must be available in any time interval of length  $w + t$ . This requires bandwidth at least  $f(t)dt/(w + t)$ . Integrating this over the entire display duration  $[0, D]$ , we obtain [22]

$$\int_0^D f(t)dt/(w + t). \quad (3)$$

In the CBR case, we have  $f(t) = r$ , and the above integration can be carried out as follows [39]:

$$B \geq \int_0^D rdt/(w + t) = r \ln(1 + D/w). \quad (4)$$

The following theorem follows from the above discussions:

*Theorem 1:* [22] The normalized server bandwidth required by any FD scheme is bounded by

$$B^* \geq \ln(1 + 1/w^*). \quad (5)$$

Eq. (5) can be rewritten to express the lower bound on  $w^*$  as a function of  $B^*$  as follows:

*Corollary 1:* The normalized waiting time required by any FD scheme is bounded by

$$w^* \geq 1/(e^{B^*} - 1). \quad (6)$$

### C. Lower bound on server bandwidth for FSP schemes

Since the FSP schemes use fixed-size segments, we shall refer to them as **pages**. H.D.L. Hollmann and C.D. Holzschere [21] first proposed the FSP policy of waiting for a slot boundary and then buffering  $m - 1$  pages before starting to displaying page 1, either from the buffer or directly from a channel. **Harmonic Broadcasting (HB)** [28] is a special case where  $m = 1$ . Clearly, in order to download and display page 1, starting at the first slot boundary, it is necessary and sufficient that page 1 appear in every slot. It is shown in [28] that for continuous display it is necessary and sufficient that page  $i$  appear at least once in any consecutive  $i$  slots. This implies that page  $i$  requires at least  $r/i$ . Let  $H_n$  denote the  $n^{\text{th}}$  **harmonic number** defined by

$$H_n = \sum_{i=1}^n 1/i.$$

Then  $n$  pages together require bandwidth at least  $rH_n$ .

In the general case, page  $i$  must appear at least once in any consecutive  $i + m - 1$  slots [39]. Since the period is shifted by  $m - 1$ , Bar-Noy et al. call it the *shifting technique* [6]. The bandwidth required in this case is at least

$$\sum_{i=1}^n \frac{r}{i + m - 1} = r(H_{n+m-1} - H_{m-1}).$$

*Lemma 1:* [20]

$$H_{n+m-1} - H_{m-1} = \ln(1 + n/m) + \frac{1}{2m(1 + m/n)} + O(m^{-2}). \quad (7)$$

We shall now establish an upper bound on the number of pages that can be sent over  $c$  channels when an FSP scheme is used. Let  $h_m(c)$  denote the maximum integer  $n$  satisfying

$$H_{n+m-1} - H_{m-1} \leq c$$

for given  $m$ . We use a simpler notation  $h(c)$  for  $h_1(c)$ .

*Theorem 2:* Let the given bandwidth be  $cr$  and the page size be  $p$  (bits), where  $r$  is the display rate and  $c$  is a real number ( $\geq 1$ ), and let  $X = c - H_{h_m(c)+m-1} + H_{m-1}$ . The following formula gives an upper bound on the amount of data that can be scheduled in any FSP schedule:

$$\frac{h_m(c) + (m - 1)X}{1 - X} p. \quad (8)$$

*Proof:* To be provided later. ■

*Corollary 2:* The normalized maximum and average waiting times can be bounded from below by

$$\frac{m(1 - X)}{h_m(c) + (m - 1)X}$$

and

$$\frac{(m - 1/2)(1 - X)}{h_m(c) + (m - 1)X}, \quad (9)$$

respectively.

*Proof:* The maximum initial waiting time is  $mp/r$  and the video duration is  $(h_m(c) + (m - 1)X)p/r(1 - X)$ . By dividing the former by the latter, we get the first formula in the corollary. Since the average waiting time is  $(m - 1/2)p/r$ , we get the second formula in the corollary. ■

Let us assume that  $X \approx 0$ . The error in this approximation is  $O(1/h(c))$ , which gets smaller for larger  $c$ . Eq. (9) is minimized when  $m = 1$  (i.e., for HB), and it becomes

$$\frac{1}{2h(c)}. \quad (10)$$

### D. Simple broadcasting schemes

The recent flurry of research activities on VOD broadcasting was precipitated by the publication of Pyramid Broadcasting (PyB), which was based on an entirely new idea. In the rest of this section, we put it in perspective, and also explain the basic idea underlying PyB.

1) *Staggered broadcasting:* Before the novel idea of broadcasting used in PyB was known, the most natural method was to stagger many streams of the same video [11]. For example, a 120-minute video could be broadcast on 120 channels, each channel repeating the same video delayed by one minute from the previous channel. This way, any user needs to wait at most one minute, and the STB needs no buffer. However, this scheme is obviously impractical, since it requires bandwidth that is 120 times the display rate ( $120r$ ) just to broadcast one video. It does have some advantages though, such as that “fastforward” and “rewind” operations are easy to implement.

2) *Fast broadcasting:* The basic idea in PyB can be best illustrated by **Fast Broadcasting (FB)** [29], which was invented later. This scheme, like PyB, divides a given video into a sequence of  $n$  segments of geometrically increasing sizes. (But each segment can be considered a sequence of fixed-size pages.) Let  $d = D/(2^n - 1)$ . Then the duration of segment  $S_i$ ,  $D_i$ , is given by

$$D_i = 2^{i-1}d. \quad (11)$$

Therefore the sum of the durations of the first  $i$  segments is

$$\sum_{j=1}^i 2^{j-1}d = (2^i - 1)d. \quad (12)$$

Suppose each segment is broadcast repeatedly on a dedicated channel of bandwidth equal to the display rate  $r$  (bits/sec), starting at some point in time. Consider a user who starts looking for the beginning of  $S_1$  on channel  $C_1$ . Within at most  $d$  (seconds) she will find it, starts viewing it, while downloading it at the same time. Let  $t_1$  denote the time when a new period starts on channel  $C_1$ . Therefore, segment  $S_1$  starts at  $t_1$  and ends at  $t_1 + d$ , and repeats this cycle indefinitely. There are two possible cases regarding the relative timing of segments  $S_1$  in channel  $C_1$  and  $S_2$  in channel  $C_2$ . In channel  $C_2$ , segment  $S_2$  starts (a) at  $t_1$  and ends at  $t_1 + 2d$ , or (b) starts  $t_1 - d$  and ends at  $t_1 + d$ . In case (b), the user can switch from  $S_1$  to  $S_2$  at time  $t_1 + d$ , but in case (a), the user should start buffering  $S_2$  at  $t_1$ , while viewing  $S_1$ . When  $S_1$  ends at time  $t_1 + d$ , the first  $d$  seconds of  $S_2$  will have been buffered in the STB and can be retrieved, while the rest of  $S_2$  is being buffered.

According to equations (11) and (12), the sum of the first  $i-1$  segments is shorter than  $S_i$  by  $d$ . In the best case scenario,  $S_i$  starts in channel  $C_i$  at exactly the time when the display of the first  $i-1$  has just ended. In this case it is not necessary to buffer  $S_i$  at all. In all other cases, various amount of  $S_i$  need to be buffered before the display of the first  $i-1$  segments ends. In any case, however, for any  $i = 1, 2, \dots, n$ , after viewing the first  $i-1$  segments, the user can continue to view  $S_i$ , either directly from channel  $C_i$  or from the local buffer.

Now how good is FB? Let us express the normalized waiting time  $w^*$  in terms of the normalized server bandwidth  $B^*$  for this scheme and compare it with some lower bounds. To compare different schemes fairly, it is important to choose fair metrics for comparison.

As we saw above, FB is an FSP scheme. Note that we want to compare the **average** waiting times, with the lower bound given by Eq. (5), but under the assumptions that led to Eq. (5), there is no difference between the average and maximum waiting times. Since the average-case waiting time is  $d/2$ , we set  $w^* = d/2D$ . Since  $D = (2^n - 1)d$  from Eq. (12), we have  $w^* = 1/2(2^n - 1)$ , hence  $n = \ln(1 + 1/2w^*)/\ln 2$ . The bandwidth required by FB is given by  $B = nr$  or  $B^* = n$ . We thus obtain the normalized average wait time for FB:

$$B_{\text{FB}}^* = \ln\left(1 + \frac{1}{2w^*}\right)/\ln 2. \quad (13)$$

We now compare its performance with a lower bound for any FSP scheme, which is provided by Harmonic Broadcasting (HB) mentioned earlier. Eq. (13) is plotted in Fig. 5 together with the lower bound of Eq. (10) for HB.

Note that HB is not actually realizable but provides a lower bound. The performance of some scheme (e.g., **Quasi Harmonic Broadcasting (QHB)**) approaches this bound asymptotically. Since these schemes are based on the FSP policy, their performance is better than that for the schemes with segment preloading, if the average waiting time is used as the performance metric. See Fig. 6.

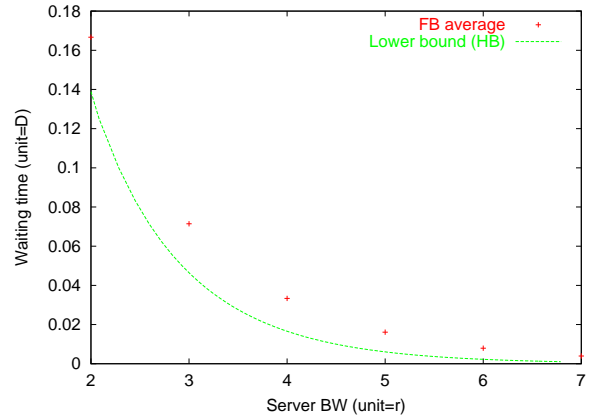


Fig. 5. Performance of FB vs. the lower bound (HB).

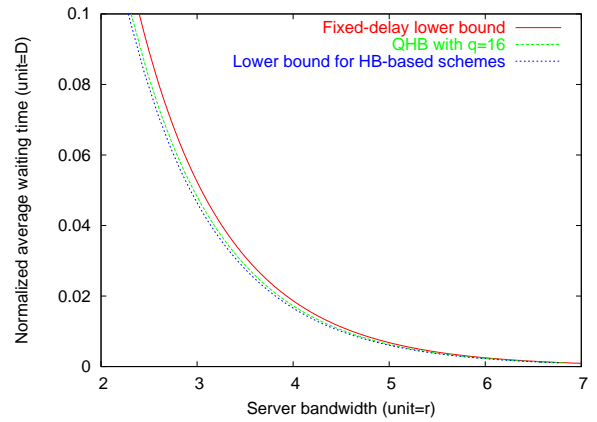


Fig. 6. Bounds for FD and FSP schemes, as well as performance of QHB ( $q = 16$ ).

### III. MOST EFFICIENT FIXED-DELAY SCHEMES

In [23] Hu et al. published the most bandwidth-efficient FD scheme, called **Greedy Equal Bandwidth Broadcasting (GEBB)**. It is the most efficient in the sense that, given any *maximum* waiting time ( $w$ ) and the number of channels ( $n$ ), GEBB uses the least bandwidth ( $B$ ). Another way of saying this is that, given bandwidth and the number of channels, it achieves the shortest *maximum* waiting time among all broadcasting schemes. In this section, we first describe how GEBB works, and then we adapt it to work with prefix caching.

#### A. Greedy Equal Bandwidth Broadcasting

GEBB uses greedy downloading. See Fig. 7.

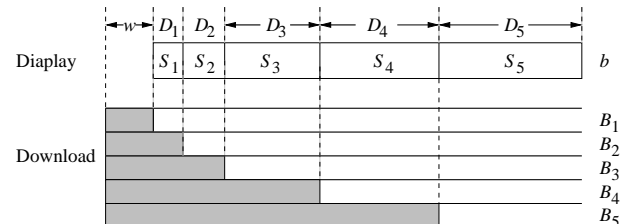


Fig. 7. Illustration of GEBB.

Let us fix the waiting time  $w$  and the number  $n$  of segments (which is the same as the number of channels), and try to minimize the total bandwidth  $\sum_{i=1}^n B_i$ , where  $B_i$  (bits/sec) is the bandwidth of  $C_i$ . From Fig. 7 we get the following set of constraints

$$\begin{aligned} (w + \sum_{j=1}^{i-1} D_j)B_i &= rD_i \text{ for } i = 1, 2, \dots, n \\ \sum_{j=1}^n D_j &= D. \end{aligned} \quad (14)$$

The left hand side of the above equation is the amount of data that can be downloaded from channel  $C_i$  by the time the initial  $i-1$  segments have been displayed. It is equated with the right hand side, which is the amount of data contained in segment  $S_i$ . It is necessary that this much data be preloaded before the display of  $S_i$  can commence, because, in general, when the downloading of  $S_i$  starts (i.e., at the ‘‘tune-in’’ time), channel  $C_i$  will be sending some bit in the middle of  $S_i$ . In the worst case, it will be sending the second bit of  $S_i$ , and the first bit of  $S_i$  won’t be available until  $(rD_i - 1)/B_i$  (seconds) later, i.e., just before the display of  $S_{i-1}$  completes. From Eq. (14), we obtain

$$1 + B_i/r = \frac{w + \sum_{j=1}^i D_j}{w + \sum_{j=1}^{i-1} D_j} \text{ for } i = 1, 2, \dots, n.$$

We thus have

$$\prod_{i=1}^n (1 + B_i/r) = (w + \sum_{j=1}^n D_j)/w = 1 + D/w = 1 + 1/w^*, \quad (15)$$

where  $D$  is the total duration of the video.

**N.B.:** Eq. (14) and all derivations up to this point are generally valid for all greedy schemes, not just for GEBB. Note that Eq. (14) involves  $2n + 1$  unknowns, including  $\{B_i\}, \{D_i\}$  and  $w$ . We can arbitrarily select  $n$  unknowns from among  $\{B_i, D_i : i = 1, 2, \dots, n\}$  and set their values appropriately. We then have  $n + 1$  unknown to be determined by  $n + 1$  equations in Eq. (14). GEBB is the best one among them. **PHB (Polyharmonic Broadcasting [41])** to be discussed later results by making all segments of equal duration (i.e.,  $D_i = \text{constant}$  for all  $i$ ). Thus  $B_i$  decreases harmonically with  $i$ , making PHB suboptimal. ■

Since the product of  $\{1 + B_i/r\}$  is a constant, the sum  $\sum_{i=1}^n (1 + B_i/r) = n + (1/r) \sum_{i=1}^n B_i$ , hence the total bandwidth  $\sum_{i=1}^n B_i$ , is minimized when they are all equal, i.e.,  $B_i = cr$  for some constant  $c$ . In this case, we obtain from Eq. (15),

$$(1 + c)^n = 1 + 1/w^*,$$

which yields

$$c = (1 + 1/w^*)^{1/n} - 1.$$

It follows that the total bandwidth is given by

$$B_{\text{GEBB}} = \sum_{i=1}^n B_i = ncr = nr((1 + 1/w^*)^{1/n} - 1),$$

or

$$B_{\text{GEBB}}^* = n((1 + 1/w^*)^{1/n} - 1). \quad (16)$$

It is seen that  $B_{\text{GEBB}}^* \rightarrow \ln(1 + 1/w^*)$  as  $n \rightarrow \infty$ . In other words, GEBB asymptotically achieves the lower bound given by Eq. (5).

Finally, from Eq. (14), we get

$$\begin{aligned} D_1 &= wcr/r = wc \\ D_2 &= (w + D_1)cr/r = D_1(1 + c) = wc(1 + c) \\ D_i &= D_{i-1}(1 + c) = wc(1 + c)^{i-1} \\ &= w((1/w^* + 1)^{1/n} - 1)(1/w^* + 1)^{(i-1)/n} \end{aligned}$$

The performance of GEBB given by Eq. (16) is plotted in Fig. 8 below for different values of  $n$ . It can be seen that GEBB approaches the lower bound fairly quickly as  $n$  increases.

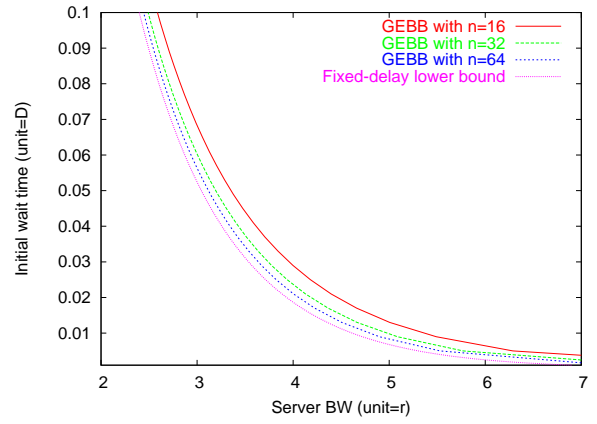


Fig. 8. Performance of GEBB.

To get an idea about the magnitude of the factor  $(1 + 1/w^*)^{1/n}$ , we have computed it for  $w^* = 0.01$  and several values of  $n$ .

$n$	16	32	48	64
$(1 + 1/w^*)^{1/n}$	1.334	1.155	1.140	1.0747

### B. GEBB with prefix caching (GEBB-PC)

Let  $D_0$  denote the duration of the cached prefix. Then the analysis is almost identical to the previous subsection if we set  $w = D_0$ , except that Eq. (15) should be changed to

$$\prod_{i=1}^n (1 + B_i/r) = (D_0 + \sum_{j=1}^n D_j)/D_0 = D/D_0. \quad (17)$$

Thus we get  $D/D_0 = (1 + c)^n$ , hence

$$c = (D/D_0)^{1/n} - 1.$$

The total bandwidth is thus

$$B_{\text{GEBB-PC}} = ncr = nr((D/D_0)^{1/n} - 1)$$

Normalized prefix duration

$$D_0/D = (1 + B/nr)^{-n} = (1 + B^*/n)^{-n} \rightarrow e^{-B^*} \text{ as } n \rightarrow \infty.$$

Finally, we get the closed form formula for  $D_i$  as follows:

$$D_i = D_{i-1}(1 + c) = D_0c(1 + c)^{i-1}$$

Since the user STB may have a rather limited capability in terms of the number of channels from which it can simultaneously download, several schemes, including PyB [52], SkB [26], some version of FDPaB [42], [39], among others, impose a limit on the user bandwidth. Such schemes will be discussed in a later section. Here we present the most efficient scheme among them, GFB.

### C. Generalized Fibonacci Broadcasting

**Generalized Fibonacci Broadcasting (GFB)** is illustrated in Fig. 9. As with many schemes we have seen so far, on the server side, each video is divided into  $n$  segments and each segment  $S_i$  is broadcast on its own dedicated logical channel  $C_i$ . GFB has three configurable parameters,  $n$ ,  $g$  and  $K$ , where  $g \geq 1$  and  $K$  is an integer such that  $1 \leq K \leq n$ . In GFB( $K/g$ ), each channel has equal bandwidth  $r/g$  (i.e.,  $c = 1/g$  in the figure), where  $r$  (bits/sec) is the display rate, which implies that the server bandwidth is  $nr/g$ . The user downloads data from at most  $K$  out of the  $n$  channels. Therefore, the user bandwidth is limited to  $Kr/g$ .

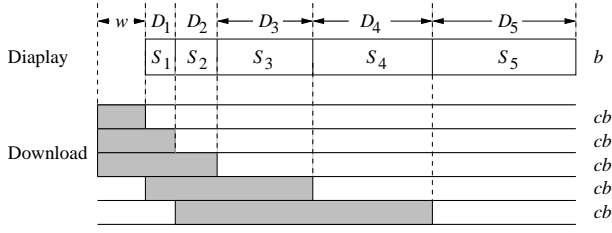


Fig. 9. Illustration of GFB

GFB is a fixed-delay scheme, and at time  $t = 0$ , the user begins to download video segments from the first  $K$  channels. After a fixed delay of  $w$  (seconds), the user can begin to display  $S_1$ . As shown in Fig. 9, at this time, downloading from the first channel stops, as the user STB moves on to download from channel  $C_{K+1}$ , while continuing to download data from channels  $C_2, \dots, C_K$ . In general, for  $i = 1, 2, \dots, n-1$ , the user ceases to receive segment  $S_{i+1}$  when the previous data segment  $S_i$  has been completely displayed.

From Fig. 9, we can derive the continuity conditions of GFB as follows:

$$\begin{aligned}
 D_1 &= (1/g)w \\
 D_2 &= (1/g)(w + D_1) = (1 + (1/g))D_1 \\
 D_3 &= (1/g)(w + D_1 + D_2) = (1 + (1/g))^2 D_1 \\
 &\dots \\
 D_K &= (1/g)(w + D_1 + \dots + D_{K-1}) \\
 &= (1 + (1/g))^{K-1} D_1 \\
 D_{K+1} &= (1/g)(D_1 + \dots + D_K) \\
 &= ((1 + (1/g))^K - 1) D_1 \\
 D_{K+2} &= (1/g)(D_2 + D_3 + \dots + D_{K+1}) \\
 D_{K+3} &= (1/g)(D_3 + D_4 + \dots + D_{K+2}) \\
 &\dots \\
 D_N &= (1/g)(D_{N-K} + D_{N-K+1} + \dots + D_{N-1})
 \end{aligned} \tag{18}$$

The above recurrence relation resembles the generalized Fibonacci sequence. Hence the name Generalized Fibonacci Broadcasting. If we set  $K = 2$  and  $g = 1$ , for example, then each broadcast channel has bandwidth equal to  $r/g = r$ , and the user downloads data from  $K = 2$  concurrent channels, limiting the user bandwidth to  $U = Kr/g = 2r$ . Different choices for  $n$  lead to different waiting time  $w$  and different server bandwidth  $nb/g$ . Independent of the user request time, the waiting time for GFB is given by

$$w = gD_1,$$

where  $D_1$  can be computed by solving the set of equations (18). The configurable parameters,  $g$ ,  $n$ , and  $K$  fine-grained choices, depending on such things as the bandwidth available to the users as well as the server.

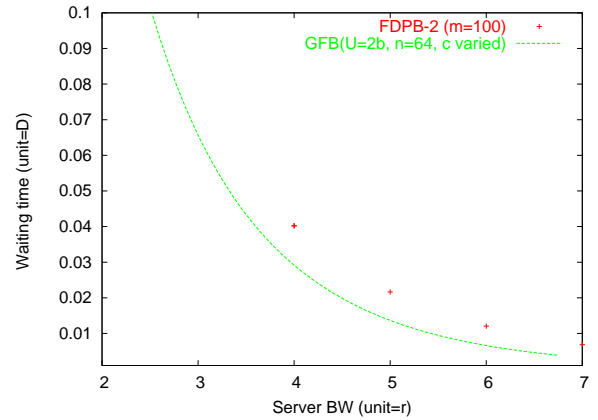


Fig. 10. Performance comparison of GFB( $K/g=2$ ) vs. FDPB-2 ( $m=100$ )

Increasing  $K$  and  $g$  proportionally, while keeping user bandwidth  $Kr/g$  fixed, has the effect of reducing the server bandwidth required to achieve a given waiting time (or reducing the waiting time for a given server bandwidth). Another great advantage of GFB is that it will be straightforward to implement it. For example, GFB( $6/3$ ) with  $n = 21$  broadcasting channels (hence 21 segments) can achieve a shorter waiting time (about 0.6% of the total video length) than FDPB-2 (with parameter  $m = 100$ ), which is one of the most efficient broadcasting schemes with user bandwidth limit that are currently known.

In actual implementation, it may be desirable to choose an integer as  $g$ . Then an existing channel with bandwidth  $r$  could be time-division multiplexed into  $g$  logical sub-channels relatively easily. Last but not least, GFB gives fine-grained choices for the server and user bandwidths, provided by different combinations of  $K$  and  $g$ .

## IV. CLASSIFICATION

Now that we know a few broadcasting schemes, let us start classifying them according to the techniques that they use. We know that some schemes use fixed delay in the beginning, while other start as soon as the beginning of  $S_1$  is encountered in  $C_1$ .

### A. Slotted schemes

This class consists of most earlier schemes, including Pyramid, FB, Harmonic, Cautious Harmonic, Quasi Harmonic, etc. Here is a partial list of known broadcasting schemes. User BW has a non-empty entry if the maximum number of channels from which the user needs to download at any given time is less than the server BW. A “full-rate” segment is broadcast at a rate  $\geq r$ , once it is started (but the frequency varies, depending on the segment).

Name	fixed delay	slotted	full-rate segments	user BW limit
GEBB	Yes			
GFB	Yes			Yes
PHB	Yes	Yes		
FDFB	Yes	Yes	All	
FDPB	Yes	Yes	All	Yes
PyB			All	$2\alpha r$
FB		Yes	All	
CHB		Yes	$S_1, S_2, S_3$	
QHB		Yes	$S_1$	
PaB		Yes	All	
NPaB		Yes	All	
StB		Yes	?	
SkB		Yes	All	$2r$

GEBB = Greedy Equal Bandwidth Broadcasting, GFB = Generalized Fibonacci Broadcasting, PHB = Polyharmonic Broadcasting, FDFB = Fixed-delay Fast Broadcasting, FDPaB = Fixed-delay Pagoda Broadcasting, PyB = Pyramid Broadcasting, FB = Fast Broadcasting, CHB = Cautious Harmonic Broadcasting, QHB = Quasi Harmonic Broadcasting, PaB = Pagoda Broadcasting, NPaB = New Pagoda Broadcasting, StB = StairCase Broadcasting, SkB = Skyscraper Broadcasting.

### B. Harmonic-based schemes

Harmonic Broadcasting (HB) was proposed in [28]. Its basic idea is shown in Fig. 11.

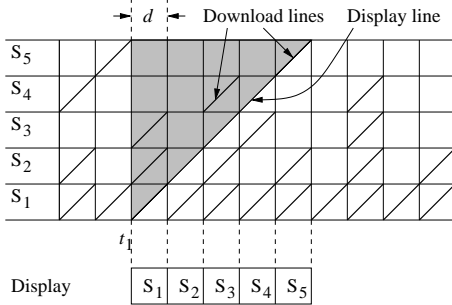


Fig. 11. Basic idea of HB:  $S_i$  is broadcast in every  $i^{\text{th}}$  slot.

Suppose the entire video is divided into  $n$  segments of equal duration  $d = D/n$ . As shown in the download-display diagram of Fig. 11, suppose for now that for  $i = 2, 3, \dots$ ,  $C_i$  has bandwidth  $r$ , and  $S_i$  is broadcast in every  $i^{\text{th}}$  slot in channel  $C_i$ . Since this is a slotted scheme, both the display

of  $S_1$  and the downloading from other channels start when the beginning of  $S_1$  is encountered in  $C_1$ . Notice that any horizontal line drawn across the download-display diagram intersects a download line in the shaded area. This implies that all segments can be downloaded in time for display.

Clearly, using every  $i^{\text{th}}$  slot in a channel of bandwidth  $r$  provides the same bandwidth as a dedicated channel of bandwidth  $r/i$ . So, as in the original HB, let us use such a channel for broadcasting  $S_i$ . Unfortunately, this does not work, as pointed out by Pâris et al. in [40]. To see why, let us examine Fig. 12.

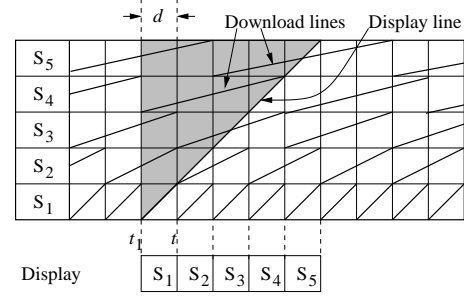


Fig. 12. Download-display diagram for HB: Starvation can occur.

Suppose display starts at time  $t_0$ . For segment  $S_2$ , for example, at time  $t$ , only its second half has been put in buffer, and the first half is not available. For a similar reason, display will starve for segments  $S_3$  and  $S_5$ , as can be seen from Fig. 12.

Pâris et al. fixed this problem by broadcasting  $S_2$  and  $S_3$  alternately in  $C_2$ , as shown in Fig. 13 [40]. This scheme is called **Cautious Harmonic Broadcasting (CHB)**. For  $i \geq 4$ ,  $S_i$  is broadcast in channel  $C_i$  with bandwidth  $r/(i-1)$ . It is easy to see that this arrangement works. The total bandwidth is obtained by adding the bandwidths of  $n$  channels.

$$B_{\text{CHB}} = r + r + \sum_{i=4}^n r/(i-1) = rH_n + r(1/2 - 1/n), \quad (19)$$

where  $H_n$  is the  $n^{\text{th}}$  harmonic number.

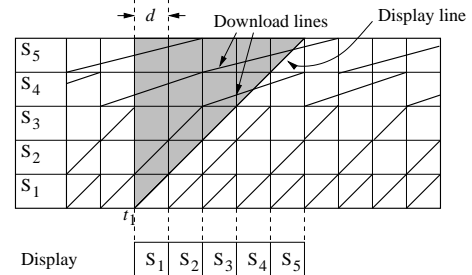


Fig. 13. Download-display diagram for CHB.

To improve on the bandwidth requirement of CHB, Pâris et al. use a more elaborate scheme called **Quasi Harmonic Broadcasting (QHB)** [40]. As illustrated in Fig. 14, QHB with parameter  $q \geq 2$  (integer) divides each time slot into  $q$  equal subslots.

For  $i = 2, \dots, n$ ,  $C_i$  uses  $iq$  subslots (i.e.,  $id$  seconds) to broadcast  $S_i$ . However,  $S_i$  is subdivided into  $iq - 1$ , not  $iq$ ,

**fragments.** Let  $S_{i,1}, S_{i,2}, \dots, S_{i,iq-1}$  denote those fragments. These  $iq-1$  fragments are allocated to  $iq$  subslots as follows. Consider the **group** of  $iq$  subslots in  $i$  consecutive slots in  $C_i$ . The  $q^{\text{th}}, 2q^{\text{th}}, \dots, iq^{\text{th}}$  subslots ( $i$  subslots in total) in this group are designated as *special*. The special subslots of successive groups are used to broadcast the first  $i-1$  fragments,  $S_{i,1}, S_{i,2}, \dots, S_{i,i-1}$ , repeatedly in this order. Therefore, these  $(i-1)$  fragments appear in any consecutive  $(i-1)$  time slots. This ensures that when the first  $(i-1)$  segments  $S_1, S_2, \dots, S_{i-1}$ , have been displayed in  $(i-1)d$  seconds, the first  $(i-1)$  fragments of  $S_i$  have been downloaded, so that they are ready to be displayed.

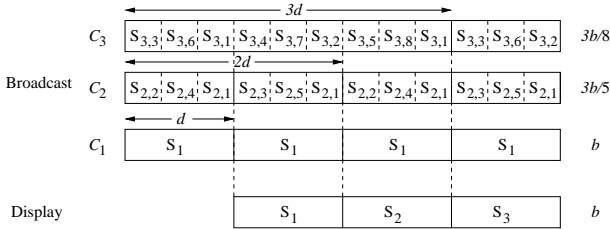


Fig. 14. QHB with parameter  $q = 3$ : each time slot is divided into  $q$  subslots, and the last subslot in each time slot is special.

Now, there are  $iq - i = i(q-1)$  non-special subslots in each group, and there are  $iq - 1 - (i-1) = i(q-1)$  unallocated fragments. Note that each group of subslots occupies  $i$  consecutive time slots. The first unallocated fragment,  $S_{i,i}$ , is placed in the first subslot in the first time slot, the second unallocated fragment,  $S_{i,i+1}$ , is placed in the first subslot in the second time slot, etc., until the  $i^{\text{th}}$  unallocated fragment,  $S_{i,2i-1}$ , is placed in the first subslot in the last (or  $i^{\text{th}}$ ) time slot. The  $i+1^{\text{st}}$  unallocated fragment,  $S_{i,2i}$ , is placed in the 2nd subslot in the first time slot, and so forth, as shown in Fig. 14 for the case  $q = 3$ .

In Fig. 15, short solid line segments indicate that all fragments of  $S_3$ , except  $S_{3,1}$  and  $S_{3,2}$ , appear in every 9th subslot, while  $S_{3,1}$  and  $S_{3,2}$  appear in every 6th subslot. Each short solid line segment shows when the bits of a fragment appear, from the first bit to the last bit. The long line shows the data in  $S_3$  consumed, if the display of  $S_1$  starts at  $t_0$ , finishing at  $t_1$  and the display of  $S_2$  starts at  $t_1$ , finishing at  $t_2$ . From the figure, we can see that all fragments of  $S_3$ , except  $S_{3,3}$  and  $S_{3,6}$ , during the last two time slots since the display of  $S_1$  started. Fragments  $S_{3,3}$  and  $S_{3,6}$  are loaded concurrently with their display.

In general, a user starts downloading/displaying at the beginning of a time slot, say  $t_1$ , and starts displaying  $S_i$   $(i-1)$  time slots later, i.e., at time  $t_1 + (i-1)d$ . During these  $(i-1)$  time slots,  $(i-1)q$  out of the  $iq-1$  fragments belonging to  $S_i$  are encountered and downloaded from channel  $C_i$ . In other words, only  $q-1$  fragments from  $S_i$  are still missing. The worst case is where the missing fragments are those that are needed in the nearest future, i.e.,  $S_{i,i}, S_{i,2i}, \dots, S_{i,(q-1)i}$ . In other words, the time slot in  $C_i$  containing these fragments has not been seen yet. The first  $(i-1)$  fragments, i.e.,  $S_{i,1}, S_{i,2}, \dots, S_{i,i-1}$ , must be available, since they appear in every consecutive  $(i-1)$  time slots. Therefore, they can be displayed

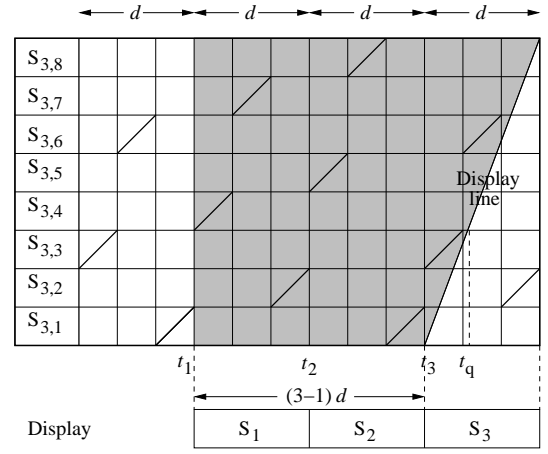


Fig. 15. Download-display diagram for QHB ( $S_3$ ).

starting at  $t_1 + (i-1)d$ . The first  $i$  fragments of  $S_i$ , including  $S_{i,i}$ , together contain  $idr/(iq-1)$  bits of data, and therefore it will be  $t_1 + (i-1) + id/(iq-1)$  when they are all displayed. This time is shown as  $t_q$  in Fig. 15, where  $q = i = 3$ . Since  $id/(iq-1) > d/q$ , if  $S_{i,i}$  was not available at  $t_1 + (i-1)d$ , it will have been downloaded in time for display.

To compute the bandwidth required by QHB, note that each fragment of  $S_i$  contains  $dr/(iq-1)$  bits of data. This amount of data is broadcast in each subslot of duration  $d/q$ . Therefore, the bandwidth required is given by  $(dr/(iq-1))/(d/q) = rq/(iq-1)$ , which means this is the bandwidth of  $C_i$ . The bandwidth of  $C_2$  in Fig. 14, for example, is  $3r/(2 \times 3 - 1) = 3r/5$ . Since  $rq/(iq-1) = r/(i-1/q)$ , this bandwidth approaches  $r/i$  as  $q$  grows larger. Therefore, we expect the total bandwidth to approach  $rH_n$  for large  $q$ . Note that

$$\frac{rq}{iq-1} = \frac{r}{i} + \frac{r}{i(iq-1)}.$$

We thus have

$$B_{\text{QHB}}(n) = r + \sum_{i=2}^n \frac{rq}{iq-1} = rH_n + \sum_{i=2}^n \frac{r}{i(iq-1)} \quad (20)$$

In summary, the greatest advantage of QHS is that its bandwidth requirement approaches the lower bound (for the FSP schemes)  $rH_n$  as  $q \rightarrow \infty$ . Its disadvantages are that each page is divided into a different number of fragments and that each channel requires a different amount of bandwidth that is not an integer multiple of some basic unit. All this makes QHS not easily implementable.

The last member of the class of Harmonic-based broadcasting schemes is **Polyharmonic Broadcasting (PHB)** [41] mentioned earlier. PHB is a fixed-delay scheme but at the same time uses time slots as well. However, it does not rely on the time slots in any essential way. As commented in Section III, where we discussed GEBB, PHB is best understood in terms of Fig. 7, where  $w = md = mD/n$ . In solving the set of equations Eq. (14), GEBB chose to make all channel bandwidths  $\{B_i\}$  equal because that led to the minimum total bandwidth. If we make all segment durations  $\{D_i\}$  equal instead by setting  $D_i = d$  for all  $i$ , we end up with PHB.

Then the solution to Eq. (14) is given by

$$B_i = \frac{r}{m+i-1}.$$

From  $w = mD/n$ , we have  $nw^* = m$ , where  $w^* = w/D$  is the normalized wait time as in Section III. The bandwidth required by PHB is obtained by

$$B_{\text{PHB}}(n) = \sum_{i=1}^n \frac{r}{m+i-1} = r(H_{n+m-1} - H_{m-1}).$$

Let

$$G_i = \frac{1}{m+i-1} + 1 = \frac{1}{nw^*+i-1} + 1 = \frac{nw^*+i}{nw^*+i-1}.$$

Then

$$\sum_{i=1}^n G_i = B_{\text{PHB}}(n)/r + n = B_{\text{PHB}}^*(n) + n,$$

where  $B_{\text{PHB}}(n)$  and  $B_{\text{PHB}}^*(n)$  are the bandwidth and the normalized bandwidth of PHB, respectively. Note that

$$\prod_{i=1}^n G_i = \frac{nw^*+n}{nw^*} = 1 + 1/w^*.$$

From the inequality between arithmetic and geometric means, we get

$$\left(\prod_{i=1}^n G_i\right)^{1/n} \leq (1/n) \sum_{i=1}^n G_i = B_{\text{PHB}}^*(n)/n + 1,$$

and therefore,

$$(1 + 1/w^*)^{1/n} \leq B_{\text{PHB}}^*(n)/n + 1.$$

From this it follows that

$$B_{\text{GEBB}}^*(n) \leq B_{\text{PHB}}^*(n),$$

where  $B_{\text{GEBB}}^*(n) = n((1 + 1/w^*)^{1/n} - 1)$  from Eq. (16). Fig. 16 below plots the performance of PHB and GEBB for the case where  $n = 32$  or  $64$ . Note that there are only a small number of data points for PHB, since  $m$  must be an integer.

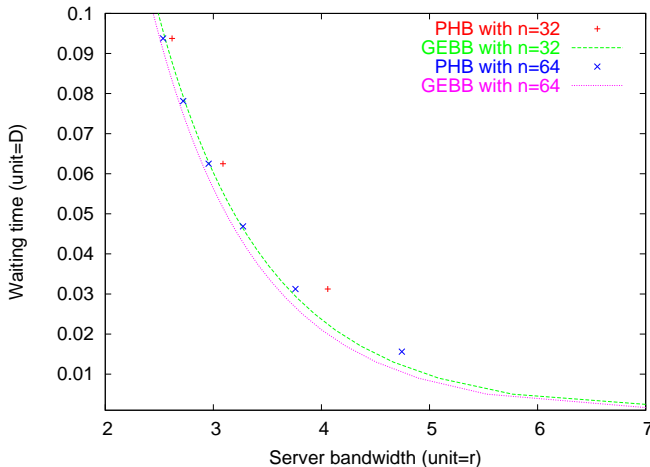


Fig. 16. Comparison of PHB with GEBB ( $n = 32$  and  $64$ ).

TABLE II  
PAGODA BROADCASTING (PaB) USING THREE CHANNELS

Slot	1	2	3	4	5	6	...
Channel 1	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	...
Channel 2	$S_2$	$S_4$	$S_2$	$S_5$	$S_2$	$S_4$	...
Channel 3	$S_3$	$S_6$	$S_8$	$S_3$	$S_7$	$S_9$	...

### C. Pagoda Broadcasting Schemes

Harmonic-based schemes have relatively low bandwidth requirement, but managing a large number of streams of decreasing rates is unwieldy. To resolve this problem, Pâris et al. proposed **Pagoda Broadcasting (PaB)** [41]. Later, Pâris extended it to a more elaborate version called **New Pagoda Broadcasting (NPaB)** [38].

Since all Pagoda-related schemes use channels with bandwidth equal to the display rate  $r$ , and each segment fits exactly into a time slot, the continuity requirement is that segment  $S_i$  should be broadcast at least every  $i^{\text{th}}$  time slot. Therefore, in Table IV-C,  $S_1$  is broadcast repeatedly in channel  $C_1$ . Since  $S_2$  need be broadcast in every second time slot in the next channel,  $C_2$ , every other slot of  $C_2$  is available for other segments.  $S_4$  is a perfect fit, since it can occupy every  $4^{\text{th}}$  slot in this channel. Even though  $S_5$  need be broadcast in every  $5^{\text{th}}$  time slot, PaB broadcasts it in every  $4^{\text{th}}$  slot of what's left in  $C_2$ .

PaB divides odd-numbered channels, except for channel 1, into three subchannels, and even-numbered channels into two subchannels (apparently for no particularly good reason). Therefore, there are 1+3+2 substreams in Table IV-C. However, five distinct logical channels are required for Channel 3, since five non-consecutive segments are interleaved. In the terminology of this paper, therefore, 1 + 3 + 5 = 9 subchannels are needed. In other words, the management overhead of this scheme is equivalent to having nine channels. The paper shows that 29 (49) segments can be broadcast with 4 (5) channels using bandwidth  $4r$  ( $5r$ ). Therefore, with 5 streams for example, PaB can pack 49 segments into 5 channels, which means that the segment size, hence the maximum wait time, is  $D/49$ .

In Table IV-C, if segment  $S_i$  is placed in channel  $k$ , then all segments  $S_j$  in channel  $k-1$  satisfies  $j < i$ . In other words, optimization was carried out only locally. For example, to avoid wasted bandwidth in  $C_2$ , segments  $S_8, S_{16}, S_{32}$ , etc., instead of  $S_5$ , could be packed into  $C_2$ . However, this leaves  $S_5$  to be broadcast in some other channel. In general, the problem of minimizing the number of channels required to broadcast a given number of segments satisfying the frequency requirement of each segment has been shown to be NP-complete [3].

**New Pagoda Broadcasting**[38] tries to put as many segments as possible into up to 7 streams of bandwidth  $r$  each by an *ad hoc* (non-systematic) method. Stream 1 carries  $S_1$  in every slot of channel  $C_1$ .  $C_2$  carrying stream 2 is divided into four logical subchannels of bandwidths  $r/2, r/4, r/8$ , and  $r/8$ , respectively.  $S_2$  is broadcast in the first subchannel, in other words, in every  $2^{\text{nd}}$  slot.  $S_4$  is broadcast in the second subchannel, i.e., in every  $4^{\text{th}}$  slot in  $C_2$ . Finally  $S_8$  and  $S_9$  are

broadcast in the third and fourth channels, respectively, i.e., in every 8<sup>th</sup> slot. Thus,  $S_2$ ,  $S_4$ , and  $S_8$  are broadcast at exactly the required frequencies, while  $S_9$  is broadcast slightly more frequently than is necessary. [38] uses a  $2 \times 2$  matrix shown below to represent this allocation.

$S_2$	$S_4$
...	$S_8$
...	$S_4$
...	$S_9$

In general, if  $S_i$  is the segment with the smallest index  $i$  that has not been assigned to any channel so far, we construct a matrix with  $i$  columns. Each column represents a subchannel consisting of every  $i^{\text{th}}$  slot of a channel of bandwidth  $r$ .  $S_i$  will occupy the entire first column, indicating that it uses bandwidth  $r/i$ .  $S_{2i}$  will occupy half of the second column, indicating that it uses bandwidth  $r/2i$ . Unfortunately, the assignment of the remaining segments is *ad hoc*.

$S_3$	$S_6$	$S_{12}$
...	$S_7$	$S_{13}$
...	...	$S_{14}$
...	...	$S_{25}$
...	...	$S_{12}$
...	...	$S_{13}$
...	...	$S_{14}$
...	...	$S_{26}$

The above table has eight rows, indicating that channel 3 has a period of  $3 \times 8$  slots.  $S_3$  occupies 1/3 of these slots, i.e., 8 slots per period.  $S_6$  and  $S_7$  occupy 1/6 each of these slots, i.e., four slots per period.  $S_{12}$ ,  $S_{13}$  and  $S_{14}$  occupy 1/12 each of these slots, i.e., two slots per period. Finally  $S_{25}$  and  $S_{26}$  each occupy 1/24 of these slots, i.e., one slot per period. We haven't assigned segments  $S_5$ ,  $S_{10}$ , etc., since we intend to use more than three channels and they will be assigned to later channels. If we were to use only three channels, then instead of segments  $S_{25}$  and  $S_{26}$ , we must fit  $S_5$ ,  $S_{10}$  and  $S_{11}$  somewhere.

Recently, Bar-Noy and his coworkers have formulated a combinatorial optimization problem, called *windows scheduling* problem, which includes PaB as a special case. Their most recent results, which considerably improve on NPAB for higher bandwidth, are summarized in the table below. The first row is the number of streams each broadcast at the display rate  $r$ . The upper bounds are obtained from the maximum  $k$  such that  $H_k \leq n$ , where  $H_k$  is the  $k^{\text{th}}$  harmonic number. The best bounds are from [4], and *Greedy* values are from the heuristic algorithm in [4].

Windows scheduling problem has been generalized to the *block windows scheduling* problem [50]. Its solutions can exceed even the upper bound given in Table IV-C.

## V. SCHEMES WITH USER BANDWIDTH LIMIT

### A. Pyramid Broadcasting

As we stated in Introduction, PyB was the first novel broadcasting introduced in 1995. In Section II, we mentioned that PyB requires  $D_{i-1}B_i \geq D_i r$ . Let  $D_i = (B_i/r)D_{i-1} =$

TABLE III  
PERFORMANCE OF NEW PAGODA BROADCASTING (NPAB)

$n$	1	2	3	4	5	6	7	8
Upper bound	1	3	10	30	82	226	615	1673
Best bound	1	3	9	28	77	211	570	1573
<i>Greedy</i>	1	3	9	25	73	201	565	1522
NPAB	1	3	9	26	66	172	442	499

$\alpha D_{i-1}$ , where  $\alpha = B_i/r$ . To be able to download  $S_i$  which is  $\alpha$  times longer than  $S_{i-1}$  in  $D_{i-1}$ ,  $B_i$  must be also  $\alpha$  times larger than  $B_{i-1}$ . From  $\sum_{j=1}^n D_j = D$ , we get

$$D_1 = D \frac{\alpha - 1}{\alpha^n - 1}.$$

Since the average wait time  $w = D_1/2\alpha$ , we have

$$w = D \frac{\alpha - 1}{2\alpha(\alpha^n - 1)},$$

and hence

$$w^* = \frac{\alpha - 1}{2\alpha(\alpha^{B^*/\alpha} - 1)} = \frac{\alpha - 1}{2\alpha((\alpha^{1/\alpha})^{B^*} - 1)}, \quad (21)$$

since  $B^* = n\alpha$ . It can be shown that  $\alpha^{1/\alpha}$  takes the maximum value for  $\alpha = e$  (the base of natural logarithm). However, since the exponent  $B^*$  is relatively small for the cases of interest, Eq. (21) is minimized for  $\alpha \leq 2$ . Fig. 17 compares the performance of PyB for various parameter values with that of FB. The data points for several different values of  $\alpha$  correspond to the cases where  $B^*/\alpha$  is an integer, representing the number of channels used. The curve labeled " $B^* = 2\alpha$ " plots Eq. (21). Therefore,  $B^*/\alpha = 2$  for all the points on the curve and any such point is realizable in theory with two channels. The curve labeled " $\alpha = e$ ", for example, plots Eq. (21) for  $\alpha = e$ . Those points on this curve for which  $B^*/\alpha$  is an integer are realizable (with  $B^*/\alpha$  channels). The formula  $D_{i-1} = D_i r/B_i$  implies the fact that the user needs to download from only two channels at a time, since the beginning of  $S_i$  will be encountered after the display of  $S_{i-1}$  started.

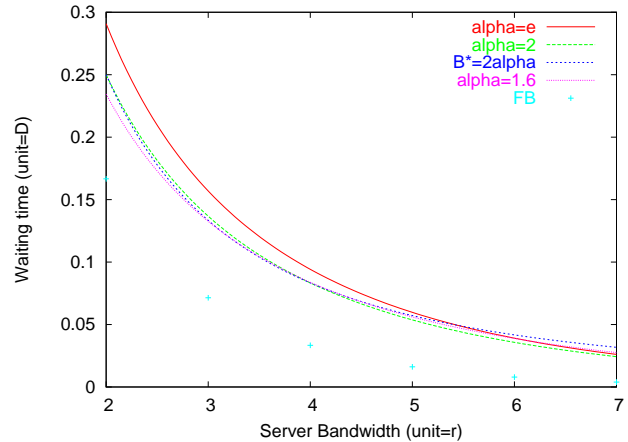


Fig. 17. Comparison of PyB with FB. In this figure, the horizontal (vertical) axis is the server bandwidth (wait time).

We now discuss another scheme with user bandwidth limit.

### B. Skyscraper Broadcasting

As we saw above, the segment durations of PyB form a geometric series  $[d, \alpha d, \alpha^2 d, \dots]$ . In SkB, the series is defined by  $[f(1)d, f(2)d, \dots] = [d, 2d, 2d, 5d, 5d, 12d, 12d, 25d, 25d, 52d, 52d, \dots]$ , where

$$f(i) = \begin{cases} 1 & i = 1 \\ 2 & i = 2, 3 \\ 2f(i-1) + 1 & i \bmod 4 = 0 \\ f(i-1) & i \bmod 4 = 1 \\ 2f(i-1) + 2 & i \bmod 4 = 2 \\ f(i-1) & i \bmod 4 = 3 \end{cases} \quad (22)$$

Thus  $d = D_1 = D / \sum_{i=1}^n f(i)$ , where  $n$  is the number of segments. As in PyB, the maximum waiting time  $w = d$ , and hence  $w^* = w/D = 1 / \sum_{i=1}^n f(i)$ .

### C. Fixed-delay PaB with user bandwidth limit

This scheme divides a video into a number of segments of equal duration  $d$ . For  $j = 1, 2, \dots$ , stream  $j$  is broadcast using bandwidth  $r$ , but it is multiplexed among  $s_j$  substreams, each of which is broadcast on its own logical channel.

If the user bandwidth is limited to  $2r$  and only three streams are used, for example, the following table shows that this scheme uses  $10+16+21=47$  logical channels, dividing the video into 1268 segments. Since  $m = 100$  in the table, the initial delay is  $w = md$ , which means the normalized waiting time is  $w^* = 100d/1268d = 0.0789$ , as shown in the last column.

Stream	Number of channels	First segment	Last segment	$w^*$
$\sigma_1$	10	$S_1$	$S_{156}$	0.641
$\sigma_2$	16	$S_{157}$	$S_{565}$	0.177
$\sigma_3$	21	$S_{566}$	$S_{1268}$	0.0789
$\sigma_4$	27	$S_{1269}$	$S_{2486}$	0.0402
$\sigma_5$	36	$S_{2487}$	$S_{4617}$	0.0217
$\sigma_6$	47	$S_{4618}$	$S_{8298}$	0.0121
$\sigma_7$	62	$S_{8299}$	$S_{14595}$	0.00685

Its performance was compared with that of GFB with user bandwidth  $2r$  in Fig. 10.

Using some heuristic, the first stream is divided into 10 substreams which uses bandwidth  $r/10$  each. Since  $w = 100d$ ,  $S_1$  needs to be put in every  $100^{th}$  slot, or every  $10^{th}$  slot in substream 1. The ten segments from  $S_1$  to  $S_{10}$  are thus placed in substream 1. The next segment  $S_{11}$  needs to be broadcast at least in every  $m+11-1 = 110^{th}$  slot, or every  $11^{th}$  slot in substream 2. The 11 segments from  $S_{11}$  to  $S_{21}$  are thus placed in substream 2. Repeating this process, the 23 segments from  $S_{134}$  to  $S_{156}$  are placed in substream 10, which is the last substream of Stream 1. The result is shown in row 1 of the above table. Row 2 can be computed analogously.

As for row 3, again, some heuristic determines the number of substreams (21) to be used. The first segment to be assigned to substream 1 of stream 3 is  $S_{566}$ . Note that initially the user can download only two streams. Therefore, the downloading of  $S_{566}$  cannot start until all segments of stream 1, in particular its last segment  $S_{156}$ , have been downloaded. As we observed

above, segments  $S_{134}$  to  $S_{156}$  are broadcast in substream 10 of stream 1. Since  $m+134-1 = 233$ ,  $S_{134}$  must be broadcast at least in every  $233^{th}$  slot in stream 1, or every  $\lceil 233/10 \rceil = 23^{rd}$  slot in substream 10. So,  $S_{156}$  is also broadcast in every  $23^{rd}$  slot in substream 10, or every  $230^{th}$  slot in stream 1. This implies that the downloading of  $S_{156}$  may finish as late as  $t = 230d - 100d$  in the worst case, where time  $t$  is measured from the display start time. On the other hand, the first segment of steam 3, i.e.,  $S_{566}$  must be available by  $t = 566d$  at the latest. It follows that  $S_{566}$  must be broadcast at least every  $566-(230-100)=436^{th}$  slot, or every  $\lceil 436/21 \rceil = 20^{th}$  slot in substream 1 of stream 3.

### REFERENCES

- [1] C.C. Aggarwal, J.L. Wolf and P.S. Yu, On optimal piggyback merging policies for video-on-demand systems, *ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, Philadelphia, May 1996, pp. 200–209.
- [2] C.C. Aggarwal, J.L. Wolf and P.S. Yu, A permutation based Pyramid broadcasting scheme for video-on-demand systems, *Proc. IEEE Int'l Conf. on Multimedia Computing and Systems (ICMC)*, Hiroshima, Japan, June 1996.
- [3] A. Bar-Noy, R. Bhatia, J. Naor, and B. Shieber, Minimizing service and operation costs of periodic scheduling, *Proc. 9<sup>th</sup> Annual ACM-SIAM Symp. on Discrete Algorithms (SODA '98)*, 1998, 11–20.
- [4] A. Bar-Noy and R.E. Ladner, Windows scheduling problems for broadcast systems, *Proc. 13<sup>th</sup> ACM-SIAM Symposium on Discrete Algorithms*, SODA 2002, pp. 433–442. [Formulates the segment-to-channel allocation problem as a combinatorial problem, and presents a good heuristic. Lists some open problems.]
- [5] A. Bar-Noy, A. Nisgiv, and B. Patt-Shamir, Nearly optimal perfectly-periodic schedules, *Proc. 20<sup>th</sup> ACM Symp. on Principles of Distributed Computing (PODC '01)*, 2001, pp. 107–116.
- [6] A. Bar-Noy, R. E. Ladner, and Tami Tamir, Scheduling Techniques for Media-on-Demand, *ACM SODA 2003*, pp. 791–80.
- [7] Y. Cai, K.A. Hua and K. Vu, Optimizing patching performance, *Proc. SPIE Conf. on Multimedia Computing and Networking (MMCN)*, San Jose, CA, Jan. 1999, pp. 204–215.
- [8] Y. Cai and K.A. Hua, An efficient bandwidth-sharing technique for true video on demand systems, *Proc. 7<sup>th</sup> ACM Multimedia Conf.* Orlando, FL, Nov. 1999, pp. 211–214.
- [9] S.W. Carter and D.D.E. Long, Improving video-on-demand server efficiency through stream tapping, *Proc. 6<sup>th</sup> Int'l Conf. on Computer Communications and Networks (ICCCN)*, Las Vegas, NV, Sept. 1997, pp. 200–207.
- [10] S.W. Carter, D.D.E. Long and J.-F. Páris, Video-on-demand broadcasting protocols, In *Multimedia Communications: Directions and Innovations* (Gibson, J.D., Ed.), Academic Press, San Diego, 2000, pp.179–189.
- [11] A. Dan, D. Sitaram and P. Shahabuddin, Scheduling policy for an on-demand video server with batching, *Proc. ACM Multimedia '94*, San Francisco, CA, Oct. 1994, pp. 15–23.
- [12] D.L. Eager and M.K. Vernon, Dynamic Skyscraper Broadcasting for video-on-demand, *Proc. 4<sup>th</sup> Int'l Workshop on Multimedia Information Systems (MIS)*, Istanbul, Turkey, Sept. 1998, pp. 18–32.
- [13] D.L. Eager, M.C. Ferris and M.K. Vernon, Optimal regional caching for on-demand data delivery, *Proc. SPIE Conf. on Multimedia Computing and Networking (MMCN)*, San Jose, CA, Jan. 1999, pp. 301–316.
- [14] D.L. Eager, M.K. Vernon and J. Zhorjan, Minimizing bandwidth requirements for on-demand data delivery, *Proc. 5<sup>th</sup> Int'l Workshop on Multimedia Information Systems (MIS)*, Indian Wells, CA, Oct. 1999, pp. 80–87.
- [15] D.L. Eager, M.K. Vernon and J. Zhorjan, Optimal and efficient merging schedules for video-on-demand servers, *Proc. 7<sup>th</sup> ACM Int'l Multimedia Conf.*, Orlando, FL, Nov. 1999, pp. 199–202.
- [16] D.L. Eager, M.K. Vernon and J. Zhorjan, Bandwidth skimming: a technique for cost-effective video-on-demand, *Proc. SPIE Conf. on Multimedia Computing and Networking (MMCN)*, San Jose, CA, Jan. 2000,
- [17] M.W. Garrett and A. Fernandez, Telcordia Technologies, Inc. <ftp://ftp.research.telcordia.com/pub/vbr.video.trace/MPEG.data>.

- [18] L. Gao and D. Towsley, Supplying instantaneous video-on-demand services using controlled multicast, *Proc. IEEE Int'l Conf. on Multimedia Computing and Systems (ICMC)*, Florence, Italy, June 1999.
- [19] L. Gao, Z.L. Zhang and D. Towsley, Catching and selective catching: efficient latency resuction techniques for delivering continuous multimedia streams, *Proc. 7th ACM Int'l Multimedia Conf.*, Orlando, FL, Nov. 1999, pp. 203–206.
- [20] L. Engebretsen and M. Sudan, Harmonic broadcasting is bandwidth-optimal assuming constant bit rate, *Proc. 13-th Annual@ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2002.
- [21] Hollmann and Holzschcher, *Philips Tech. Rept.* 1991, European Patent (1991) and US patent #5524271 (1995).
- [22] A. Hu, Video-on-demand broadcasting protocols: A comprehensive study, *Proc. INFOCOM '01*, Anchorage, AK, April 2001, pp. 508–517. [A good overview of principles behind broadcasting schemes, including those for VBR videos.]
- [23] A. Hu, I. Nikolaidis and P. van Beek, On the design of efficient video-on-demand broadcast schedules, *Proc. 7th Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '99)*, 1999, pp. 262–269. [Proposes GEBB.]
- [24] K.A. Hua, Y. Cai and S. Sheu, Exploiting client bandwidth for more efficient video broadcast,
- [25] K.A. Hua, Y. Cai and S. Sheu, Patching: a multicast technique for true video-on-demand services, *Proc. ACM Multimedia*, Bristol, UK, Sept. 1998.
- [26] K.A. Hua and S. Sheu, Skyscraper broadcasting - A new broadcasting scheme for metropolitan video-on-demand systems, *Proc. ACM SIGCOMM '97*, Cannes, France, Sept. 1997, pp. 89–100. [Proposes Skyscraper Broadcasting.]
- [27] R. Janakiraman, M. Waldvogel and L. Xu, Fuzzycast: efficient video-on-demand over multicast,
- [28] L. Juhn and L. Tseng, Harmonic broadcasting protocols for video-on-demand service, *IEEE Trans. on Broadcasting* 43, Sept. 1997, pp. 268–271. [Introduces Harmonic Broadcasting (HB). It doesn't work as advertised. Later modified by Paris et al. in PCL98a.]
- [29] L. Juhn and L. Tseng, Fast data broadcasting and receiving scheme for popular video service, *IEEE Trans. on Broadcasting* 44 (1), Mar. 1998, pp.100–105. [Proposes FB.]
- [30] L. Juhn and L. Tseng, Enhanced Harmonic data broadcasting and receiving scheme for popular video service, *IEEE Transactions on Consumer Electronics* 44, May 1998, pp. 343–346. [Unfortunately, EHB also has a bug similar to the original HB.]
- [31] T. Kameda, Y. Sun and L. Goddyn, An optimization problem related to VoD broadcasting, *Proc. ISAAC 2005*, Springer-Verlag, pp. 116–125. Hainan, Dec. 2005. [Discusses optimization of FDPB. Shows that the search space for optimal number of subchannels can be limited to  $O(\sqrt{m})$ , where  $m$  is the initial delay parameter.]
- [32] T. Kameda and S. Wu, A lossless VOD broadcasting scheme for VBR videos using standard channels, *Int'l Conf. on Distributed Computing Systems (ICDCS) '04*, submitted. [Introduces a backward segmentation scheme for VBR videos.]
- [33] S.W. Lau, J.C.S. Lui and L. Golubchik, Merging video streams in a multimedia storage server: Complexity and heuristics, *ACM Multimedia Systems Journal* 6, 1 (Jan. 1998), pp. 29–42.
- [34] F. Li and I. Nikolaidis, Trace-adaptive fragmentation for periodic broadcast of vbr video, *Proc. 9th Int'l Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '99)*, Basking Ridge, NJ, July 1999.
- [35] F. Li and I. Nikolaidis, SCB: Staircase broadcast for media-on-demand systems, *Proc. IEEE Mobile Multimedia Communications (MoMuC'00)*, 3B-1-1-3B-1-5. [A segmentation heuristic for VBR videos.]
- [36] I. Nikolaidis, F. Li and A. Hu, An inherently lossless and bandwidth efficient scheme for periodic broadcast of vbr video, *Proc. ACM SIGMETRICS '00*, 2000, pp.116–117.
- [37] J.-F. Pâris, A broadcasting protocol for compressed video, *Proc. EU-ROMEDIA '99 Conf.*, Munich, April 1999, pp. 78–84.
- [38] J.-F. Pâris, A simple low-bandwidth broadcasting protocol for video-on-demand, *Proceedings of the 8th International Conference on Computer Communications and Networks (ICCCN'99)*, Boston-Natick, MA, Oct. 1999 pp. 118–123. [Proposes New PaB.]
- [39] J.-F. Pâris, A fixed-delay broadcasting protocol for video-on-demand, *Proc. 10th Int'l Conf. on Computer Communications and Networks*, 2001, pp. 418–423. [Discusses FDPB with/without user bandwidth limit.]
- [40] J.-F. Pâris, S.W. Carter and D.D.E. Long, Efficient broadcasting protocols for video on demand, *Proc. 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98)*, July 1998, pp. 127–132. [Discusses CHB and QHB and compares them with PyB, HB, and Skyscraper.]
- [41] J.-F. Pâris, S.W. Carter, and D.D.E. Long, A low-bandwidth broadcasting protocol for video-on-demand, *Proc. 7th Int'l Conf. on Computer Communications and Networks (ICCCN '98)*, Oct. 1998, pp. 690–697. [Proposes PHB.]
- [42] J.-F. Pâris and D.D.E. Long, Limiting the receiving bandwidth of broadcasting protocols for video-on-demand, *Proc. Euromedia '00*, pp. 107–111.
- [43] J.-F. Pâris and D.D.E. Long, The case for aggressive partial preloading in broadcasting protocols for video-on-demand, *Proc. 2001 IEEE Int'l Conf. on Multimedia and Expo*, Tokyo, Aug. 2001, pp.113–116.
- [44] J.-F. Pâris, S.W. Carter and P.E. Mantey, Zero-delay broadcasting protocols for video-on-demand, *Proc. 1999 ACM Multimedia Conf.*, Orlando, FL., Nov. 1999, pp. 189–197. [Introduced the idea of prefix caching, called partial preloading.]
- [45] O. Rose, Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems, *Tech. Rept. 101*, Universitaet Wuerzburg, Feb. 1995. [Has many MPEG2 video traces. Only initial 20–30min. portions.]
- [46] J. Salehi, Z. Zhang, J. Kurose and D. Towsley, Supporting store video: Reducing rate variability and end-to-end resource requirements through optimal smoothing, *IEEE/ACM Tran. on Networking* 6 (4), Aug. 1998, pp. 397–410.
- [47] D. Saporilla, K. Ross and M. Reisslein, Periodic broadcasting with vbr-encoded video, *Proc. IEEE INFOCOM '99*, pp. 464–471.
- [48] S. Sen, L. Gao, J. Rexford and D. Towsley, Optimal patching schemes for efficient multimedia streaming, *Proc. 9th Int'l Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '99)*, Basking Ridge, NJ, July 1999.
- [49] W. Stallng, *Data & Computer Communications*, Sixth Ed., Upper Saddle River, NJ: Prentice-Hall, 1999. [A popular text book on communications.]
- [50] Yi Sun and T. Kameda, Harmonic block windows scheduling through Harmonic windows scheduling, *Proc. International Workshop on Multimedia Information Systems (MIS)*, Springer-Verlag, Sorrento, Sept. 2005, pp. 190–206. [Tries to find a near-optimal HBW schedule, starting with a HW schedule.]
- [51] K. Thirumalai, J.-F. Pâris and D. D. E. Long, Tabbycat: an inexpensive scalable server for video-on-demand, *Proceedings of the IEEE 2003 International Conference on Communications (ICC 2003)*, Anchorage, AK, May 2003, pp. 896–900. [Reports on an experimental implementation of FDPB.]
- [52] S. Viswanathan and T. Imielinski, Pyramid Broadcasting for video on demand service, *Proc. IEEE Conf. on Multimedia Computing and Networking*, Vol. 2417, San Jose, CA, 1995, pp. 66–77. [The paper which started it all.]
- [53] S. Viswanathan and T. Imielinski, Metropolitan area video-on-demand service using pyramid broadcasting, *Multimedia Systems* 4, no. 4, 1996, pp. 197–208.
- [54] M. Waldvogel and R. Janakiraman, Efficient media-on-demand over multiple multicast groups, *Proc. Globecom '01*, [Introduces a new metric, called “network cost”]
- [55] E.M. Yan E.M. and T. Kameda, An efficient VOD broadcasting scheme with user bandwidth limit, *Proc. SPIE/ACM Conf. on Multimedia Computing and Networking*, Vol. 5019, Santa Clara, CA, January 2003, pp. 200–208. [Discusses Generalized Fibonacci Broadcasting.]